ZEROSWARM

ZERO-enabling Smart networked control framework for Agile cyber physical production systems of systems

D4.1 – Digital twin configuration, edge AI and deep learning v1

Project funded by Horizon Europe, Grant Agreement #101057083

Topic	HORIZON-CL4-2021-TWIN-TRANSITION-01-08
Project Title	ZERO-enabling Smart networked control framework for Agile cyber physical production systems of systems
Project Number	101057083
Project Acronym	Zero-SWARM
Deliverable No/Title	D4.1 – Digital twin configuration, edge AI and deep learning v1
Contractual Delivery Date	M13
Actual Delivery Date	M13
Contributing WP	WP4 – Runtime information continuum enabling awareness in the industrial domain
Project Start Date, Duration	01/06/2022, 30 Months
Dissemination Level	Public (PU)
Nature of Deliverable	Document, report (R)

Document History

Date	*Version*	*Author*	*Description*
30-03-2023	0.1	*Roberto Fernandez (AIMEN)*	Deliverable structure preparation
26-06-2023	0.2	*Roi Méndez-Rial (AIMEN)*	First complete draft
28-06-2023	0.3	*Roi Méndez-Rial (AIMEN)*	Address internal review comments
30-06-2023	1.0	*Anastasios Drosou (CERTH)*	Final submission

Authors List

Leading Author (Editor)			
Surname	***Initials***	***Beneficiary Name***	***Contact email***
Méndez-Rial	RM	AIMEN	*roi.mendez@aimen.es*
Co-authors (in alphabetic order)			
#	***Surname***	***Initials***	***Beneficiary Name***	***Contact email***
1	*Alonso*	LA	AIMEN	*lucia.alonso@aimen.es*

Contributors (in alphabetic order)				
#	*Surname*	*Initials*	*Beneficiary Name*	*Contact email*
1	*Abadía*	*AA*	*AIM*	antonio.abadia@aimen.es
2	*Barja*	*LB*	*AIM*	lara.barja@aimen.es
3	Gil	*IG*	*AIM*	lago.gil@aimen.es

Reviewers List

List of Reviewers (in alphabetic order)				
#	*Surname*	*Initials*	*Beneficiary Name*	*Contact email*
1	*Khodashenas*	*PSK*	*HWE*	*pouria.khodashenas@huawei.com*
2	*Mpatziakas*	*AM*	*CERTH*	*ampatziakas@iti.gr*
3	*Drosou*	*AD*	*CERTH*	*drosou@iti.gr*

DISCLAIMER OF WARRANTIES

This document has been prepared by Zero-SWARM project partners as an account of work carried out within the framework of the contract no 101057083.

Neither Project Coordinator, nor any signatory party of Zero-SWARM Project Consortium Agreement, nor any person acting on behalf of any of them:

- makes any warranty or representation whatsoever, express or implied,
    - with respect to the use of any information, apparatus, method, process, or similar item disclosed in this document, including merchantability and fitness for a particular purpose, or
    - that such use does not infringe on or interfere with privately owned rights, including any party's intellectual property, or
- that this document is suitable to any particular user's circumstance; or
- assumes responsibility for any damages or other liability whatsoever (including any consequential damages, even if Project Coordinator or any representative of a signatory party of the Zero-SWARM Project Consortium Agreement, has been advised of the possibility of such damages) resulting from your selection or use of this document or any information, apparatus, method, process, or similar item disclosed in this document.

Zero-SWARM has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101057083. The content of this deliverable does not reflect the official opinion of the European Union. Responsibility for the information and views expressed in the deliverable lies entirely with the author(s).

Table of Contents

List of Figures

List of Abbreviation

Abbreviation	Description
AM	Additive Manufacturing
AI	Artificial Intelligence
AAS	Asset Administration Shell
AML	Automation Markup Language
DT	Digital Twin
I4.0	Industry 4.0
IoT	Internet of things
OPC-UA	Open Platform Communications – Unified Architecture
RAMI4.0	Reference Architecture Model for Industry 4.0

Executive Summary

This deliverable reports the first outcomes of the Task 4.1 “Digital twin configuration, edge AI and deep learning”. The digital twin concept is essential for enabling interoperability in Industry 4.0, creating a digital representation of physical systems that communicate with their environments to collect, analyze, and simulate data. The Asset Administration Shell (AAS) is the implementation of the digital twin for Industry 4.0. The aim of this task is to provide a comprehensive methodology and open toolkit for implementing Asset Administration Shell (AAS) with embedded AI capabilities on the edge and a set of open-source tools for supporting their implementations. The deliverable provides a review of current challenges and trends associated to the development of AAS and a summary of current open-source software solutions supporting these implementations. Based on a technology gap analysis, the design principles of the Zeros-Swarm AAS toolkit, which includes an AAS modelling tool and an edge computing framework, is described together with an example application in one of the project trials.

1 Introduction

The adoption of Industry 4.0 requires interoperability so that components, devices and applications can communicate seamlessly across companies, industries and countries. In this context the concept of Digital Twin is becoming essential, which is related to the creation of a digital representation of the physical systems that communicate with their environments in various ways to collect, analyze, and simulate data in the digital world and improve the performance of the physical systems.

A digital twin can be defined as a formal digital representation of an entity, with attributes and optionally computational, geometrical, visualization and other models, offering a service interface for interacting with it, adequate for communication, storage, interpretation, process and analysis of data pertaining to the entity in order to monitor and predict its states and behaviors within a certain context [[1]]. The content of the digital representation is determined by the set of use cases for which digital twin is designed. The level of abstraction and complexity of a digital twin must be such that it is sufficient to meet the requirements of the use cases for which the digital twin is designed.

The industrial systems and the industrial usage scenarios are complex and diverse, therefore, their requirements on digital twin are naturally complex and diverse as well, leading to different flavors of digital twin implementations [[1]]. Some implementations of digital twin may contain many attributes and data, computational capabilities and perhaps even a formal interface for communication to satisfy the application requirements, some others may only need a small set of attributes and data to be sufficient to support their application. Generally, a digital twin consists of three elementary aspects, namely, data, models and service interfaces.

The complexity behind the digital twin is translated in to the amalgama of technologies that are required to implement the concept, addressing challenging technical aspects as: information modelling, interoperability, data management and aggregation, real-time synchronization, scalability or security. The use of open standards and open-source software is crucial for the digitalization of manufacturing, including the implementation of Digital Twins [[2]].

The Asset Administration Shell (AAS) is described as the implementation of a digital twin for Industry 4.0. The AAS is the digital representation of a physical asset, enabling it to interoperate with other assets and systems. The AAS together with the asset constitute an Industry 4.0 component. AAS provides the basis for the development and use of unified and open Industry 4.0 standards enabling interoperability as one of the strategic fields. A series of specifications have been published to help developers to implement AAS since the introduction of the term in the reference architecture model for Industry 4.0 (RAMI4.0). To name few, [[3]], [[4]], [[5]], [[6]].

The AAS specification is still a work in progress, and therefore the implementation is still challenging. There are already several open-source frameworks available supporting developers to work with Asset Administration Shells. However, existing tools do not implement the complete specification and only facilitate the modelling or implementation of certain types of AAS. Most solutions are focused on cloud-based digital twins with limited embedded processing capabilities and do not support properly the synchronization of the AAS with the asset.

Zero-SWARM aim to provide a comprehensive methodology and an open Toolkit to support the development and implementation of AAS of manufacturing assets in the operation phase with integrated AI and near real-time processing capabilities. The main application of the framework will be on AAS that can be deployed in field devices or embedded devices which operate close to the machines where the data is generated. The goal is to transform assets into standalone intelligent digital twins that can operate in a decentralized yet coordinated manner. Many industry solutions will require

digital twins with near real-time data and event processing capabilities, for instance exploiting edge devices endowed with 5G connectivity for distributed control or autonomous systems.

1.1 Purpose of the document

This deliverable shows the first results of task T4.1 “Digital twin configuration, edge AI and deep learning” which aims to deliver a comprehensive methodology and an open toolkit to support the development of edge digital twins based on AAS with embedded AI capabilities. The toolkit will include a novel open-source AAS modeling tool and a framework/SDK for implementing digital twins and make them deployable on heterogeneous Linux-based embedded hardware platforms. A second version of this report will be delivered at the end of the task with the final outcomes. Section 2 introduces the challenges associated to the development of digital twins in I4.0 based on AAS. Section 3 provides an overview of current AAS open tools and open-source software solutions that can be used for their implementation. Section 4 describes the main innovation to be delivered by Zero-SWARM, i.e., the AAS Open Toolkit, explaining the overall methodology for AAS implementation and the design principles of the software tools. Section 5 shows an example of a practical implementation of an AAS for one of the trials using the toolkit.

1.2 Connection with other work packages

This task is related to the following work packages:

- WP2: The overall zero-swarm architecture and the trials requirements are defined in WP2. The implementation of the digital twins will support mainly the Data Aggregation and Processing functional blocks and partially the Automation Application Layer in the reference architecture. WP2 provides information about the trials and the assets involved (machines, robots, etc..) for which AAS will be developed.
- WP3: WP3 aims to model 5G technology with AAS. WP3 and WP4 are collaborating and sharing knowledge about modelling with AAS. WP3 will provide 5G edge infrastructure upon which the interoperability test of the AAS will be performed.
- WP4: The digital twins will leverage and provide compatible interfaces with the data streaming technologies from edge to cloud (T4.2), and MLOps cloud services for ML/DL model development and deployment back on the edge (T4.4.).
- WP5: On-going discussion regarding the feasibility and interest of modelling IEC 61499 applications with AAS or OPC-UA information models. Synergies for integration approaches of AI models in the edge.
- WP6: T4.1 will support the implementation of AAS for the different trials in WP6.

2 Digital Twins in I4.0

Interoperability refers to the ability of different devices, systems, and technologies to communicate and work together seamlessly, enabling the free flow of data and information across the entire value chain. Its effective implementation will be crucial to unlocking the full potential of I4.0. The goal is to have industrial assets (components, systems) providing universal and cross-manufacturing communication standards that can be easily integrated into smart factories through plug & play, similar to the way USB devices are easily connected to computers. This requires a big effort in the development of integration software for assets and systems and complex digital models, defining communication structures and a common language.

