MASIMO: AUTONOMOUS QUALITY INSPECTION IN AVIATION MRO AND MANUFACTURING USING ASSET ADMINISTRATION SHELL, DIGITAL PRODUCT PASSPORT AND ECLIPSE DATASPACE CONNECTOR

M. Weiss[+),*)], E. Grishchenko[+)], F. Raddatz[+)], G. Wende[+)]
[+)] Deutsches Zentrum für Luft- und Raumfahrt, DLR-MO, Hamburg, Germany

Abstract

In the age of data-driven aviation, enabling seamless, standardized, and sovereign collaboration among all participating stakeholders remains a critical challenge. Central to this is the task of ensuring and restoring product conformity – spanning from quality assurance during manufacturing to non-conformity resolution in MRO operations and manufacturing. A promising approach to this challenge is the concept of self-managed, digitally represented assets in the form of Industry 4.0 Asset Administration Shells that autonomously coordinate their own quality-related services within choreographed, federated data spaces. The paper demonstrates this vision through an analysis of a jet-engine blade use case, confirming feasibility of federated inspection while revealing limits in registration accuracy and sensor resolution.

1. INTRODUCTION AND MOTIVATION

In modern aviation manufacturing and maintenance ecosystems, ensuring high-quality standards across distributed value networks remains a critical challenge. As the complexity and heterogeneity of systems increase, so does the demand for traceable, reliable, and autonomous quality assurance processes [33]. Traditional quality inspection strategies – often reliant on tightly coupled infrastructures or human coordination – struggle to scale in dynamic, multi-organizational environments [22]. This is particularly true in Maintenance, Repair and Overhaul (MRO), where components such as turbomachinery blades undergo multiple lifecycle transitions, change service locations, and require frequent requalification. Without a standardized, interoperable digital backbone, quality-related data often remains isolated, impeding both conformance verification and effective corrective action [9]. A promising solution combines three emergent Industry 4.0 technologies:

- **Asset Administration Shell** (AAS): a standardized digital representation of assets to manage their data (including quality-relevant information) [7].
- **Inspection-as-a-Service** (IaaS): an architecture to modularize and outsource quality inspection tasks as on-demand services.
- **Data Spaces**: federated data ecosystems to ensure secure, sovereign cross-company data exchange.

This approach is supported by various publications. For instance, [2] and [22] each proposed a standardized quality data submodel (SM) within the AAS for product features, tolerances, and inspection results, enabling autonomous quality control loops through a common machine-interpretable language. Meanwhile, [20] demonstrated that offering NDI (Non-Destructive Inspection) through a cloud-based marketplace can lower barriers for Small Medium Enterprises (SMEs) and improve data traceability and trust, especially when paired with smart contracts and pay-per-use models. These findings illustrate emerging business models where inspection solution providers offer NDI services via flexible contracts.

Complementing these developments, data-space architectures such as International Data Spaces (IDS) and Gaia-X have emerged to enforce data sovereignty and interoperability at scale. In this context, [9] identified data spaces as key enablers for cross-company quality collaboration, highlighting their role in breaking down integration barriers while protecting intellectual property. Industrial initiatives such as Manufacturing-X further demonstrate the feasibility of combining AAS with the Eclipse Dataspace Connector (EDC) to enable distributed, inspection-related service orchestration across federated networks.

The vision of a federated, self-managed inspection system requires assets to actively participate in their own quality assurance processes. This requires that both tangible components and intangible services have operational representations that are semantically interoperable, lifecycle-aware, and securely networked. The AAS standard [7], which emerged from the Plattform Industry 4.0 Reference Architecture [6], fulfills this requirement by providing a machine-interpretable digital representation of a physical or intangible asset. When extended to its proactive variant, the AAS not only holds descriptive metadata and service interfaces, but can also initiate and coordinate inspections autonomously – using standardized interaction protocols such as those defined in the VDI norm “Language for I4.0” [26] [27]. Recent publications have further underlined the need for autonomous, interoperable inspection processes. Grunau et al. [59] demonstrated the potential of proactive AAS instances to autonomously coordinate service tasks; however, their implementation was limited to logistics scenarios and lacked inspection-specific SMs, dataspace governance, and lifecycle traceability. Parallel efforts by Fraunhofer [22] and KIT [2] proposed structured AAS-based quality models and semantic interoperability layers, yet did not address orchestration across federated systems or event-triggered coordination. Recent analyses of AAS types [33] confirmed that only proactive (Type 3) AASs offer sufficient autonomy and reasoning for decentralized collaboration, but empirical implementation examples remain scarce.

*) Corresponding author: Marco.Weiss@DLR.de

 doi: 10.25967/650135

To facilitate secure, policy-compliant data exchange among these digital representatives, a Dataspace Connector (such as the Eclipse Dataspace Connector) provides a governance layer that respects organizational sovereignty while enabling cross-domain orchestration. Inspection workflows are further grounded in the use of structured SMs such as the Digital Product Passport (DPP) [29] and an embedded SM Quality Control for Machining (QCM) [21], which capture technical details, inspection parameters, and traceability information. Embedding these models in lifecycle-aware coordination sequences allows both passive and active components to be seamlessly integrated into automated workflows.

Building on these insights, this paper proposes a modular architecture for autonomous quality inspection, combining proactive AAS entities, structured SM templates, and event-based communication models conforming to the Industry 4.0 language and interaction protocols. An exemplary implementation is presented using a low-pressure compressor (LPC) blade, for which inspection is coordinated across scanning systems, path-planning tools, and evaluation services. The implementation respects real-world constraints of dataspace governance and multi-party interaction. The proposed approach builds on experimental system architectures previously described in [30], [31], [29], and extends them with:

- a detailed inspection sequence model,
- SM structuring for capability & skill exchange, and
- persistent asset traceability throughout the inspection lifecycle.

The architecture integrates proactive AASs (Type 3) with inspection-specific SMs such as ServiceRequestNotification [12], CapabilityDescription (currently not available, but described here [28] and [4]), and QualityControlMachining [21], which formalize service needs and enable automated matching with provider capabilities. By embedding these SMs in I4.0-compliant messages (e.g.“Call for Proposal” (CfP)), the system enables fully machine-interpretable service orchestration. The coordination is lifecycle-aware: inspection requests may originate during design verification, post-production qualification, in-service anomaly detection, or routine maintenance. By using permanent traceability identifiers – as opposed to variable serial or asset IDs – this approach ensures consistent inspection records across organizational boundaries.

In contrast to prior implementations limited to intra-factory logistics or passive monitoring, the system described here enables multi-party, event-driven coordination of inspection services through proactive AASs operating within governed data-space environments. It thus represents a concrete instantiation of a federated, autonomous inspection infrastructure aligned with Industry 4.0 principles. Inspection services are composed dynamically, triggered by lifecycle events, and governed through secure EDC interfaces – positioning the approach as a functional realization of autonomous inspection in distributed Industry 4.0 ecosystems. The research questions addressed in this study are:

1. How can proactive AASs autonomously orchestrate inspection workflows across federated, policy-governed data spaces?
2. Which AAS submodel structures enable machine-interpretable negotiation, capability matching, and lifecycle traceability in quality inspection?
3. How can interconnected inspection assets – such as scanning, planning, and evaluation systems – be composed as AAS-based services to realize autonomous, federated quality control?

2. ARCHITECTURE AND CHOREOGRAPHY

2.1. System-Level Architecture

Industrial quality inspection in distributed manufacturing and MRO data ecosystems demands a system architecture that can accommodate both passive and active physical components, enabling their self-managed participation in autonomous workflows in dataspaces. In such federated ecosystems, assets and services must interact seamlessly across organizational and technical boundaries while upholding data sovereignty and interoperability standards. This vision is supported by the MX-Port Concept [8], released 2025 under Factory-X as part of the broader Manufacturing-X initiative. In this context, the MX-Port provides a five-layered reference model (**Table 1**) for enabling trusted data exchange and orchestration across participants in industrial data spaces.