The Digital Twin (DT) concept is one of the cornerstone technologies to solve the interoperability problem by combining isolated data in a semantically consistent manner ensuring that users have an

integrated, unified view of the data and information. Different types of DT, standards and specifications are being proposed [[7]]. DT should not be proprietary solutions, as it becomes hard to maintains its sustainable in the long term, instead it should be based on open standards.

In this context, Industry 4.0 platform through RAMI4.0 intend to make sure that all participants involved in I4.0 discussions and activities have a common framework to understand each other. The RAMI 4.0 framework is intended to enable open standards and address the interoperability issue in a structured manner. This model is complemented by the industry 4.0 components. The AAS specification is the concrete adaptation of the generic Digital Twin concept to industrial applications, and the only available industrial DT open standard.

2.1 RAMI4.0

The Reference Architecture Model for Industry 4.0 (RAMI 4.0) is a service-oriented architecture independent of vendors, products, and technologies, which consists of a three-dimensional map. These tree axes describe all crucial aspects of Industry 4.0 using a common perspective. In this way, complex interrelations are broken down into smaller and simpler clusters.

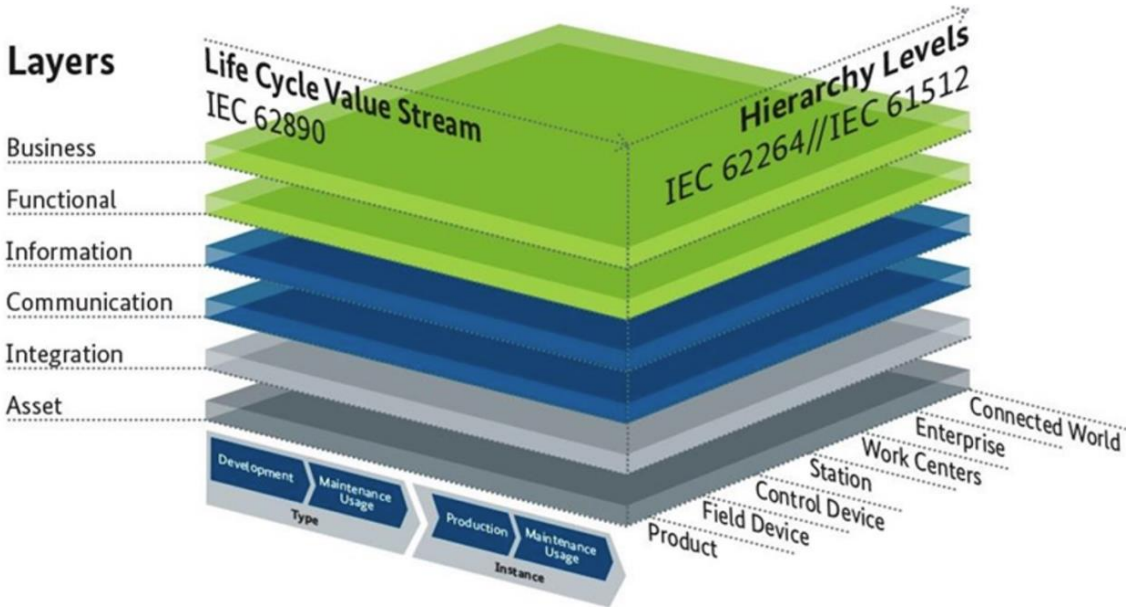


Figure 1: Reference Architecture Model for Industry 4.0 (RAMI 4.0)

The three axis that composes the RAMI 4.0 are:

- The horizontal "Hierarchy Levels" axis of IEC 62264, which is the international standards series for enterprise IT and control systems. This axis represents the different functionalities within factories or facilities.
- The horizontal "Life Cycle Value Stream" axis, which represents the life cycle, from the development to disposal, of facilities and products. It is based on IEC 62890, Life-cycle management for systems and products, used in industrial-process measurement, control, and automation.
- The vertical "Layers" axis that describes the decomposition of a machine into its properties, structured layer by layer (the virtual mapping of a machine). This type of representation has its origin in information and communication technology where the properties of complex systems are decomposed into layers.

Within these three axes, all crucial aspects of Industry 4.0 can be mapped, even the more highly flexible concepts, allowing objects such as machines to be classified according to the model.

2.2 Asset Administration Shell

An Industry 4.0 component is composed by an asset, which is defined as a physical, digital or intangible entity that has value to an individual or an organization, and the information which describes it through digital models, i.e., its digital twin. The AAS is the representation of the digital twin in the I4.0, enabling the use of assets in its digital form. This model has been selected to be used in I4.0 due its high degree of maturity to map the asset information across its entire lifecycle.

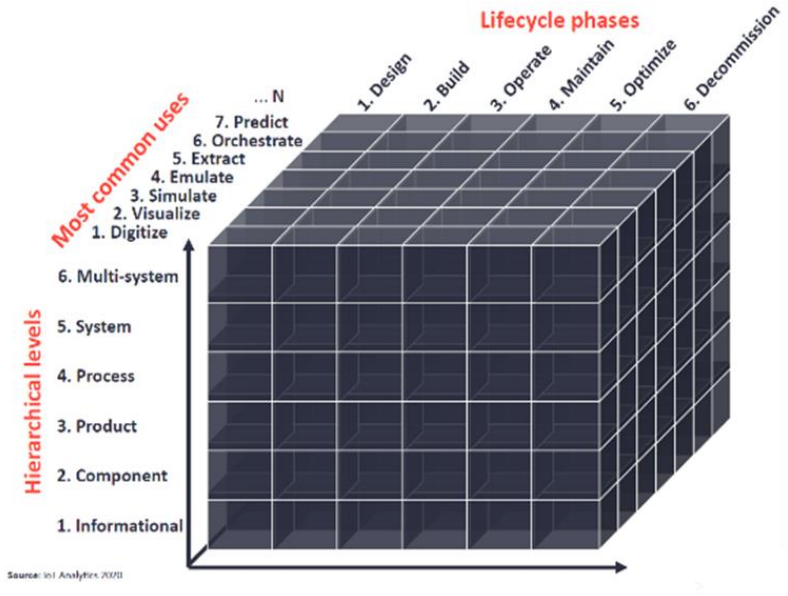


Figure 2: Classification of a Digital Twin

An asset can have several AAS or digital twins, as it is represented in the Figure 2: Classification of a Digital Twin, in function of the hierarchical level where the Digital Twin is applied, the common use of it and the lifecycle phase in which it is used. Therefore, same asset can be represented by completely different Digital Twins in function of what they are representing.

Using RAMI 4.0, a digital twin can represent it in the system through the hierarchical level and its LifeCycle phase, with the digital use as the most common. Finally, the architecture layers represent the properties of the I4.0 component, composed by the digital twin and asset.

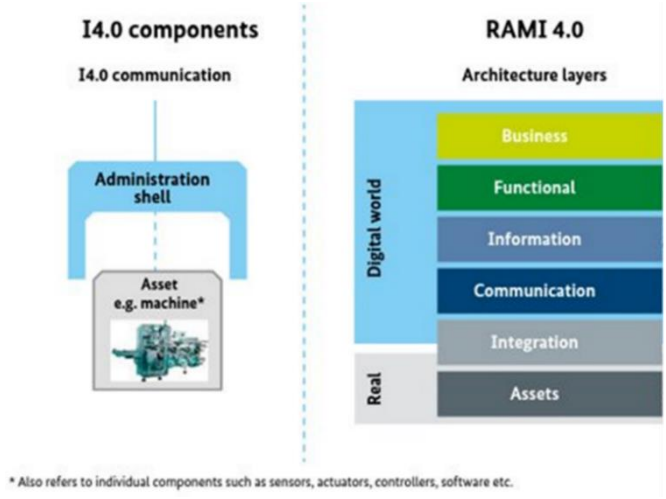


Figure 3: Layers axis of RAMI 4.0

The composition of the I4.0 component is shown in Figure 3, where the architecture layers can be differentiated in what are representing: the asset in the real world or the digital twin (Asset Administration Shell) in the digital world.

- Asset layer: it represents the physical world, the asset itself.
- Integration layer: it represents the connection between the physical and digital world, giving the capability to monitor and control the asset from a digital entity.
- Communication layer: it provides access to information through different protocols (as OPC UA, MQTT, etc.), giving standard communication to the information layer and services to the integration layer. Underlying networking technology which is responsible to provide the connectivity for information exchange between the mentioned protocols (OPC UA, MQTT, etc.) in the Zero-SWARM scope is the cellular communication system, i.e. 5G and beyond. Please note that there are alternative solutions available too such as legacy wired solutions (property – e.g. PROFINET or standard – e.g. Ethernet LAN) or even wireless solutions such as WiFi.
- Information layer: it describes the necessary data of the asset represented in information models (as AAS, AutomationML, OPC UA datamodel, etc.), i.e., the services and data that are offered, used, generated, or modified by the logical functions of the asset.
- Functional layer: it describes the logical functions of the asset in function of the role of it in the system.
- Business layer: it orchestrates functions to business processes and links them under the legal and regulatory constraints.

The Asset Administration Shell is standardized in IEC 63278-1 as a digital representation of an asset, where the asset is defined as a physical, digital or intangible entity with value to an individual or an organization. Therefore, an asset is identified as an entity in a specific state of its life, providing all the technical functionality and communication ability contained by it. The metamodel is the generic technology-neutral manufacturer-independent standardized interface used to implement the AAS. Using the metamodel, the information of the asset can be managed and structured in submodels, which are the aspects' representation of the asset.

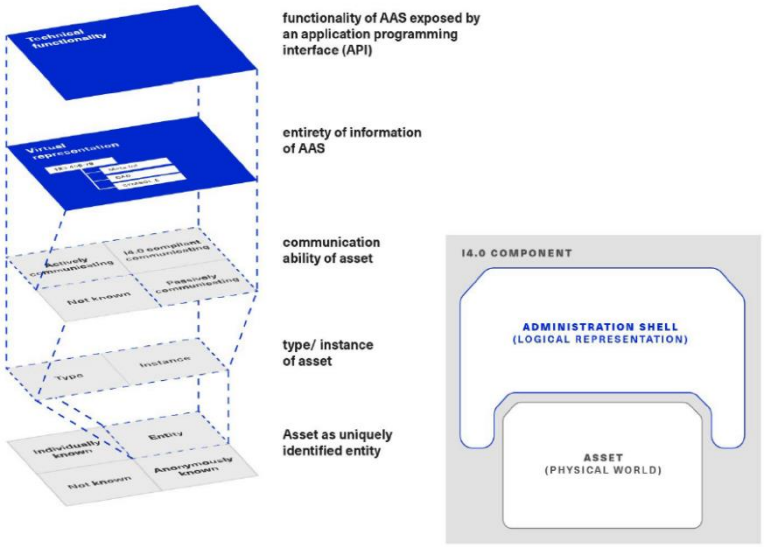


Figure 4: I4.0 component represented by an Asset Administration Shell

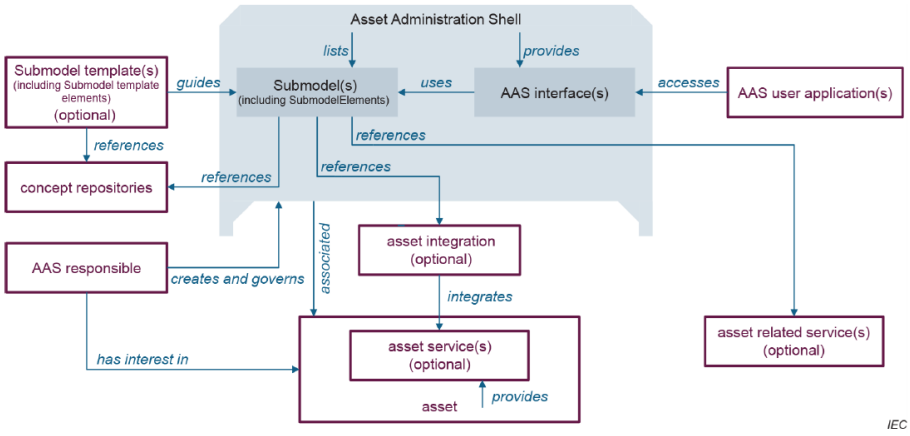


Figure 5: Asset Administration Shell structure defined in IEC 63278-1

The Asset Administration Shell is associated with a unique asset and organizes its information in a list of submodels, enabling to manage complex information. Each submodel is the representation of an aspect of an asset and it is used to structure the digital representation and technical functionality in a set of submodel elements, which can be properties, references, relationships, operations, etc.

The submodels can be standardized as templates, which are essential for interoperability. Industrial Digital Twin Association (IDTA) works in submodel standardization and provides a list of potential common submodels like Identification, TechnicalData, ConfigurationData or OperationalData. However, most submodels in the AAS that describe specifics functionalities of the asset cannot be standardized and their implementation will depend on the granularity of the model, abstraction level and use case.

Even the AAS is implemented using the metamodel, which is technology-neutral, the AAS can be instantiated using different formats to enable the exchange of information between industrial applications. This instantiation can be made using common industrial protocols like JSON, OPC UA data model or AML.

Asset Administration Shells can be classified in three types based on the way they exchange information:

- AAS Type 1: passive AAS as a file providing only static data . Type 1 are serialized in files such as XML, JSON, following the AAS metamodel. For instance, product catalog of directly provided by vendors.
- AAS Type 2: active AAS with a software abstraction layer implementing an API to access the information. Runtime instances containing both static and motion data. Type 2 provides properties and operations and is able to signal changing conditions with events. The data model is defined by the AAS meta model, with a generic runtime interface that enables accessing properties, operations, and events. The implementation of the AAS API can be done in HTTP, OPC-UA or MQTT according to the AAS specification. An AAS server application with HTTP/REST or OPC UA API is called reactive AAS.
- AAS Type 3: proactive interaction between AAS using Industrie 4.0 language. Type 3 extend type 2 with active behavior. They can start to communicate and to negotiate on their own. VDI/VDE 2139 defines a language for type 3 AAS. The AAS I40 language specifies interaction patterns and submodels, which communicate proactively by the defined interfaces via http/REST, OPC UA or MQTT. An AAS Server Application with a corresponding interface for the I40 language is at the same time a client and a server and is a so called proactive AAS.

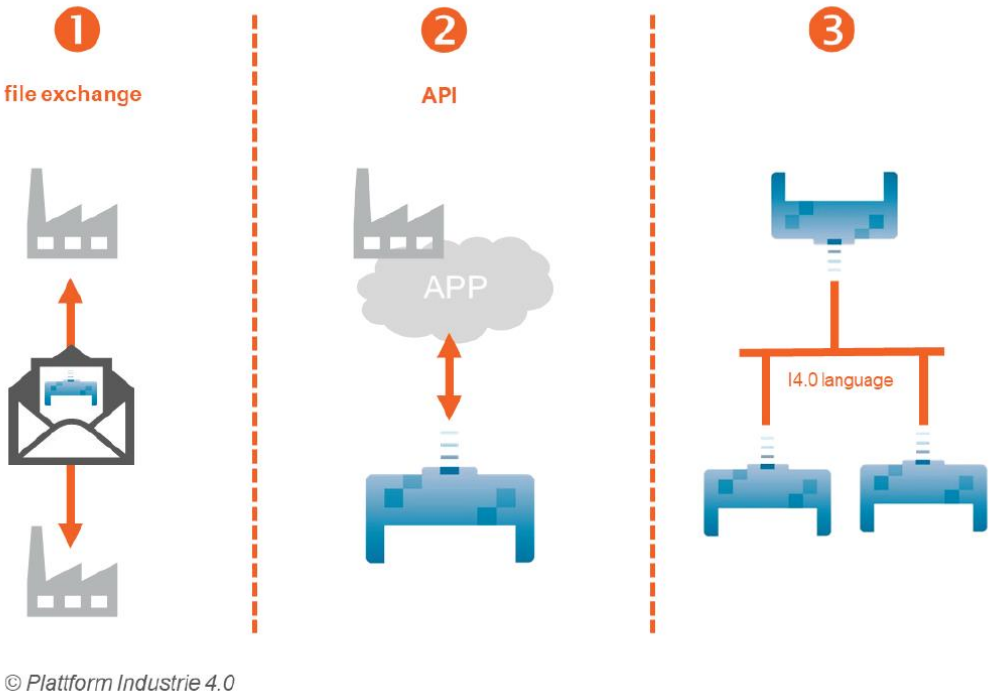


Figure 6: Types of information Exchange via AAS

AAS type II or type III run on top of the physical device making the machines interoperable. The AAS is a common information point for data about the asset offering key innovations such as: searchable on the internet, explorable, data integration across the lifecycle, generic concept across vendors.

2.3 AAS, AutomationML and OPC-UA

Industry 4.0 platform recommend the use of AAS, OPC-UA with its associated information models, and AutomationML, as key technologies offering comprehensive concepts for unified digital interoperability between I4.0 components (e.g., machines and systems) through their life-cycle. These technologies are promoted by different industrial associations, looking for open standards, common digital object models, and harmonization of solutions. AAS, OPC-UA and AML follow the same concept, aiming to develop object models templates and instance creation.

At the same time, this recommendation create confusion in the industrial domain since these technologies seem to overlap with each other and duplicate development efforts, e.g., double modelling, double standardization. To moderate and avoid frustration industry need consensus and comprehensive guidelines. Recently, a group of industrial associations AutomationML eV, IDTA, OPC Foundation, and VDMA have joint efforts to provide a common understanding and deliver a common message on how to use and combine AAS, OPC-UA and AutomationML, delivering the big picture of interoperability. The original discussion paper can be found in german language [[9]]. This section summarizes the first findings and orientations.

AutomationML is an XML based object-oriented modelling language and a data format at the same time to store and exchange information. It allows the modelling, storage and exchange of engineering models covering multitude of relevant aspects of engineering (CAD, electrical planning, mechanical planning, simulation, PLC programming). There is no software stack behind AML do not provide any functionality. OPC-UA can provide the communication layer for rump-up the AutomationML object model information exchange. AML is mainly use in engineering.

OPC-UA is a platform independent open standard (IEC62541) for industrial communication and data exchange. It defines an object-oriented data model and a flexible communication protocol for

industrial automation. It provides a software framework with key features enabling access to operative data, live data, historical, built-in security features, and is extensible to field and cloud communication. OPC-UA is mainly use in operation.

AAS is object metamodel for data exchange. AASs run usually in the cloud, not on the shopfloor, and are focused on lifecycle spanning information. There are different working groups (e.g., IDTA) developing submodels of several aspects of an assets. AAS usually do not provide real-time data as OPC-UA or detail engineering data as AML provides. The aim of the AAS is to help people to find the information they need about an asset throughout its lifecycle. Not all the detail information has to be modelled in the AAS, only what is required for the use case.

The interaction of these technologies AAS, OPC UA and AutomationML becomes clearer when they are mapped to RAMI4.0. The x-axis describes the life cycle of the assets, the y-axis describes the communication content of the assets, and the z-axis describes the hierarchy levels of the networks according to "ISA95" (see Figure 7), in which asset data is communicated.