	Layer	Purpose
L5	Discovery	enabling participant and asset registration and matchmaking
L4	Access & Usage Control	enforcing data policies and usage constraints
L3	Gate	providing harmonized APIs and standard interfaces
L2	Converter	mapping data to common information models,
L1	Adapter	interfacing with heterogeneous operational technology

Table 1: Five layers of the MX-Port concept [8]

In this data ecosystem, stakeholder cooperation takes place within a hybrid infrastructure of both centralized and decentralized components, as illustrated in Figure 1. These components interact via standardized information models (e.g. AAS at the Converter layer) and offer unified interfaces for two-way communication toward both the asset side (L1) and the dataspace (L3), which are bridged by a converter mechanism (L2). This structure enables dynamic discovery of services (L5) and governed data transfer with policy enforcement (L4) across company boundaries.

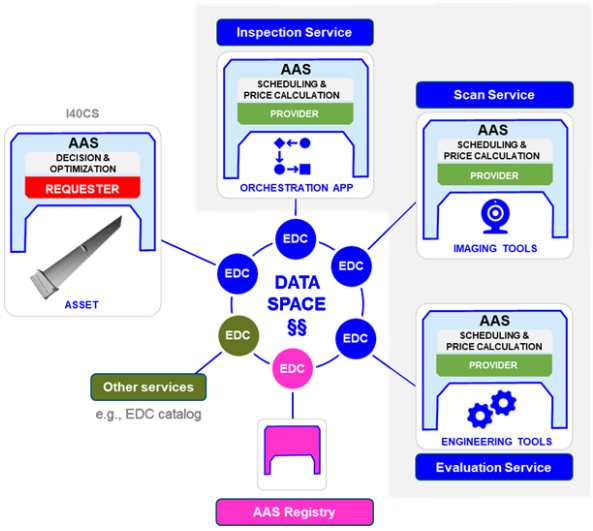


Figure 1: I4.0CS interactions in federated dataspaces

To realize the architecture, the authors have been developing an Industry 4.0 Component Stack (I4.0CS), which is a

set of modules that, when combined, make an asset or service dataspace-ready and semantically interoperable. All participating systems are upgraded to Industry 4.0 Components (I4.0C) – each consisting of an asset coupled with its AAS. At a high level, every I4.0CS (whether representing a product requesting an inspection or a tool/service providing it) is built from the same set of interoperable building blocks centered on a proactive Type 3 AAS. It serves as a structured, semantically defined, and autonomous digital representative of its asset by managing service requests and responses and negotiates contracts. It enables even passive assets to become active participants in digitally governed processes, while also integrating service-providing assets into a common semantic and operational framework. The AAS can play different roles across the asset's lifecycle or value chain. In early stages, it serves primarily as a digital container for design and production data. During operational use or maintenance, it becomes an active entity – making decisions based on real-time inputs (from the system in which it is embedded), coordinating required inspection services, and embedding resulting quality data directly into the asset's DPP as described in [29]. Through this lifecycle-aware approach, assets are endowed with autonomy without needing to embed complex intelligence in each physical component – ensuring scalability and compliance with industry constraints.

The I4.0CS architecture follows the layered guidelines of the Industrial Digital Twin Association. Similar to frameworks like Eclipse BaSyx [23] and Fraunhofer's FA³ST [18], it separates the AAS core from protocol-specific endpoints. However, unlike BaSyx – which requires external customization or orchestration to achieve proactive behavior – this implementation natively integrates and manages state-machine-based interaction logics to realize a fully proactive Type 3 AAS [61][62][59]. In practice, the DLR I4.0 Component Stack consists of five interoperable components which directly align with the MX-Port layers 1 to 4:

- **Asset**: the physical piece of equipment or logical entity being represented.
- **Asset Administration Shell** (Type 3): the digital twin of the asset, including data models and embedded autonomous logic. (L1, L2, L3)
- **Asset Data Server** (ADS): a middleware layer that interfaces with the asset's hardware/data, performing data acquisition and preprocessing. (L1, L2)
- **Eclipse Dataspace Connector** (EDC): the component enabling secure data exchange and contract-based communication with external parties. (L4)
- **Control Server** (CS): a supervisory control component for overarching coordination or hosting of the AAS runtime.

The combination of this components reflects and extends architectural patterns seen in platforms like BaSyx or FA³ST (which support mainly reactive Type 2 AAS). Type 1 and Type 2 AASs are increasingly common in industry, whereas Type 3 AASs remain rare and mostly confined to experimental demonstrators [10]. Traditional frameworks often require external orchestrators or hard-coded behaviors to achieve autonomy. By contrast, the I4.0CS provides an embedded Type 3 implementation since the proactive logic is built into the AAS runtime itself. Internally, event-driven state machines allow the AAS to interpret conditions and initiate actions independently, while externally the AAS can engage in peer-to-peer interactions and negotiations using the standard I4.0 communication language. To this end, the AAS not only statically represents the asset's state but also integrates autonomous functions for context-dependent decision-making and self-managed negotiation. The AAS additionally features control capabilities to directly command or adjust its corresponding asset as needed.

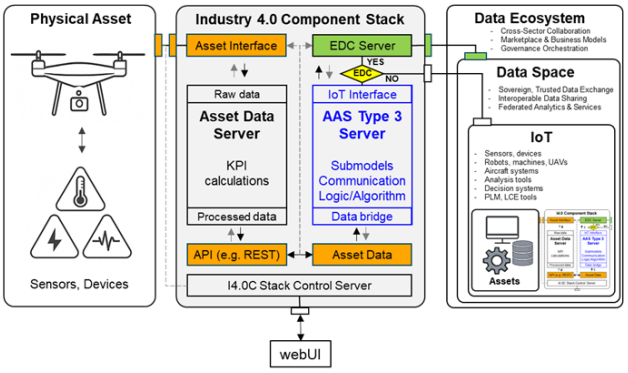


Figure 2: DLR Industry 4.0 Component Stack I4.0CS

Complementing the AAS, an Asset Data Server (ADS) provides a flexible abstraction layer between the asset and its digital shell. An ADS is typically required in two situations: (1) when an asset does not natively support network interfaces or IoT protocols (e.g., a legacy machine or a PLC-controlled system), and (2) when the asset's raw data streams need preprocessing (filtering, unit conversion, aggregation, analytics) before used by the AAS or other services. In many MRO and manufacturing scenarios, the ADS initializes a digital interface, converting raw input/output (I/O) or analogue signals into standardized, semantically annotated data. This provides the first opportunity for digital visibility of traditionally disconnected devices, which is critical in brownfield environments. The ADS performs four key functions for asset data:

- Data acquisition from physical interfaces (e.g., GPIO, fieldbus, PLC signals), thereby bringing offline or analog data into the digital realm.
- Preprocessing of raw data (e.g., noise suppression, scaling and unit conversion, timestamping, and basic analytics).
- Data exposure via standard communication protocols such as OPC UA, MQTT, CoAP, or REST, making the data accessible in a uniform way.
- Event-driven publishing of updates, so that the AAS and other subscribed services can receive real-time notifications of changes.

Upstream, the AAS either subscribes to or pulls data from ADS feeds, mapping these inputs into its internal models. This ensures consistency across the digital twin's data and mirrors the behaviour of similar mechanisms in other platforms. Conceptually, this is comparable to the 'Asset Connection' in FA³ST or the Data Bridge in Eclipse BaSyx. Depending on complexity, the ADS itself can be implemented in various forms:

- A (Python) script that reads serial sensor data, applies scaling, and publishes via standard protocols.
- A streaming pipeline using frameworks like Apache StreamPipes, as presented for FA³ST in [17], Kafka, or NiFi, which handle ingestion, preprocessing, and data distribution

Finally, the EDC serves as a provider and consumer on behalf of the asset and its AAS by facilitating sovereign data exchange via policy-enforced contracts. The EDC provider acts as a kind of proxy, transferring AAS data through its data-plane once a dataspace contractual agreement with a counterpart has been established, e.g., with an AAS registry or other participants.