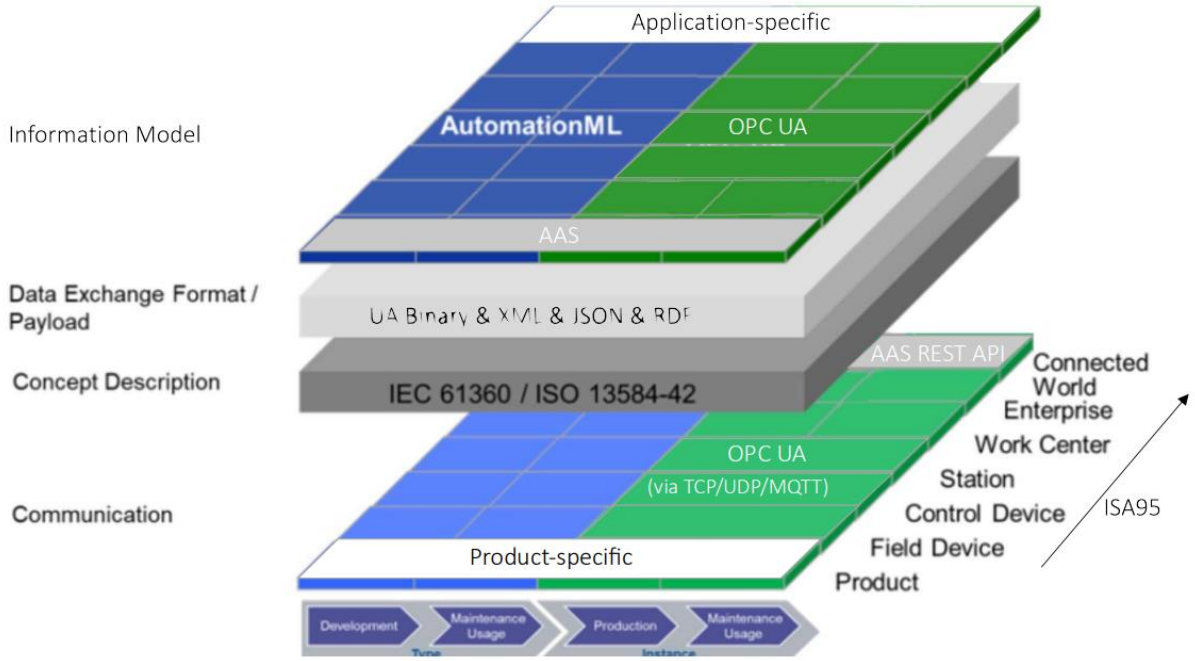


Figure 7: Open Standards in RAMI4.0 (figure from [[9]]).

The key message from [[9]] is to avoid double modelling and use the appropriate technology for each phase in the lifecycle. Use AAS for exchanging data across the lifecycle, OPC-UA for operational data (access to live data of devices, plant, applications), AML for exchanging engineering information. Before modelling check with domain experts and the respective organisations (e.g., IDTA, OPC, AutomationML, VDMA) what is already there and reuse what is existing.

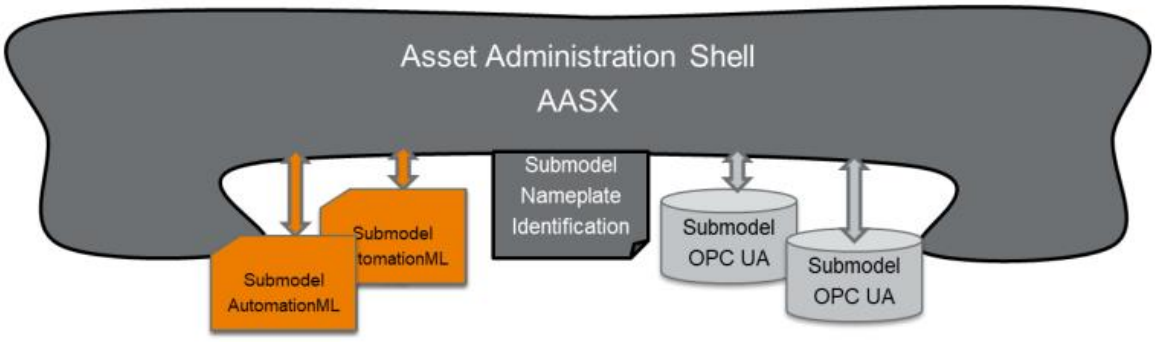


Figure 8: Composition of an AAS from a base model and submodels for engineering an operation (from [[9]]).

The AAS assumes here a predominantly organizing and aggregating role. The AAS can store static and dynamic data. The production information can be fed into the AAS via OPC UA in a certain interval. However, the most sustainable approach for implementing the AAS seems to be a hybrid mix of technology formats, having AAS submodels to describe how to access OPC-UA data (e.g., referencing an OPC-UA nodeset file), and other submodels to offer and access point for AutomationML models.

3 Open-source tools

In Zero-SWARM we aim to develop edge digital twins with AI capabilities and provide the necessary open-source software for enabling their implementation. Open-source software is recognized as a decisive factor for implementing I4.0 successfully [[10]]. In this section we briefly review relevant existing open-source software that can be utilized for this implementation. We group the technology in three blocks: **AAS tools** – most prominent AAS ecosystems for modelling and development -, **Middleware** – that may facilitate to address the integration layer in RAMI4.0, i.e., the synchronization of the assets with the AAS API -, **Edge AI** – software tools that could be used to embed AI and ML/ DL models in the edge digital twins-.

3.1 AAS Tools

3.1.1 AASX Server

The AASX Server [[11]] is being developed in the context of IDTA. The main package is the AASX Package Explorer, a software tool with a graphical user interface to view and edit Asset Administration Shell. Through its graphical interface, it is aimed at both tech-savvy and less technical users to develop and demonstrate the use of Asset Administration Shell standard.

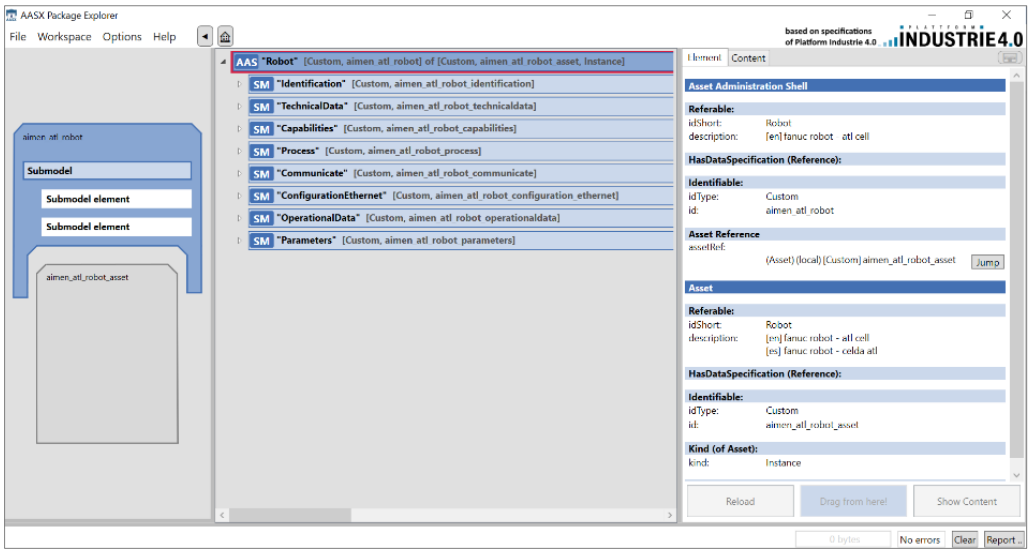


Figure 9: AASX Package Explorer user interface

Using the AASX Package Explorer, the AAS can be instantiated in different formats like JSON, XML, AutomationML or nodeset.xml to use in a OPC UA server, but the main format used to save the AAS instances are AASX files. With those format files, the assets can be registered and be used in Compliance with the IEC 63278-1 standard.

The AASX server includes a C# client library, the model editor, a registry service and visualization. AASX does not claim to provide an AAS-compliant REST API. The API supports the operations of reading data but not the ones related to creating, updating, or deleting elements. It supports HTTP and OPC UA as API implementations. Main drawback is that AASX Server provides limited support for asset synchronization.

3.1.2 NOVAAS

The NOVA Asset Administration Shell (NOVAAS) is an open-source reference implementation and execution environment for AAS designed with the AASX package explorer and developed by NOVA School of Science and Technology [[12]]. It follows a low-programming paradigm using Node-RED and Node.js. It was designed to be cloud native. It provides the back-end, front-end and web-based GUI for creating dashboards. The ecosystem does not include a client library, model editor, model validation or registry. The asset synchronization is based on Node-RED offering the possibility to add predefined extension points in the flow to synchronize assets via the read interaction pattern with HTTP and OPC UA.

3.1.3 Eclipse BaSyx

Eclipse BaSyx [[13]] provides a complete ecosystem for implementing AAS type I, II, including AAS server, AAS Registry, AAS Web client, and an SDK available in Java, C# and C++. Main contributor is Fraunhofer IESE. It provides different ways to address the synchronization with the asset: custom coding using a component called DataBridge – a service external to the AAS Service that can send data through HTTP -, or via a mechanism called property/operation delegation that allows to read property values and executing operations via HTTP.

Eclipse AAS Model for Java implements the specification of the Asset Administration Shell (AAS) such as metamodels, submodels, serialization and deserialization modules, validators, and transformation libraries based on the AAS specifications.

The Eclipse AAS Model for Java projects are focusing on the following features / functionalities:

- AAS Java Generator: The Java Generator automatically updates AAS Java Model classes and interfaces upon change in the specification documents.
- AAS Java Model: The AAS Java Model enables users to model and to represent their Digital Twins using AAS metamodels and submodels.
- AAS Java Serializer: The AAS Java Serializer enables users to serialize the AAS model into JSON, RDF, XML, AASX, OPC UA, AutomationML. This feature is necessary to support the import/export of AAS from/to these formats and validate them.
- AAS Transformation Library: The AAS Transformation Library brings the content from existing industry-standards such as OPC UA and AutomationML to the AAS.

3.1.4 FA³ST

FA³ST is a Java-based software ecosystem used to create and manage I4.0-compliant, hybrid and data sovereign digital twins. It is developed and maintained by Fraunhofer IOSB [[14]]. Main features are: ensure the usage of the latest AAS specifications, easily extendible configuration, provides support for hybrid models, can be used with or without IDS. It can integrate Apache StreamPipes as external runtime services for adding stream processing capabilities to the AAS. It provides tools for AAS visualization and management, registry, client, OPC UA Crawler form mapping information models and ASS configurator.

3.2 Middleware

An IoT middleware is software that serves as a bridge between IoT devices and the applications that use them. Its primary purpose is to provide a standardized way to connect and manage IoT devices. IoT middleware typically includes features such as device management, data acquisition and processing, security, and integration with other systems. By providing a unified interface for interacting with IoT devices, middleware simplifies the development process and enables developers to focus on creating innovative applications that leverage the unique capabilities of IoT devices.

3.2.1 Apache Kafka

Apache Kafka is an open-source distributed event-streaming platform that can publish, subscribe, store and process log streams in "(soft) real-time" [[15]]. It is designed to handle data streams from multiple sources and deliver them to multiple consumers. Kafka is highly scalable, handling millions of data points per second, which makes it well suited for the Internet of Things (IoT), where data can grow exponentially. Because of these functionalities, this technology is used by most of the largest manufacturing companies.

Kafka can be used as edge combining data integration (Kafka connect), data processing (Kafka streams, KSQL), data storage (Kafka core) and ease of integration with artificial intelligence system framework in a single system Figure 10: Kafka architecture for stream processing.

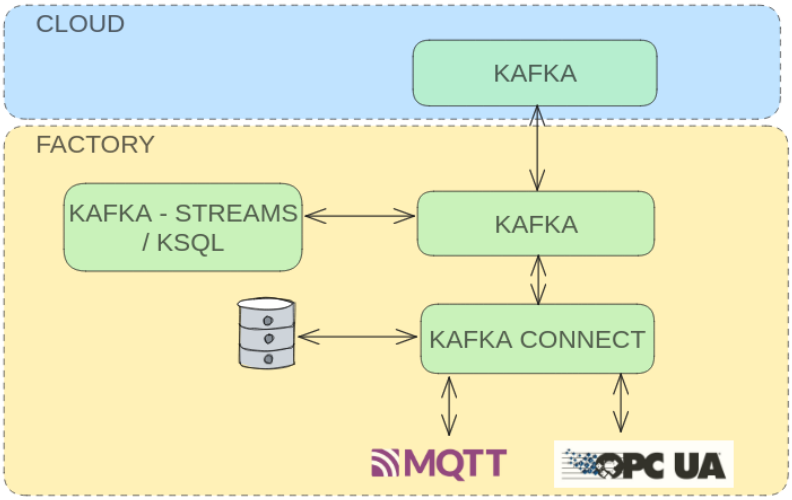


Figure 10: Kafka architecture for stream processing on the edge.

3.2.2 Redis

Redis, stands for Remote Dictionary Server, is a fast, open source, in-memory, key-value data store [[16]]. It is mainly used as a database, cache, message broker, or queue. It runs in-memory, which enables low-latency and high throughput with sub-millisecond queries. Redis also provides advanced features and modules that extend its capabilities in edge environments such as:

- RedisGears – a programmable engine for Redis that runs inside Redis, closer to where your data lives, and which allows cluster-wide operations across shards, nodes, data structures, and data models at a sub-millisecond speed.
- RedisAI – a machine learning data type that runs inside Redis and allows you to train and predict on your data. Additionally provides a common layer among different formats and platforms, including PyTorch, TensorFlow/TensorRT, and ONNX Runtime.
- RedisTimeSeries – a time series data type with capabilities like automatic downsampling, aggregations, labeling and search, compression, and enhanced multi-range queries as well as built-in connectors to popular monitoring tools like Prometheus and Grafana to enable the extraction of data into useful formats for visualization and monitoring.

Figure 11: Redis diagram used for real-time analysis and representation of video streams on the edge shows an example of Redis used for real-time video analytics. Redis has partnered with the emerging leaders in the IoT edge platform space, EdgeX Foundry and Microsoft Azure IoT Edge, to bring RedisEdge to their users.

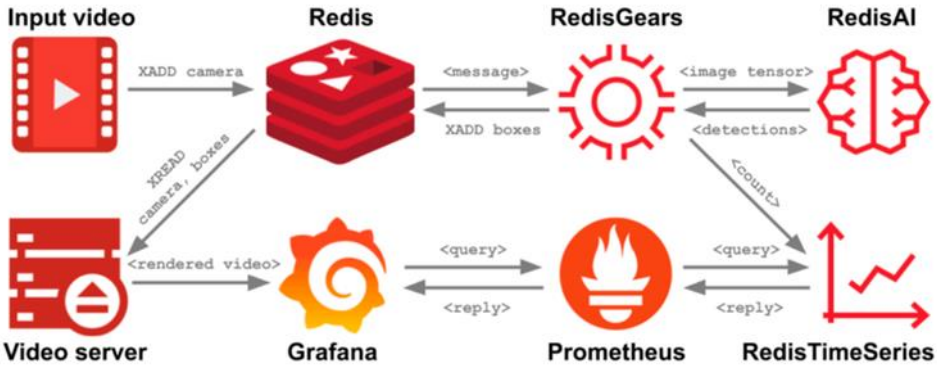


Figure 11: Redis diagram used for real-time analysis and representation of video streams on the edge.

3.2.3 EDGE X Foundry

EdgeX is a common open framework for IoT edge computing providing an ecosystem of interoperable components to accelerate time to market and facilitate scaling [[17]]. It is managed by the Linux Foundation.

The objectives of the EdgeX Foundry are:

- Build and promote EdgeX as the common open platform that unifies Internet of Things (IoT) edge computing.
- Enable and encourage the growing community of IoT solution providers to create an ecosystem of interoperable plug-and-play components around the EdgeX platform architecture.
- Certify EdgeX components to ensure interoperability and compatibility.
- Provide tools to rapidly build EdgeX-based IoT solutions that can easily adapt to changing business needs.
- Collaborate with relevant open-source projects, standards groups and industry alliances to ensure consistency and interoperability across the IoT.

3.3 Edge AI

Edge AI refers to the use of artificial intelligence (AI) algorithms and models at the edge of a network, closer to the source of data. This contrasts with traditional ML and AI approaches, which often involves processing data in a centralized location, such as a data center or cloud. Edge AI is used in situations where it is not feasible or practical to transmit large amounts of data to a central location for processing. For example, it can be used by field devices, where data is generated by sensors. Edge AI algorithms are designed to be lightweight and efficient, so they can run on devices with limited processing power and memory. They can also be designed to operate in real-time, making them useful in applications of autonomous systems and robots. The technology enables the development of standalone intelligent digital twins that can operate in a decentralized manner and make decisions based on local data.

3.3.1 Gstreamer

GStreamer is a free and open-source multimedia framework that enables the creation of a wide range of media processing system with complex workflows [[18]]. It is written in C based on Glib and provides a simple API for constructing processing pipelines combining a series of pluggable components which can be connected to form a data flow from an input source to an output destination. These components can be used to perform a wide range of operations such as encoding, filtering, multimedia manipulation. It is being used for creating AI powered video analytics applications and services that

can be deployed on the edge and connected to cloud services. Intel (Deep Learning Streamer [[19]]) and Nvidia (DeepStream [[20]]) have recently integrated GStreamer into their deep learning inference tools to provide developers with a flexible and powerful multimedia processing framework for deep learning applications.

3.3.2 ONNX Runtime

ONNX (Open Neural Network Exchange) is an open format for representing machine learning models. It was developed by Microsoft and is now maintained by the ONNX community [[21]]. ONNX provides a standard way to represent deep learning models, making it easier to share and deploy models across different platforms and applications. ONNX supports a wide range of neural network architectures, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more. ONNX also provides tools and libraries for converting models between different deep learning frameworks, such as TensorFlow, PyTorch, and Keras. ONNX is used in a variety of applications, including image and speech recognition, natural language processing, and computer vision.

ONNX Runtime is an open-source deep learning inference engine that enables the deployment of ONNX models on a variety of devices and platforms. It was developed by Microsoft and is now maintained by the ONNX community. ONNX Runtime provides an efficient and portable way to run ONNX models, making it easy to deploy models on edge devices, such as IoT devices and mobile phones, as well as in the cloud. ONNX Runtime supports a wide range of hardware and software platforms, including CPUs, GPUs, and specialized accelerators, such as Tensor Processing Units (TPUs).

3.3.3 NVIDIA TensorRT

NVIDIA TensorRT is an SDK for high-performance deep learning inference, includes a deep learning inference optimizer and runtime that delivers low latency and high throughput for inference applications. The SDK is free but is not open-source software [[22]]. We mention it here because it provides best performance across many industries standard benchmark. TensorRT can optimize and deploy applications to NVIDIA embedded system-on-chip solutions as Jetson family boards.

3.4 Discussion

Section 3.1 summarized the most relevant open tools for the creation of AAS: AASX server, Eclipse BaSyx, NOVAAS and FA³ST, following the review work in [[23]]. All implementations support at least a minimal set of required functionalities but none of them implement the full AAS specification. Most solutions provide a complete ecosystem or platform, including services for modelling, repository of AAS, registry discover functionalities, visualization tools. The implementation of AAS is still challenging and there are incompatibilities among different implementations. The main reason is that AAS API specification is still a work in progress and there is no final stable release, expected to change soon.

In addition, and more importantly, a common point that is not clearly or completely addressed by the exiting AAS frameworks is the synchronization of the AAS with the asset. Synchronization here refers to the integration layer in RAMI4.0. Although the integration layer is not cover by the AAS specification, it is an essential feature of a digital twin, and usually work left to the system integrators. Besides, the integration of AI embedded processing capabilities in the digital twins is not addressed properly by any framework. BaSyx offers the possibility to address the synchronization task by custom coding using a DataBridge component. NOVAAS uses Node-RED as a backend offering different connectors and the possibility to customize some extending elements. FA³ST integrate a processing pipeline mechanism using Apache StreamPipes but this is link as an external AAS service to enhance capabilities.

One approach to implement the integration layer and synchronize the AAS with the asset is to use an existing open-source middleware. Among most suitable solutions identified: Kafka, Redis, Edge X Foundry since they already provide several connectors and stream processors. However, their use for data aggregation and AI processing from different sources in an embedded devices and providing e.g., OPC-UA interface in the northbound is not straightforward.