2.2. Service Interaction Choreography

The service choreography in the previously introduced dataspace is demonstrated with an autonomous inspection of a low-pressure compressor (LPC) blade. We will prepare this scenario by outlining the sequence of interactions shown in Figure 4: The architecture enables multiple distributed assets to form a collaborative unit, with each participant realized as an AAS-driven component fulfilling a specific role: the product asset, a Quality Inspection Service Station (QSS) orchestrator, and various tooling services (e.g., scanning and evaluation systems). Standardized interaction protocols and semantic information models ensure that service discovery, capability matching, and contract negotiation among these participants are fully machine-interpretable. All service outcomes – such as deviation reports or quality metrics – are intended to be recorded into the blade's Digital Product Passport (DPP), maintaining end-to-end lifecycle traceability of quality data. As introduced in previous work [29], the DPP serves in general as a container for essential inspection-related SMs (e.g., nameplate, CAD models, requirement specifications), including a Quality Control for Machining (QCM) to cover inspection-specific data, requirements and results (Figure 3).

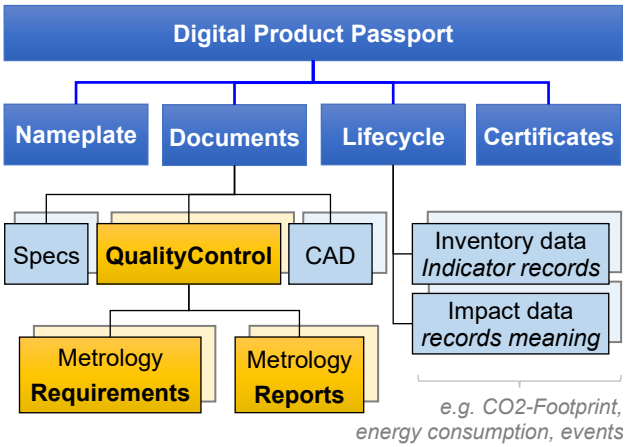


Figure 3: DPP [29] extended with inspection's documentation with a *Quality Control* submodel

As shown in Figure 4 left, the LPC blade acts as a service requester, while a higher-level Quality Service Station (QSS) acts as the inspection service provider. The QSS itself then becomes a requester for subordinate services like scanning and evaluation (these nested interactions correspond to the greyed region in Figure 1). Notably, the blade is a passive item with no built-in electronics or sensing; however, its AAS persists throughout the blade's lifecycle and becomes functionally active once the blade is part of a larger system capable of monitoring it (for example, installed in an engine or attached to a test rig). Those embedding systems provide the necessary data streams and control interfaces to the blade's AAS, allowing it to participate meaningfully in the inspection process.

The inspection process formally begins when the blade's AAS issues a CfP into the federated data space. This can follow two strategies: either as a broadcast to all registered providers, supporting open and dynamic matchmaking; or as a targeted message to a preselected recipient, enforcing the use of specific providers, e.g., pre-selected tools. The interaction sequence diagram shown in Figure 4 reflects the second strategy, as it omits the broadcast and registry lookup steps for the paper's clarity.

The request is structured as an I4.0 message, consisting of a header (with metadata such as sender and receiver IDs, message type, timestamps, etc.) and an interaction payload specifying the service requirements (Figure 5, right side). At the core of this payload, we have decided to use the *IDTA Service Request Notification SM* [12], which encapsulates the technical and contextual details of the request. This is supplemented in its detailed section by a *BiddingOrder SM*, which contains commercial and technical details of the service request, and is introduced in [1] and [25]. Within the *BiddingOrder SM*, the technical collection is organized into a functional block (Figure 5, right middle). This includes a *requiredCapabilities* (e.g., "Inspection") and a *Requirements* collection that lists the elements needed to fulfil the requested capability. They are instantiated as part of a

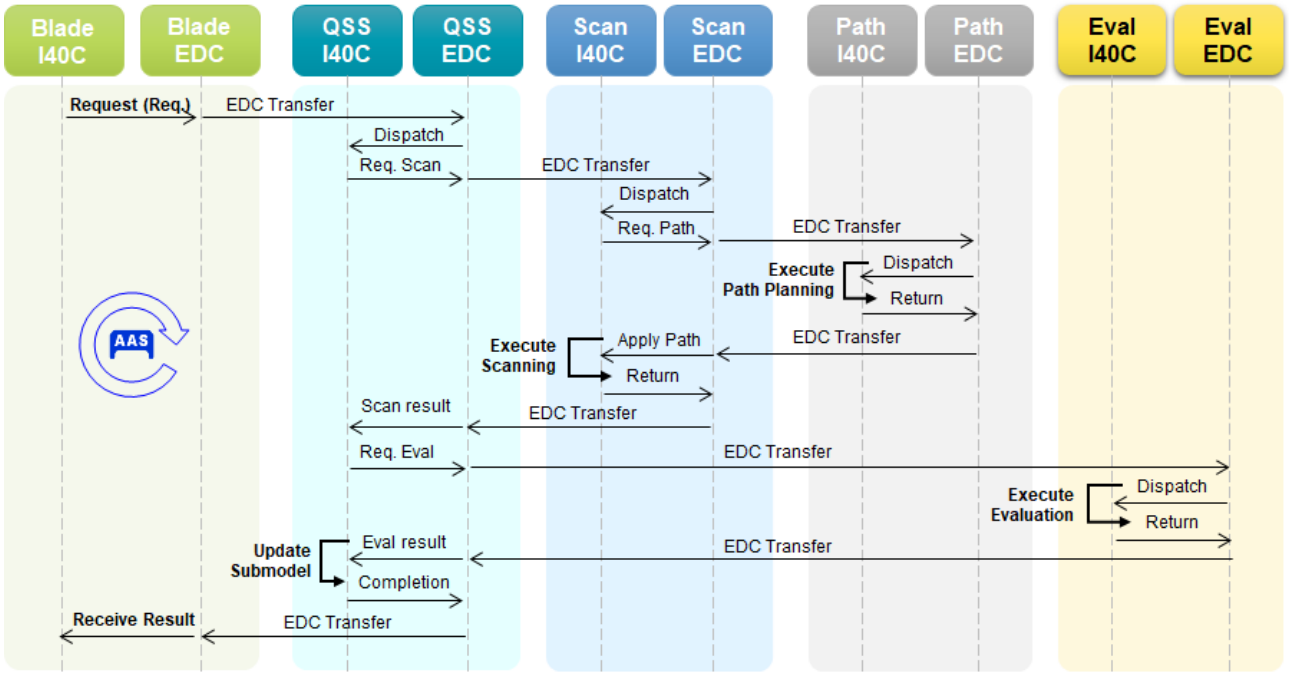


Figure 4: Top-level inspection service sequence diagram with the LPC blade as requester on the left

BillofProcess (BoP) SM, representing the planned configuration and execution logic. The relevant capability instance and its parameters are then appended to the *BiddingOrder SM* from the BoP to communicate the service expectations.

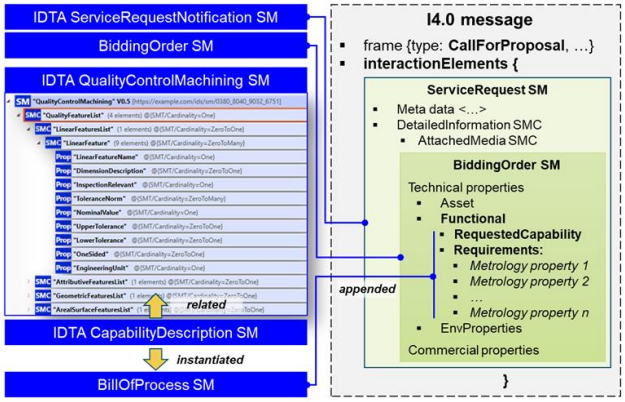


Figure 5: I4.0 message as CfP

The underlying structure and semantics of the capability and its parameters are defined in a *CapabilityDescription* SM (e.g. IDTA 02020), which standardizes it across the data ecosystem. These parameters formally describe the service conditions and expectations – such as required inspection resolution and surface coverage, necessary service inputs (e.g., CAD model or scan trajectory), expected outputs (e.g., 3D mesh and deviation report), and context conditions (e.g., in situ or off-wing execution). Some of these parameters originate from other SMs – here in particular, the *QualityControlMachining (QCM)* SM [21], which specifies the metrology features for the asset.

When a registered Quality Service Station (QSS) receives the CfP, its AAS evaluates the request against the capabilities ("*Is the requested capability available?*") and skills ("*Can it be executed under the specified features?*") of the service assets. If the request is feasible, the QSS returns a proposal describing how the inspection would be carried out – including supported data formats, required resources, delivery time, and expected costs. The blade's AAS automatically evaluates all received proposals and, if one of the offers meets its requirements, sends an acceptance message to finalize the agreement.

Once accepted, the blade's AAS will provide the QSS with controlled access to selected portions of its Digital Product Passport (DPP), such as CADs. The method by which access control will be managed has not yet been decided – it may be managed solely by the AAS, via connector-based, policy-enforced data exchange, or via a combination of both. For now, we are granting full access for our experimental verification: The QSS retrieves the necessary context information for service execution – including the CAD geometry, material composition, and specified inspection requirements. In certain cases, the QSS may also issue a secondary Call for Proposal (CfP) to suppliers for auxiliary services, such as scan-path planning or specialised evaluation tools. This links service provision across participants in the federated data space, even if the data space is established locally (e.g. company-wide).

The first stage of the service is the 3D scanning of the blade. The QSS uses its EDC consumer interface to engage a scanning service AAS and negotiate a data exchange contract. Once access is granted, the QSS sends the scan request, which includes the blade's identifier and the necessary input data (such as the blade's CAD reference model). The scanning service may require a precomputed robot path to perform the measurement. If so, it can invoke auxiliary planning tools – for instance, a camera position planner to determine optimal viewpoints based on the CAD geometry, followed by a path planner to generate a continuous motion trajectory covering those viewpoints. Guided by the resulting scan trajectory, the scanning system captures the blade's geometry and produces a digital 3D model of the part (e.g., an STL mesh file). The completed scan data is then transferred back to the QSS through the dataspace.

Next, the QSS initiates the evaluation stage by invoking an evaluation service AAS. The QSS establishes a contract and transmits the required inputs – namely, the 3D scan data of the blade, the blade's original CAD model (as the nominal reference), and the blade's identifier – via the EDC to the evaluation service. The evaluation service aligns the scan with the reference model and computes the deviation for each feature specified in the QCM, comparing the measured values against their allowable tolerances. It then generates a structured result set indicating whether each inspected feature is within tolerance (pass/fail) and returns this results package to the QSS.

Finally, the QSS parses the inspection data according to the result properties defined in the QCM SM template. Each feature entry in the QCM is populated with its measured value, an indicator of whether it passed within tolerance and, where necessary, references to supporting evidence (such as the scan file or analysis report). These updated properties, which are uniquely identifiable by semanticID, are returned to the blade's AAS. The AAS then attaches them to the DPP's QCM, e.g. a traceable timeseries, completing the autonomous inspection workflow.

3. INSPECTION-AS-A-SERVICE

In this chapter, we apply the concepts introduced earlier by demonstrating how existing assets – such as a robot-based inspection cell and a digital twin of a LPC blade – are incrementally advanced to Industry 4.0 Components (I4.0C) using the I4.0 Component Stack (I4.0CS). This enables their seamless participation in an inspection dataspace through standardized interaction, negotiation, and service execution mechanisms.

3.1. General Implementation

The technical implementation of the Inspection system begins with the upgrade of all participating assets to Industry 4.0 (I4.0C) components. This transformation presumes a functioning network infrastructure and proceeds by creating both the digital representation of the asset and the corresponding data connection to support monitoring and control functionalities. The data backbone of the digital representation is modeled using the AAS server, which is hosted in the I40 Component Stack (I40CS). Its connection to the physical asset is established via the Asset Data Server (ADS), which either collects sensor signals directly from the asset or provides virtualized access points in case of non-instrumented components.

The first step in setting up the system involves creating a digital representation of the asset to be inspected: a compressor blade from the RB199 turbojet engine acting as the service requester. For the basic identification, characterization, process or historical data for the blade, obligatory

submodels are the IDTA Nameplate, IDTA TechnicalDescription, IDTA 3Dprovision, QualityControlForMachining (QCM) and BoP, all of them are also organized as a Digital Product Passport. As shown in Figure 6 right, in the runtime environment they are parsed for further data processing such as in the state machines which most important are:

- The Event Manager SM continuously monitors preset properties against thresholds (e.g. start_trigger = True; parameter_2 = 'INSPECT'), initiating tasks such as inspections or manufacturing processes via the Production Manager SM.
- The Production Manager SM: Executing and supervising the BoP by preparing and calling for proposals, selecting and finally placing the order in terms of requested capabilities to the service provider, for each step of the BoP.

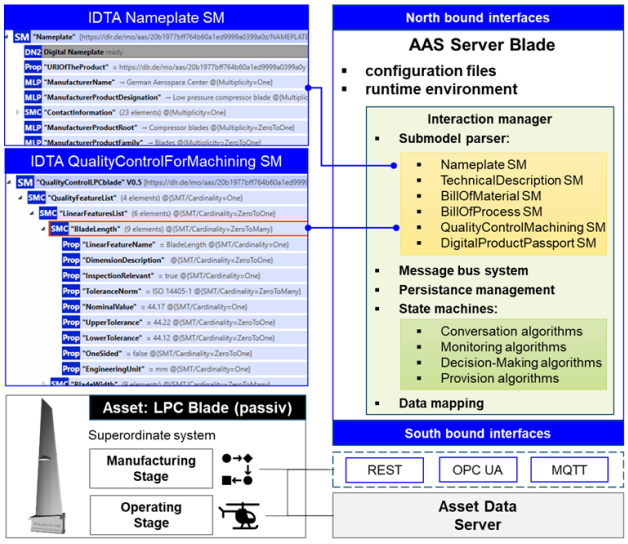


Figure 6: AAS Server Blade

As the AAS also requires runtime data of its blade to manage it properly over the life-cycle stages, but the blade is a component without active data interfaces, the ADS is configured to receive or proxy contextual data from its superordinate system, such as the aircraft engine control unit during use phase, or the manufacturing cell during production stage (Figure 6 bottom left). This data is semantically bound to the AAS properties using a declarative mapping described in the Asset Interface Mapping Configuration (AIMC), in accordance with the IDTA-02027 specification [16]. The interfaces themselves are described through the Asset Interface Description (AID) submodel IDTA-02017 [15], which formalizes the available endpoints, protocols, and security constraints.

In our use case, we run an OPC UA Server that pretends to be a superordinate system of the passive LPC blade. This server provides the Event Manager with the variable *start_trigger* as an initial flag. When this flag is combined with the *parameter_1* or *parameter_2* variables, it either starts the blade manufacturing or the quality inspection interaction protocol. Both tasks correspond to either multiple (BoP) or single service requests containing all the metadata necessary for task execution and the requested capability, which is forwarded to the receiving service provider such as the Quality Service Station (QSS). As described in Chapter 2.2, the order is sent as a service request notification including a set of properties containing technical and commercial data. The inspection's capability description contains runtime-instantiated metrology features based on the QCM template and superordinate properties that all inspection capabilities have in common, regardless of their implemented skills. These include bounding box data, referenced drafts (CAD), positioning coordinates, the asset's dimensions, and the primary material it is made of, among others. The service provider checks the request's completeness and rejects it if mandatory parameters are missing. It also rejects the request if the parameter set is complete but the required values fail the feasibility check of the implemented skill(s).