For the deployment of stream processing pipelines with AI capabilities in the edge, Gstreamer offers a well-established open-source framework that has been recently integrated by big IT companies such as NVIDIA and Intel who offer plugins based on ONNX and TensorRT for optimising Deep Learning models for running in in embedded hardware solution combining CPU and GPU.

4 Open Toolkit for implementing AAS

The aim of this task is to provide a comprehensive methodology for developing reactive AAS (type II) of field devices and machines in the operation phase and open tools supporting this development.

More specifically, we aim two develop two main open tools:

- **A novel AAS editor**. A web-based editor aiming to simplify the modelling of the assets by providing a library of templates.
- **Edge computing framework** to enable the implementation of digital twins with embedded near real-time AI processing capabilities and AAS API (e.g., integrating an OPC-UA server). The intention is to have standalone intelligent AAS that can run in the asset itself or in edge devices close to the assets following the edge computing paradigm. The framework will support developers to address the integration layer in RAMI4.0 (i.e., the synchronization of the physical asset with the AAS) and the creation of complex processing pipelines with AI capabilities in a simple way.

In Zero-Swarm the aim is not to create a new complete AAS platform or ecosystem. The idea is to complement existing solutions by providing tools to support the development of other types of AAS (e.g., edge AAS) with different functionalities. Ideally, once the AAS specification will be stable, all AAS ecosystems should be compatible with each other.

4.1 AAS modelling tool

As reported in Section 3, there are available generic tools and libraries for AAS modelling, but there is a lack of a tool able to support and guide a user to model the necessary information when it comes to communication protocols (e.g., OPC UA). Some tools (i.e., AASx Package Explorer) have the possibility to use some templates for the submodel definition, but using a non-friendly user interface as well as the current version is not fully aligned with the standard IEC 63278 (definition of the structure of the Assset Administration Shell); whereas another tools (i.e., Basyx) have more potential and flexibility, but using command line.

Nowadays the AASx Package Explorer is one of the most known and powerful C# based viewer / editor for Asset Administration Shell. This tool has the functionality of import and export submodels and the complete AAS for different formats (i.e., Json, AutomationML and xml-based formats like OPC UA Nodeset2.xml). Regarding the OPC UA information is possible to export part of the information needed for the creation/configuration of an OPC UA server; but some information is missing (i.e., Namespaces, Profiles, Facets). Currently, IDTA is working on a template (*OPC UA Server Data Sheet 02009)* that will include information to find and access the OPC UA server and information about the content implemented in the OPC UA server. But this submodel template is not available and will only allow the definition of the information and not the export into the xml needed for the OPC UA server.

Taking into account the limitations found in the current frameworks, in Zero-Swarm AIMEN will develop a user-friendly modeling tool for Asset Administration Shell, paying special attention of the information needed for communication protocols definition (i.e., OPC UA Server).

The AAS toolkit will be a powerful and user-friendly web interface for modeling, and managing AAS, allowing users to navigate and understand the structure and relationships of the digital twin easily. This tool will comply with the latest AAS standards (IEC 63278) and it also will include validation

capabilities to ensure compliance with defined standards and constraints helping end-users to adopt this technology as well as templates and guidelines for the users.

The main features/functionalities of this toolkit are:

- Web based application based on Angular framework.
- Create AAS digital twins of a manufacturing cell and physical resources using templates.
- Export the AAS and submodels into AASx, Json and xml format following the standards and recommendations, compatible with the AASs generated with other frameworks.
- Templates for defining OPC UA server information. If the official template is available this will be considered and available in the toolkit.
- Export nodeset2.xml and XXXX.xml, as configuration files for the generation of the OPC UA server that can be later uploaded/used in an edge device.
- AAS registry and database.

4.2 Framework for developing edge digital twins with AI

The edge computing framework is an embedded framework that facilitates the development and deployment of high-performance stream processing applications in edge devices providing interoperability with IIoT. The core of the system has been developed by AIMEN in H2020 MULTIPLE project (GA 871783) with the purpose of providing a common framework for developing embedded vision applications.

The architecture of the framework is inspired by Gstreamer, combining a set of nodes (data sources, data sinks, filters, and I/O modules) that can be connected to create complex processing pipelines. Gstreamer was originally designed for developing audio/video multimedia applications where data packet synchronization is a must. Therefore, Gstreamer has associated some overhead related to e.g., packet timestamp, synchronization, signalling mechanism, etc. that may harm the latency when processing high dimensional data. This will depend on the memory management and the implementation quality of some existing plugins that are usually difficult to decipher. Besides, the learning curve for developing new plugins for Gstreamer is high, since it was implemented in C based on Glib objects. The main aim of the new framework it to overcome some limitations of Gstreamer, providing a fast, low latency and lightweight framework enabling the efficient processing of high dimensional data exploiting two key points:

- Minimizing memory copies among nodes processors. Zero-copy processing.
- Maximizing use of sequential memory reads and writes (batch processing).

Other design principles and features are:

- Flow-based programming pattern.
- Low latency and efficiency.
- Programmed in C++.
- Compatible with Linux-based systems. Windows support under discussion.
- Concurrent programming exploiting e.g., multicore CPU and GPU.
- Signaling mechanism based on fast C++ delegates.
- Inference deep learning engines based on ONNX runtime, TensorRT.
- API via embedded OPC-UA server enabling modification of settings online.
- Synchronized data acquisition and aggregation

- Easy configuration of the processing pipelines with YAML files.

Current implementation provides the following library of nodes supporting the connection of industrial devices:

- Data source elements: GeniCam GigE/USB Cameras, Profilometers, Modbus.
- Data Sink elements: FileSink, RTSP, RestAPI, Display.
- Filters: Inference engine based on TensorRT, Inference engine for ONNX runtime.
- I/O modules: OPC-UA, TCP/ROS bridge, Profibus, Digital Analg and Digital I/O

Each node has settings, i.e., parameters and methods, that can be read/write and invoked online, and queues for data sharing. Each node can have several data queues, i.e., data objects for streaming data managing concurrent access. The framework allows developers to easily implement additional custom nodes in C++ using templates. Nodes can be connected through queues to create complex processing pipelines. The pipeline can be instantiated and configure in code or using a simple yaml config file. The compiled pipelines can be deployed in any Linux-based device with different computing capabilities. It has been tested for instance in embedded devices using SoC form NVIDIA integrating ARM and GPU.

Figure 12: Edge computing devices integrating Nvidia Jetson System-on-Chip with ARM/GPU and Raspberry Pi.

One key feature of the framework is the ability to create an embedded lightweight OPC-UA server using the open-source OPC-UA stack Open62541. This server node can expose all the settings of each node in the pipeline and the data streamed in the queues through an OPC-UA interface.

Next figure shows a very simple example of a processing pipeline for implementing a machine vision system. The pipeline is composed by three elements: Camera Node (for interfacing the camera through GigE), Display (local GUI for visualising the video stream), and FileSink (local storage of the data stream). The OPC-UA node implements an embedded server that enables the control of each element in the pipeline remotely and streaming of images as 2D arrays. Next figure shows the scheme of the pipeline configuration. The black arrows indicate data streaming, blue arrows events used for node settings control and monitoring. The Q indicates a queue object handling concurrent access to the resources. Figure 14: UAExpert GUI showing OPC-UA server structure of the edge computing pipeline from Figure 13.Figure 14 shows the OPC-UA server structure inspected with a third party OPC-UA client (UAExpert).

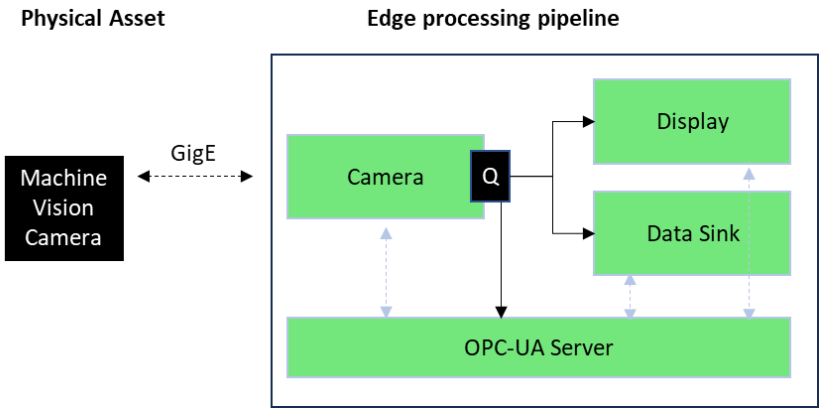


Figure 13: Simple example of a processing pipeline with OPC-UA embedded interface