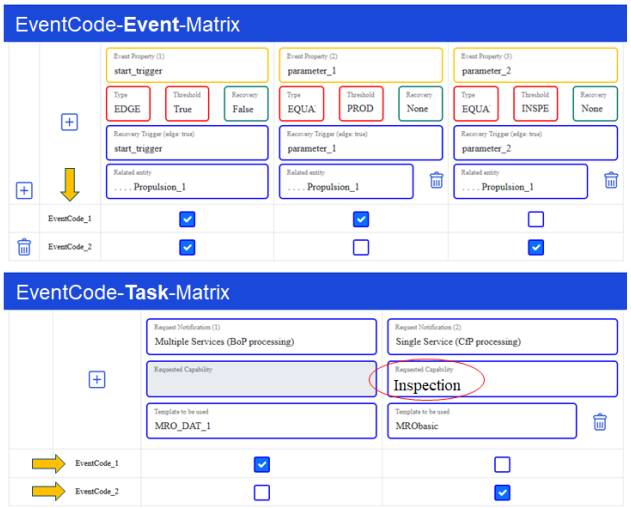
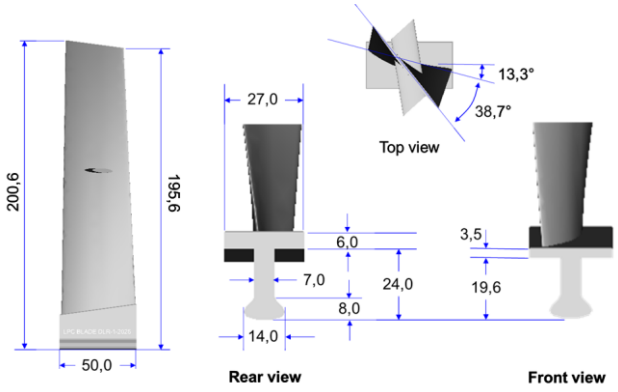


Figure 7: EventManager GUI

In the case of an inspection, the CfP includes mandatory metrology data as outlined in Chapter 2.2 and illustrated in Figure 5. To support this, we instantiated the service order with 17 distinct inspection features from the QCM template, categorized into 4 geometric, 7 linear, 4 attributive and 2 surface texture characteristics (Figure 8): Most features reflect conventional dimensional checks on the base, airfoil, and snubber sections, such as nominal length and height, and airfoil thickness at 30 %, 60 %, and 90 % span levels. Angular features include twist angles at both hub and tip cross-sections. In our application, the key geometric product specification (GPS) is the overall blade surface deviation, defined as the pointwise deviation from the nominal CAD geometry, measured across the entire blade surface. The surface deviation is evaluated against a tolerance zone of 0.1 mm with symmetric extensions of ±0.05 mm around the nominal geometry. This zone is parameterized in the QCM using the GPS_ToleranceZone, which includes the properties *Shape*, *WidthExtend* values, and *SpecificationModificator* (CZ). The CZ modifier defines a combined tolerance zone, meaning the entire blade surface is treated as a single unified feature that must lie within one continuous tolerance volume. This constrains not only local surface deviations but also global shape shifts such as twisting, bending, or positional offsets.

The metrology measurements are executed by a Quality Service Station (QSS), which comprises two core services: robotic scanning and deviation evaluation. The underlying instantiation of their I4.0CS follows the same structural approach as that of the inspected blade, but is extended with features required to represent the dynamic behavior of active assets – specifically, robot-assisted object scanning and computer-based deviation assessment – within the Type 3 AAS. As sketched in Figure 9, top right, the key lies in the implementation of the assets' capabilities and skills. These are defined using the ControlComponent submodel

IDTA 02015 [13] and IDTA 02016 [14] to link the asset-specific skill implementations (e.g., scanning sequence or evaluation algorithm) with the general shared CapabilityDescription submodel across all participants.



Feature	Value	Tolerance
BladeSurfaceDeviation		
├ WidthExtendValue	0.1 mm	**± 0.05 mm**
├ Shape	Surface	–
├ SpecificationModificator	CC	–
├ EngineeringUnit	mm	–
├ DatumField1	CAD nominal	–
├ DatumField2	CAD scan	–
BladeTwistAngle (Hub-Tip)	13.3° → 38.7°	**± 0.5°**
BladeLengthLE (z-axis)	200.6 mm	**± 0.2 mm**
BladeLengthTE (z-axis)	195.6 mm	**± 0.2 mm**
ProfileThickness @ 30 %	5.0 mm	**± 0.1 mm**
ProfileThickness @ 60 %	3.5 mm	**± 0.1 mm**
ProfileThickness @ 90 %	2.5 mm	**± 0.1 mm**
BaseLength (x-axis)	50.0 mm	**± 0.1 mm**
BaseLength (y-axis)	27.0 mm	**± 0.1 mm**
ProfileRadiusLE	0.5 mm	**± 0.05 mm**
SurfaceRoughness	6.0 mm	**± 0.05 mm**
…	…	…

Figure 8: Nominal LPC Blade CAD (top); Selected metrology features from the QualityControl SM (bottom)

To operationalize its declared capabilities, the QSS AAS instantiates an internal ServiceProvider state machine, as is the case for all proactive AAS in the system – including the inspected blade. However, while the blade's state machine primarily orchestrates virtualized or data-driven services (e.g., document delivery, metadata responses), the QSS binds its capabilities to responsive connected assets. As a result, the QSS not only interprets incoming service requests semantically but also physically executes corresponding workflows through internally coordinated skills. The state machine runs the full execution logic – capability and feasibility check, configuration, execution of commands, processing of responses and delivery of results – through condition-based transitions, typically triggered after a positive response to a Call for Proposal (Figure 9, top left). Execution involves direct interaction with robotic or computational subsystems, which might be orchestrated via the Asset Data Server (ADS).

Even though the QSS's connected assets already operate their own local control software and runtime environments (Figure 9, left), the I4.0CS still integrates an ADS to coordinate the execution context holistically (Figure 9, bottom right). In this configuration, the ADS does not serve as a basic abstraction layer – as is often the case for non-instrumented assets – but instead supervises messaging, job control, and internal synchronization across multiple responsive subsystems. For example, in the robotic scanning scenario, the ADS hosts the execution ecosystem that enables the collaboration between robot arm and scan effector, exposing service-level entry points through a meta-command interface. These endpoints may wrap ROS nodes, scripted procedures, or Python-based services, accessible via standard protocols such as REST or OPC UA. The ADS handles job queueing, assigns job ticket IDs, and enables asynchronous polling of status or results by external requesters. This supervisory role ensures decoupled, reliable service orchestration within the proactive AAS, while maintaining full observability and control from the semantic layer. The implementation of such execution workflows – including job invocation, result handling, and integration with the ServiceProvider state machine – is described in Chapter 3.2.

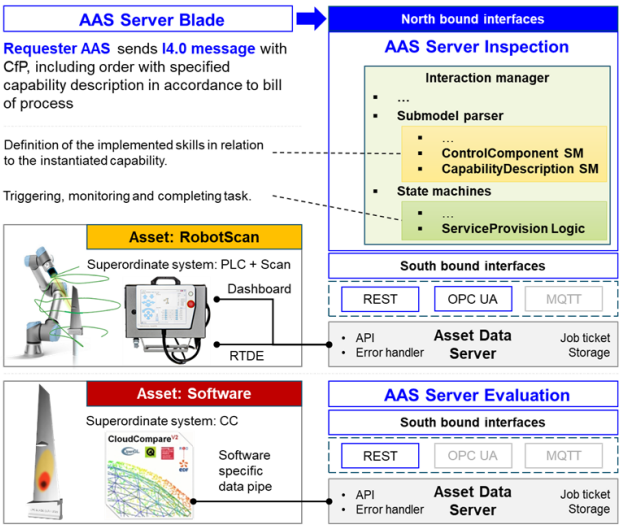


Figure 9: AAS abstracted Quality Service Station

3.2. Inspection Service Implementation

The assets of the Inspection Service comprise the *PathPlanner*, the *ScanService*, and the *EvaluationService* (cf. Figure 11). When a geometry inspection request is transmitted from the QSS to the Inspection Service, the corresponding CAD geometry forms the basis for the inspection process. To ensure simplified and automated data acquisition, the inspection is carried out in a section-based manner, where the geometry is scanned and analyzed in predefined regions (scan sections) communicated by the QSS (*face_id* of the CAD file). The Robot-Scan-Unit (RSU), which integrates both the scanning and robotic subsystems, executes the *ScanService* and provides the data acquisition capability. The RSU consists of an Ensenso B57-4-GN 3D camera [11] in combination with a UR10 robotic arm (Universal Robots) [24] (cf. Figure 10 a)).