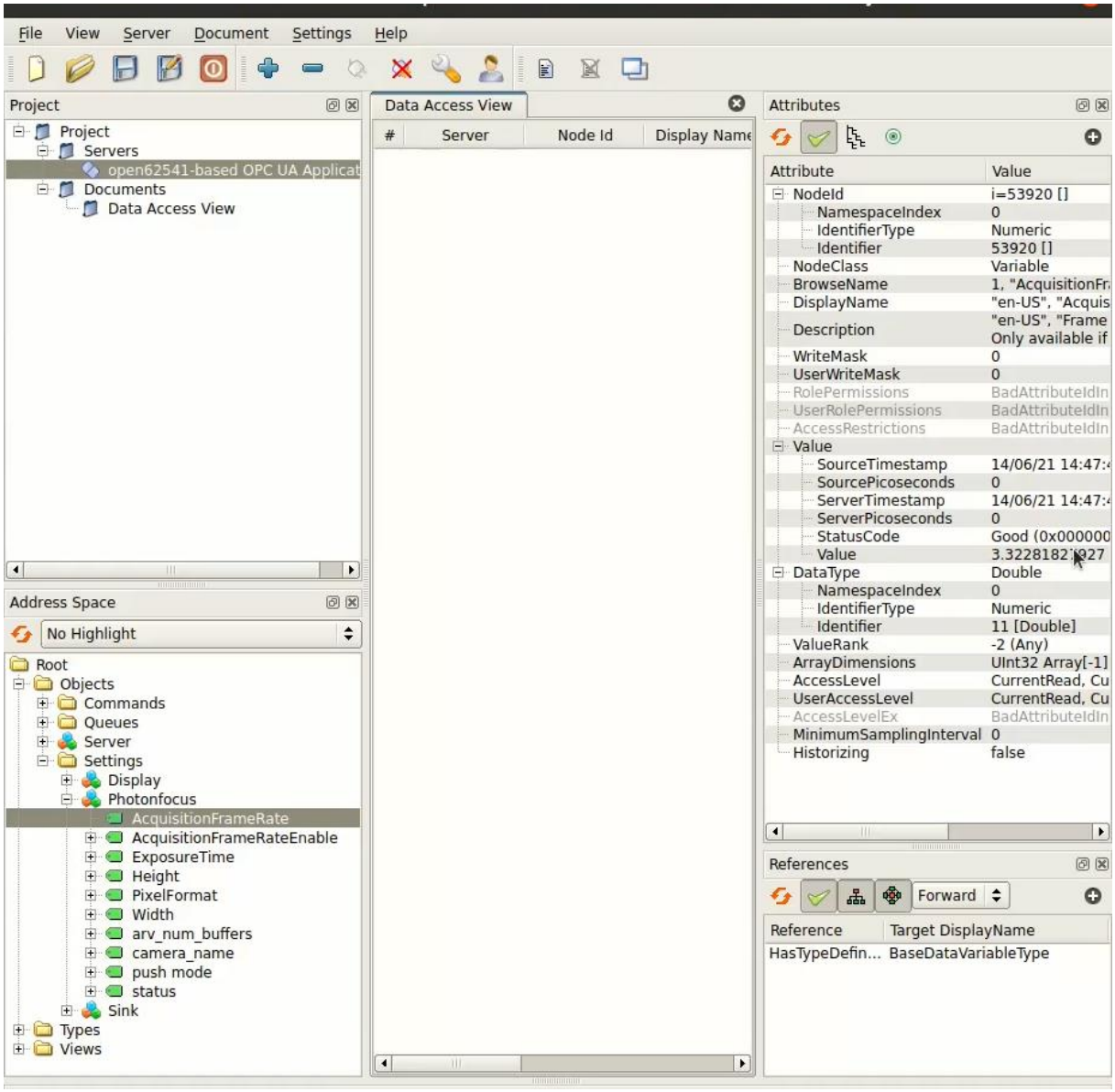


Figure 14: UAExpert GUI showing OPC-UA server structure of the edge computing pipeline from Figure 13.

The purpose in ZeroSwarm is to release the Edge Computing Framework as an open-source stable repository for AAS implementation. This requires addressing the following challenges:

- Refactoring of the framework adding new core functionalities. The refactoring will affect the complete framework simplifying the core classes, current nodes, and improving the singling mechanism via events. Tasks related to the development of good documentation, improve styling and port the build system to CMAKE are ongoing. The aim is to deliver a easy to use and comprehensive solution.
- Integration of given Information models and AAS API with an embedded OPC-UA server. The main challenge to address in ZeroSwarm is to map the current OPC-UA server structure with setting and data from queues as shown in Figure 14 to a given information model following AAS or OPC-UA. The aim is to provide an easy solution for doing this mapping / binding of data, properties and settings without the need of writing much code.

The foresee structure of the edge AAS implemented with Edge Computing Framework pipelines is showed in the next figure:

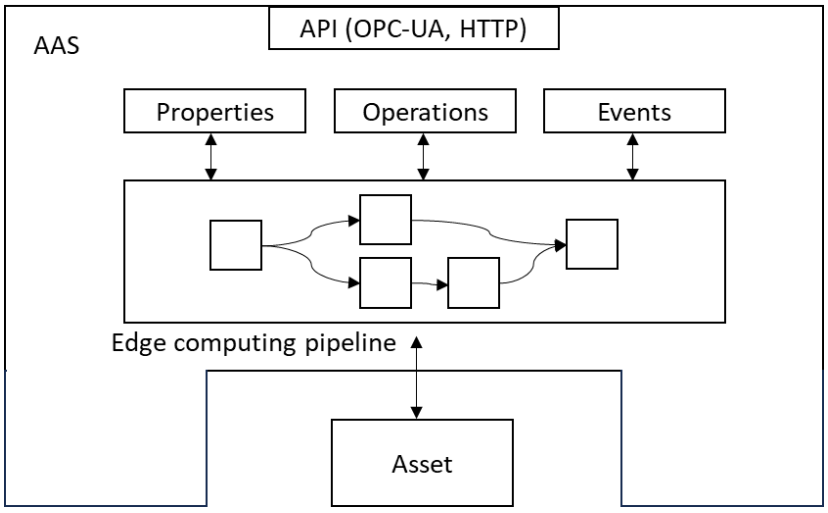


Figure 15: AAS proposed structure.

The edge computing pipeline will be responsible for synchronising the Asset with the AAS properties, operations, events, and provide near real-time processing capabilities. Besides, it facilitates the integration of an OPC-UA server embedded within the AAS implementing the API specification. The implementation of an additional API based on HTTP is under discussion. The structure is similar to the one proposed by other open-source frameworks like FA³ST [[23], [24][or [[25]]. The main difference is the edge computing pipeline, which is embedded in the AAS and not linked as an external service outside the AAS as in FAAST, and the functionalities it provides such as online settings modification, bidirectional communication, AI inference in the edge.

4.3 AAS implementation workflow

For the realization of an edge AAS, these are the foreseen steps:

1. Create the AAS model, considering the standardized metamodel, and following these phases of modelling.

   1.1: identify the asset and study its information (i.e., checking the available technical datasheets).

   1.2: define the structure using the AAS metamodel. This includes the division of the information into submodels.

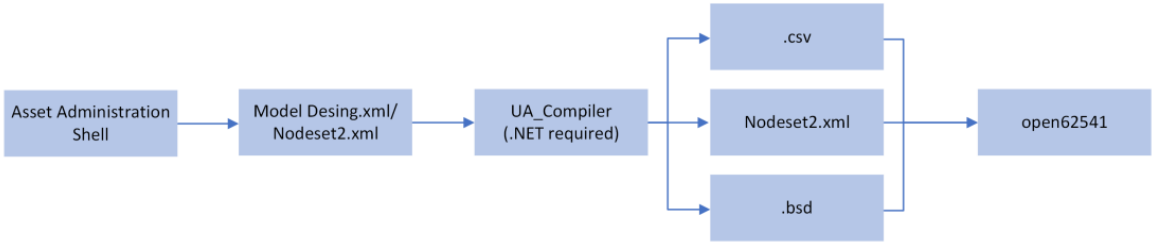
   Following the recommendations in [[8]], we should avoid double modelling and double standardization. Therefore, the first thing is to check with the domain experts if there are existing information models already available for our devices and application (e.g., OPC-UA companion specifications by VDMA, IDTA templates, AutomationML). If there are no information models that satisfy our requirements, we can start the development of a new template of a specific submodel using the web-based AAS editor or the AASx Package Explorer.

   1.3: define the communication of the AAS with the external elements. This step includes the identification of the interesting data that can be sent from the edge device to a cloud/server/repository.

   1.4: Serialization of the model in Nodeset2 XML. The toolkit will provide the functionality of export the AAS model into a Nodeset2 XML format, so this step will be a transparent to the user of the toolkit.

2. Development of the AAS instance.

2.1 Transform the Nodeset2 XML into datafiles needed by the Open62541 stack to create the OPC-UA server instance providing .c/h files. For the transformation there are available tools from OPC foundation such as *UA_Compiler*.



2.2. In parallel, create the processing pipeline using the Edge Computing Framework to interface the devices and external elements following 1.3 and implement the required data aggregation and processing functionalities.

2.3. Bind the properties and operations of the metamodel in the OPC-UA server with the data and settings of the pipeline elements. Most probably this step will require the modification of a config file. Recompile to generate the final executable.

3. Deployment of the AAS in the edge devices and verification.

5 AAS implementation examples

The development of an AAS of a metal additive manufacturing (AM) cell in the south node trial 3 (SN03) -*5G enabled remote quality control for zero defect resilient manufacturing*- has been taken as a reference use case to start the discussion and define the requirements for the ASS toolkit.

Figure 16: Robotized Additive Manufacturing Cell

A robotized metal AM cell additive manufacturing cell is a manufacturing system that uses laser technology to deposit metal powder onto a substrate, layer by layer, to create a three-dimensional component. The robot handles the substrate and controls the movement of the laser. This allows for precise and automated manufacturing of parts with complex geometries. AIMEN is working on centralizing manufacturing data from their current Additive Manufacturing cells into a centralized

cloud platform. Data in the past was captured in a device inside the cell locally in files, and then these files were transferred manually by cell operators to a centralized folder. AIMEN aims to connect several of these Additive Manufacturing cells to the cloud via 5G so they become monolithic portable manufacturing cells that can be easily moved inside a factory (movable with crane) or field (movable inside a container). The cells typically include a laser system, a powder delivery system, an industrial robot, and a machine vision system (camera+ industrial pc) to monitor the melt pool during the entire process.

The goal is to develop an AAS for monitoring the AM cell. Ideally the AAS would be hosted in an edge device in the Cell with 5G connectivity. Several possibilities for modelling the cell and implementing the AAS are being evaluated.