The path planning process can be initiated once the CAD geometry and the corresponding face of interest (FOI) have been provided. For simplification, the path calculation is represented as the determination of a robot pose in which the camera can capture the desired section of the geometry. Accordingly, the *PathPlanner* first extracts the FOI from the CAD file (.STEP format) and computes a reference point above the FOI within the virtual coordinate system of the CAD model. Considering the fixed relative position of the blade to the robot, these virtual coordinates are then transformed into real-world 3D joint coordinates of the UR10, expressed as $[q_1, q_2, q_3, q_4, q_5, q_6]$, where q_i

corresponds to each joint angle.

After the robot assumes the computed pose, the scanning process is triggered and the 3D camera records the scene. The recorded point cloud is made available via the camera's *Python* API. Within the *ScanService* module, a preprocessing step removes background data and outliers using a Statistical Outlier Removal (SOR) algorithm in combination with predefined distance-based filters. The CAD geometry is then roughly aligned to the point cloud by means of predefined transformations (cf. Figure 10b)). A fine registration step is subsequently performed using the Iterative Closest Point (ICP) algorithm, which finely aligns the preprocessed point cloud with the CAD geometry. The resulting CAD transformation matrix is stored and passed to the evaluation step (cf. Figure 11).

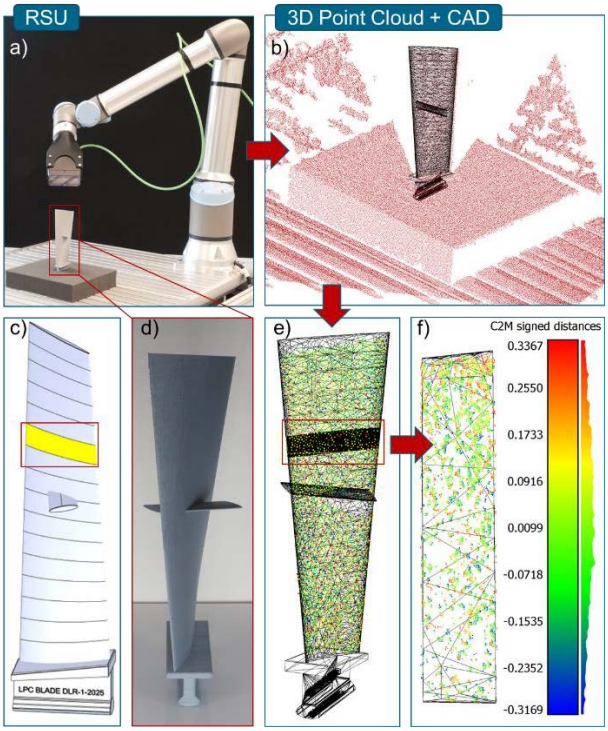


Figure 10: RSU Asset a), 3D Point Cloud with pre-aligned CAD (illustrated in CC) b), FOI definition on CAD-geometry c), 3D printed sample d), Processed 3D point cloud with aligned FOI (illustrated in CC) e) Exemplary Segment-Deviation-Analysis (illustrated in CC) f)

The *EvaluationService*, representing the third principal asset of the Inspection Service, performs the geometric deviation analysis between the FOI on the CAD model and the corresponding segment of the scanned point cloud. Its core functionality leverages CloudCompare (CC) [3] in headless mode for deviation computation, complemented by *Python*-based postprocessing routines for classification and result preparation. Applying the CAD transformations to the FOI produces the aligned 3D data (point cloud, CAD, and FOI) shown in Figure 10 e). The aligned CAD geometry and point cloud can then be cropped using the bounding box of the FOI, yielding CAD and point-cloud segments for a localized deviation analysis of the FOI region (cf. Figure 10 f)).

Finally, a simplified tolerance check, which neglects the statistical behavior of the recorded data at this stage, compares the maximum deviation values with a predefined tolerance threshold for each section, depending on its location on the geometry. If the tolerance threshold is exceeded, the asset variable *tolerance_out* is set, otherwise, the *tolerance_in* flag is set (cf. Figure 12). The evaluation result is then stored and transmitted back to the QSS (cf. Figure 11).

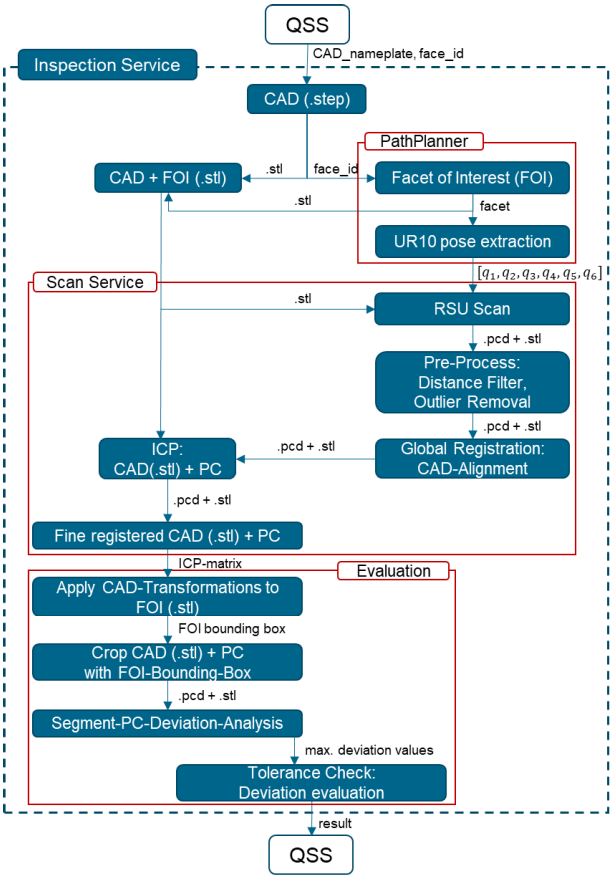


Figure 11: Implementation of the Inspection Service

To embed this functionality into the I4.0CS, a dedicated Asset Data Server (ADS) was developed. It encapsulates the CC logic, manages job lifecycles, and exposes a network-capable HTTP API – effectively transforming the standalone evaluation pipeline into a callable microservice. In addition to simple task invocation, the ADS handles multi-threaded job execution, queueing, result processing, and optional visualization. This API is linked to the Asset Administration Shell via its asset interface handler, while the executable endpoints are semantically mapped through the *ControlComponent* submodel [13] [14] to the corresponding skill implementation. This, in turn, connects them to the declared capability and its parameter definitions in the *CapabilityDescription* submodel.

The actual service execution is managed by the internal behavior logic of the proactive AAS, specifically the **Service Provider** state machine (Figure 12). Upon receiving and checking a CfP for "Inspection", and entering *State10_ServiceProvision*, the AAS initiates execution of the requested capability – in this case, geometric deviation evaluation. The state implementation retrieves all previously resolved service parameters from a *SRparameterValueMatrix*, which had been assembled and validated during earlier states (e.g., State03_CheckFeasibility). These parameters are then mapped to the evaluator's API interface. For instance, tolerances such as *WidthExtendTolerance1*, *WidthExtendValue*, and *WidthExtendTolerance2*, used in the common QualityControlMachining submodel, are translated into the corresponding tolerance_low, tolerance_middle, and tolerance_high fields expected by the evaluator. Similarly,

DatumField1 and *DatumField2* specify the URLs of the STL models to be compared. An HTTP POST request is issued to the evaluator's /start-job endpoint using these bound parameters. Internally, the evaluator ADS handles the job in a threaded environment, launching the CloudCompare-based deviation analysis and managing its status asynchronously. The AAS ServiceProvider, meanwhile, enters a polling loop and regularly checks the evaluator's /job/{job_id}/status endpoint until a terminal state (done or failed) is reached (not illustrated in Figure 12).

```
class State01_WaitForCallForProposal(AState): …
…
class State10_ServiceProvision(AState):
  def initialize(self): …
  def skill_execute(self, SRparameterValueMatrix):
        Asset variables          Submodel properties
      form_data = {
         "tolerance_in":     SR…[0]["WidthExtendTolerance1"],
         "tolerance_out":    SR…[0]["WidthExtendValue"],
         "reference_url":    SR…[0]["DatumField1"],
         "compare_url":      SR…[0]["DatumField2"],
          … }
      Asset command
      res = requests.post(f"{url}/start-job", data=form_data)
  def actions(self) -> None: …
  def transitions(self) -> object: State11_Send…
class State11_SendCompletion(AState): …
  def initialize(self): …
  def actions(self) -> None: …
  def transitions(self) -> object: State01_Wait…
…
```

Figure 12: Simplified service provider state machine

Upon completion, the AAS retrieves the final result using /job/{job_id}/result. This response includes structured classification metrics as well as metadata for downstream documentation:

- A conformity decision (yes/no),
- Percentage of deviation points exceeding configured thresholds (e.g, *WidthExtendValue*),
- Path to preview images, GLB model, and raw deviation CSV if required.

The AAS parses the result, optionally evaluates conformity rules, and then transitions to State11_SendCompletion. The outcome is then returned to the service requester, who formally appends it to the QualityControlMachining submodel results section of the inspected part. Since the QualityControlMachining is also a referenced part of the Digital Product Passport in our architecture, the data are conserved in a persistent and traceable structure.

3.3. Processing results and Discussion

The processing workflow demonstrates that quality inspection in a federated aviation dataspace can be executed in an automated manner. After the CAD reference and the scanned mesh are retrieved, the deviation analysis is performed using CloudCompare in headless mode, with Iterative Closest Point (ICP) alignment applied when required. The results are subsequently post-processed in Python to derive structured deviation metrics, which are then written back into the QualityControlMachining submodel. While the implementation already demonstrates important strengths, it also exposes technical limitations that need to be addressed. The key challenges, along with the first implementations achieved, are summarized below:

Preliminarily implemented:

- Automated workflow from CAD reference and mesh acquisition to deviation analysis.
- ICP-based alignment and Python post-processing provide structured deviation metrics.
- Results are semantically integrated in a standardized form, here into the QualityControlMachining submodel, ensuring interoperability and traceability.
- Demonstrated feasibility of federated quality inspection using AAS, DPP, and EDC.

Issues to be clarified in the next steps:

- CAD-to-scan registration requires predefined transformations; no fully autonomous matching yet.
- Camera system resolution limited to 0.1–0.3 mm, constraining accuracy for critical blade features.
- Maximum-deviation evaluation insufficient; lacks statistical characterization of 3D data and damage variants.
- Scaling to industrial batch inspection limited by sensor performance and data throughput.

To address these issues, several improvements are required. Autonomous registration strategies based on feature recognition or machine learning could replace predefined transformations, enabling more robust CAD-to-scan alignment. Higher-resolution sensors, or hybrid systems combining optical and tactile measurement, would improve data fidelity for critical airfoil features. Statistical evaluation methods, such as distribution-based deviation metrics or topology-aware damage indicators, should complement maximum-deviation checks to capture subtle defect patterns. Finally, optimization of data throughput and scheduling within the Quality Service Station is needed to enable scalable batch inspection.

While a simplified evaluation of maximum values within the deviation analysis can provide reference values for quality assessment, it neglects currently the statistical behavior of the 3D data and the consideration of damage variants. It is therefore conceivable that, in the case of damage, the deviation values may remain within the admissible tolerance limits, while alternative statistical topologies of the recorded 3D data emerge within the same tolerance band. If unresolved, these limitations directly affect the reliability of conformity decisions, the sensitivity to early-stage damage, and the feasibility of scaling the service to industrial environments. Conversely, addressing them will not only improve measurement accuracy but also strengthen trust in federated inspection workflows where results must be shared and reused across multiple stakeholders.

Nevertheless, the experimental implementation confirms the feasibility of autonomous quality inspection using AAS, DPP, and EDC as enabling technologies. The achieved level of integration illustrates how inspection results can be semantically captured, linked to product passports, and distributed across stakeholders in a controlled dataspace. At the same time, the findings underline the importance of

advanced statistical evaluation methods and robust data acquisition strategies to ensure industrial applicability.

4. CONCLUSION AND OUTLOOK

This paper demonstrated the feasibility of autonomous quality inspection in aviation MRO and manufacturing using proactive Asset Administration Shells (AAS) within a federated dataspace. An Industry 4.0 Component Stack integrating the AAS, including a Digital Product Passport (DPP), an Asset Data Server (ADS) and the Eclipse Dataspace Connector (EDC) was instantiated for a low-pressure compressor blade, covering robotic scanning, path planning, and deviation analysis. The implementation confirmed end-to-end orchestration and semantic integration of inspection results, enabling traceable and machine-interpretable outcomes across organizational boundaries.

In relation to the central research question, the findings indicate that an autonomous and federated quality control service can indeed be realized. The overall framework has proven valid and scalable; the identified constraints concern the current implementation. Limitations in sensor resolution (0.1–0.3 mm), predefined CAD-to-scan registration, and maximum-deviation based evaluation affect inspection accuracy and robustness but do not contradict the general approach. Similarly, challenges in dataspace integration – such as contract negotiation, credential exchange, and audit-ready provenance – reflect practical maturity issues rather than conceptual weaknesses.

Future directions include, at the framework level, advancing policy enforcement, harmonizing semantics, and ensuring sovereign data exchange. At the implementation level, progress requires higher-resolution sensing, more robust registration methods, and statistical evaluation beyond maximum values. A simplified physical setup is foreseen, where the blade surface is divided into CAD-based sections; each section number is passed as reference and directly mapped to UR10 robot coordinates via the PathPlanner, enabling targeted inspection of critical areas instead of full circumferential scans.

Nevertheless, the presented architecture already delivers an MX-Port aligned, standards-based realization of the Type-3 AAS. It shows how digitally described skills can be semantically configured at runtime to control complex services through interoperable interfaces, marking a significant step toward autonomous and federated inspection in aviation MRO and manufacturing.

5. ACKNOWLEDGEMENT

The authors gratefully acknowledge the support from The German Federal Ministry for Economic Affairs and Climate Action (BMWK) through the Aerospace-X project (Grant No.13MX004B) as part of the Manufacturing-X initiatives. In addition, we would like to thank our partners in the project for the fruitful collaboration.

6. ABBREVIATIONS

AAS	Asset Administration Shell
ADS	Asset Data Server
AID	Asset Interface Description
API	Application Programming Interface
BP	Battery Pass
CAD	Computer Aided Design
CC	Cloud Compare
CfP	Call for Proposal
CPS	Cyber-Physical System
CPSS	Cyber-Physical-Social System
CX	Catena-X
DPP	Digital Product Passport
EDC	Eclipse Dataspace Connector
I4.0C	Industry 4.0 Component
I4.0CS	Industry 4.0 Component Stack
ICP	Iterative Closest Point
IDS	International Data Space
IDSA	International Data Spaces Association
IDTA	Industrial Digital Twin Association e.V.
IIoT	Industrial Internet of Things
IoT	Internet of Things
IT	Information Technology
LPC	Low-Pressure Compressor
MaSiMO	Maintenance Simulation Model @ DLR MO
MRO	Maintenance, Repair and Overhaul
OPC UA	Open Platform Communication Unified Architecture
PI4.0	Plattform Industrie 4.0
QCM	Quality Control for Machining
QSS	Quality Service Station
Ref	Reference
RSU	Robot-Scan-Unit
SMC	SubmodelElementCollection
SML	SubmodelElementList
SM	Submodel
SP	Service Provider
SR	Service Requester
WebUI	Web-based User Interface

7. REFERENCES

[1] Belyaev, A.; Diedrich, C.; Köther, H.; Dogan, A. Dezentraler IOTA-basierter Industrie-Marktplatz: Industrie-Marktplatz auf Basis von IOTA, eCl@ss und I4.0-Verwaltungsschale, **2020**, Industry 4.0 Management, 36, ISSN: 2364-9208

[2] Bilen, A.; Stamer, F.; Behrendt, S.; Lanza, G. A Quality Data Model Based on Asset Administration Shell Technology to Enable Autonomous Quality Control Loops, Paper, Production at the Leading Edge of Technology, **2024**, Springer, Cham. DOI: 10.1007/978-3-031-47394-4_20

[3] CloudCompare (version 2.13) [Computer software], **2025**, https://www.cloudcompare.org/ (accessed on 16.09.2025)