First option is to create a unique AAS model of the complete AM cell including only the relevant monitoring data. Some operational data of interest are the robot position and orientation, laser power, images and melt pool width (feature extracted from the camera with an AI model). Figure 17 and Figure 18 show a possible implementation of the AAS using the edge computing framework, and

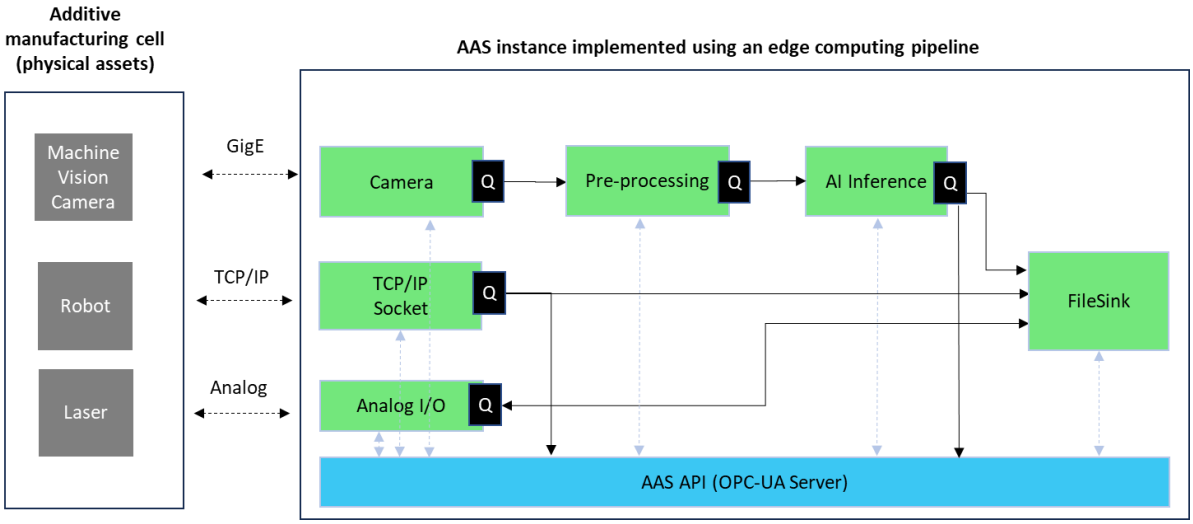


Figure 17: Example of a processing pipeline for implementing an AAS of the complete Additive Manufacturing Cell.

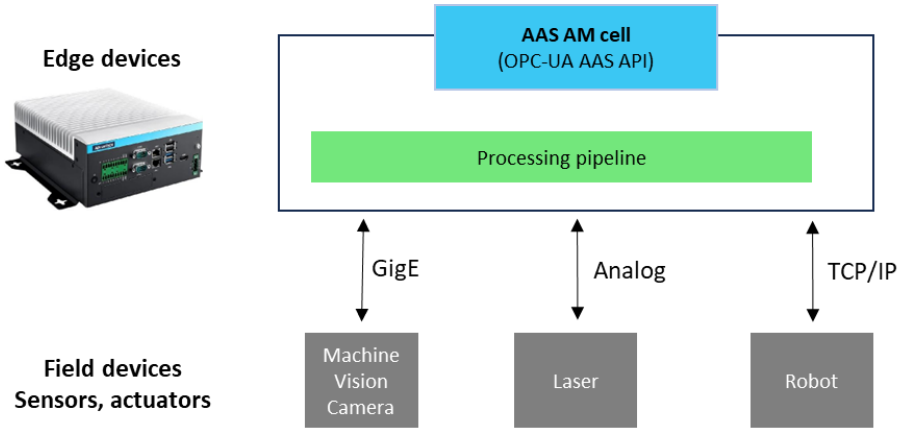


Figure 18: AAS instance of the AM cell with embedded OPC-UA server API

Another option is to create two embedded OPC-UA servers, one for the Robot and one for the Machine Vision System following existing OPC-UA information models defined by VDMA [[26], [27]]. Then an AAS for the complete cell on top of them including submodules referencing the OPC-UA information models. The OPC-UA severs would run in an edge device while the AAS instance would be hosted on a server.

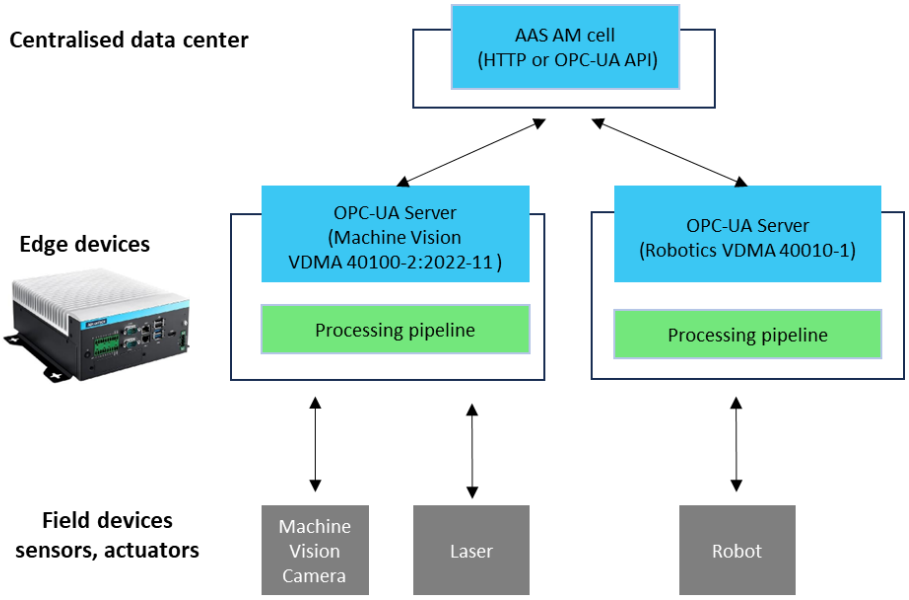


Figure 19: AAS instance of the AM cell on a server (e.g., private cloud) referencing by submodels two OPC-UA embedded servers on edge devices.

One requirement in both cases, is that the OPC-UA server that controls the camera should be able to deploy AI models for online analysis of the video stream. The aim is to extract the image features from the camera, e.g., the melt pool width, and send them to the cloud together with other process parameters (robot positions, laser power). In addition, the AAS should provide a common interface independently of the robot brand that is used in the cell. The intention is to implement both approaches. We are starting with option 1 since is simpler. Once this is operative, we will proceed with option two.

6 Conclusion

We have reviewed the challenges related to the implementation of Digital Twins based on Asset Administration Shell specification for Industry 4.0, and the features and limitations of most important AAS open-source frameworks that allow developers to work with AAS.

Zero-SWARM aims to deliver a comprehensive methodology and novel open-source software for supporting the implementation of reactive AAS (type II) of field devices and machines during the operation phase with embedded AI processing capabilities. These AAS could be deployed in the assets itself or in edge devices close to the machines following the edge computing paradigm. This would enable the implementation of decentralized applications such as distributed control or autonomous system.

Two software tools are being developed in this task and will be published as open-source repositories:

- An AAS web-based modelling tool following the latest AAS specification that will simplify the modelling task using pre-defined templates.
- An edge computing framework that will facilitate the synchronization of the asset with the AAS API embedding an OPC-UA server. This framework enables the creation of flexible and modular processing pipelines combining data sources, filters and sinks modules and can be applied for embedded data aggregation and processing.

An overall workflow for the implementation of new AAS and examples will be provided as templates to support the implementation of AAS in the trials of the project. The final outcomes will be reported in an updated version of this deliverable.

References

[1] Boss, B. et al. (2020). Digital Twin and Asset Administration Shell Concepts and Application in the Industrial Internet and Industrie 4.0. An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper.

[2] Martikkala, A. et al. (2021). Trends for Low-Cost and Open-Source IoT Solutions Development for Industry 4.0. Procedia Manuf. 2021, 55, 298–305.

[3] Plattform Industrie 4.0. Structure of the Administration Shell, continuation of the development of the reference model for the Industrie 4.0 component. Working Paper, April 2016.

[4] Plattform Industrie 4.0. Details of the Asset Administration Shell—Part 1 The Exchange of Information between Partners in the Value Chain of Industrie 4.0. (Version 3.0RC01).

[5] Plattform Industrie 4.0. Details of the Asset Administration Shell Part 2—Interoperability at Runtime—Part 2—Interoperability at Runtime—Exchanging Information via Application Programming Interfaces: (Version 1.0RC01). 2020.

[6] IEC 63278-1:2023 PRV (Prerelease version). Asset Administration Shell for industrial applications – Part 1: Asset Administration Shell structure.

[7] Jacoby, M. et al. (2020). Digital Twin and Internet of Things—Current Standards Landscape. Applied Sciences 10, no. 18: 6519.

[8] Darth, R. et al. (2023) Discussion paper – Interoperability with the Administration Shell, OPC-UA and AutomationML. Target image and recommendations for actions for industrial interoperability. Discussion paper 2023 – Discussion paper 2023 – AutomationML.

[9] https://www.automationml.org/wp-content/uploads/2021/07/PosPapier_Interrelation-of-AAS-and-AML.pdf

[10] Bitkome, V. (2021). Open-Source Monitor – Survey Report 2021.

[11] https://github.com/admin-shell-io/aasx-server

[12] https://gitlab.com/novaas/catalog/nova-school-of-science-and-technology/novaas

[13] https://www.eclipse.org/basyx/

[14] https://github.com/FraunhoferIOSB/FAAAST-Service

[15] https://kafka.apache.org/documentation/

[16] **Error! Hyperlink reference not valid.**https://github.com/RedisGears/EdgeRealtimeVideoAnalytics

[17] https://docs.edgexfoundry.org/2.3/

[18] https://gstreamer.freedesktop.org/

[19] https://dlstreamer.github.io/

[20] https://developer.nvidia.com/deepstream-sdk

[21] https://onnx.ai/

[22] https://github.com/NVIDIA/TensorRT

[23] Jacoby, M. et al. (2023). Open-Source Implementations of the Reactive Asset Administration Shell: A Survey. Sensors. 23. 5229.

[24] Jacoby, M. et al. (2021). An Approach for Realizing Hybrid Digital Twins Using Asset Administration Shells and Apache StreamPipes. information (Switzerland). 12. 10.3390/info12060217.

[25] Pribiš, R. et al. (2021). Asset administration shell design methodology using embedded OPC unified architecture server. Electronics (Switzerland), 10(20).

[26] https://www.vdma.org/viewer/-/v2article/render/67592778

[27] https://reference.opcfoundation.org/Robotics/v100/docs/