[4] Diedrich, C.; Belyaev, A.; Hermann, J. et al. Information Model for Capabilities, Skills & Services. Plattform Industrie 4.0: Berlin, Germany, **2022**. Available online: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/CapabilitiesSkillsServices.pdf (accessed on 15.09.2025)

[5] Diedrich, C.; Schröder, T.; Belyaev, A. Interoperability of Cyber Physical Systems, **2022**, Kommunikation und Bildverarbeitung in der Automation, Technologien für die intelligente Automation 14, Springer Vieweg, Germany DOI: 10.1007/978-3-662-64283-2_8

[6] Digital Twin and Asset Administration Shell Concepts and Application in the Industrial Internet and Industry 4.0. An Industrial Internet Consortium and Plattform Industry 4.0 Joint Whitepaper, **2020** https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Digital-Twin-and-Asset-Administration-Shell-Concepts.html (accessed on 15.09.2025)

[7] DIN EN IEC 63278-1 VDE 0810-781:2022-07 Asset Administration Shell for industrial applications, **2022**, VDE Standards, VDE Verlag, Berlin

[8] Factory-X. MX-Port Concept: Enable data sharing across industries, Executive Summary: Introducing MX-Port Concept for Manufacturing-X Data Spaces. Version 1.00, **2025**. Factory-X, https://factory-x.org/wp-content/uploads/MX-Port-Concept-V1.00.pdf (accessed on 15.09.2025)

[9] Gabellini, M.; Civolani, L.; Ronchi, M.; Naldi, L.D.; Regattieri, A. Data Spaces in Manufacturing and Supply Chains: A Review and Insights from European Initiatives, Article, **2025**, Appl. Sci., 15, 5802. DOI: 10.3390/app15115802

[10] Grunau, S.; Redeker, M.; Göllner, D.; Wisniewski, L. The Implementation of Proactive Asset Administration Shells: Evaluation of Possibilities and Realization in an Order Driven Production. **2022**, In: Jasperneite, J., Lohweg, V. (eds) Kommunikation und Bildverarbeitung in der Automation. Technologien für die intelligente Automation, vol 14. Springer Vieweg, Berlin, Heidelberg
DOI: 10.1007/978-3-662-64283-2_10

[11] IDS Imaging Development Systems GmbH: *Ensenso B-Series 3D-vision at close range, **2025**, Obersulm, Germany.* https://en.ids-imaging.com/ensenso-3d-camera-b-series.html (accessed on 07.09.2025)

[12] IDTA 02010-1-0 Service Request Notification, Submodel Template of the AAS, **2023**, publisher: Industrial Digital Twin Association, Frankfurt am Main, Germany, https://industrialdigitaltwin.org (accessed on 04.09.2025)

[13] IDTA 02015-1-0, Control Component Type, Submodel Template of the Asset Administration Shell, **2023**, publisher: Industrial Digital Twin Association, Frankfurt am Main, Germany, available online: https://industrialdigitaltwin.org (accessed on 02.09.2025)

[14] IDTA 02016-1-0, Control Component Instance, Submodel Template of the Asset Administration Shell, **2023**, publisher: Industrial Digital Twin Association, Frankfurt am Main, Germany, available online: https://industrialdigitaltwin.org (accessed on 02.09.2025)

[15] IDTA 02017-1-0 Asset Interface Description, **2024**, publisher: Industrial Digital Twin Association, Frankfurt am Main, Germany, https://industrialdigitaltwin.org
(accessed on 04.09.2025)

[16] IDTA 02027-1-0 Asset Interfaces Mapping Configuration, **2024**, publisher: Industrial Digital Twin Association, Frankfurt am Main, Germany, https://industrialdigitaltwin.org
(accessed on 04.09.2025)

[17] Jacoby, M.; Jovicic, B.; Stojanovic, L.; Stojanovi´c, N. An Approach for Realizing Hybrid Digital Twins Using Asset Administration Shells and Apache StreamPipes, Article, Information, **2021**, 12, 217. DOI: 10.3390/info12060217

[18] Jacoby, M.; Volz, F.; Weißenbacher, C. Müller, J. FA³ST Service – An Open Source Implementation of the Reactive Asset Administration Shell, **2022** DOI: 10.1109/ETFA52439.2022.9921584.

[19] Jacoby, M.; Volz, F.; Weißenbacher, C.; Stojanovic, L.; Usländer, T. An approach for Industrie 4.0-compliant and data-sovereign Digital Twins: Realization of the Industrie 4.0 Asset Administration Shell with a data-sovereignty extension, Article, Automatisierungstechnik, vol. 69, no. 12, **2021**, pp. 1051-1061,
DOI: 10.1515/auto-2021-0074

[20] Lario, J.; Terol, M.; Mendizabal, B.; Tomas, N. Inspection as a Service Business Model for Deploying Non-Destructive Inspection Solutions Within a Blockchain Framework, **2025**, J. Theor. Appl. Electron. Commer. Res., 20, 52.
DOI: 10.3390/jtaer20010052

[21] Liedl, P.; Biahmou, A.; Friedrich, M. et al. InterOpera Submodel Specification: Quality Control for Machining Submodel, Version 0.5, **2023**; Steinbeis Innovation gGmbH: Stuttgart, Germany. Available online: https://landkarte.interopera.de
(accessed on 15.09.2025).

[22] Shi, D.; Liedl, P.; Bauernhansl, T. Interoperable information modelling leveraging asset administration shell and large language model for quality control toward zero defect manufacturing, Article, **2024**, J. of Manufacturing Systems, 77, DOI: 10.1016/j.jmsy.2024.10.011

[23] The Eclipse Foundation. Eclipse BaSyx: Industry 4.0 Operating System. https://eclipse.dev/basyx
(accessed on 15.09.2025)

[24] Universal Robots: *UR10e*, **2025**, München, Germany. https://www.universal-robots.com/de/produkte/ur10e/
(accessed on 07.09.2025)

[25] Urban, C.; Belyaev, A.; Diedrich, C. Verwaltungsschale-basierter Ansatz für die Umsetzung von auftragsgesteuerter Produktion, **2022**, Conference paper, 23. VDI-Kongress AUTOMATION – Leitkongress der Mess- und Automatisierungstechnik, Baden-Baden, Germany

[26] VDI/VDE 2193 Part 1:2020-04 Language for I4.0 Components - Structure of messages, VDI/VDE guideline **2020**, VDI-VDE Manual, Düsseldorf

[27] VDI/VDE 2193 Part 2:2020-01 Language for I4.0 components - Interaction protocol for bidding procedures, VDI/VDE guideline **2020**, VDI-VDE Manual, Düsseldorf

[28] Vieira da Silva, L. M.; Köcher, A.; Gill, M.; Weiss, M.; Fay, A. Toward a Mapping of Capability and Skill Models using Asset Administration Shells and Ontologies, Paper, **2023**, IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)
DOI: 10.1109/ETFA54631.2023.10275459

[29] Weiss, M.; Raddatz, F.; Wende, G. MaSiMO – Digital Product Passport and autonomous event management of Industry 4.0 Components with proactive AAS in data-driven aviation maintenance and production, **2024**, Paper, Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V.
DOI: 10.25967/630084

[30] Weiss, M.; Wicke, K.; et al. MaSiMO - Development and Research of Industry 4.0 Components with a Focus on Experimental Applications of Proactive Asset Administration Shells in Data-Driven Maintenance Environments, **2023**, Paper, Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V.
DOI: 10.25967/610125

[31] Weiss, M.; Wicke, K.; Wende, G. MaSiMO - A Hybrid Experimental Platform for the Simulation and Evaluation of Data-Driven Maintenance Enterprises, **2022**, Paper, Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V.
DOI: 10.25967/570154

[32] Winkler, D..; Gill, M. S.; Fay, A. The Asset Administration Shell as a solution concept for the realisation of interoperable Digital Twins of Aircraft Components in Maintenance, Repair and Overhaul, **2022**, Paper, Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V.
DOI: 10.25967/570274

[33] Zhang, J., Ellwein, C., Heithoff, M. et al. Digital twin and the asset administration shell, Article, **2025**, Softw Syst Model 24, 771–793,
DOI: 10.1007/s10270-024-01255-0