http://dx.doi.org/10.25673/111743

Automatic generation and orchestration of active Asset Administration Shells with IO-Link

Benjamin Evans[1] Victor Chavez[2] Jörg Wollert[3]

Abstract: This paper presents a proof of concept for automatically generating and orchestrating active asset administration shells (AAS) with IO-Link. AAS are software-based representations of physical assets that enable interoperability and standardised communication across different industrial systems. IO-Link is a widely adopted communication protocol for sensors and actuators in industrial automation. Our method uses an approach to generate AASs based on the IO-Link device description files. The generated AASs can then be orchestrated to form a distributed system that provides dynamic information about the status and performance of the connected assets. We demonstrate the effectiveness of our method through a proof of concept that involves the automatic generation and orchestration of AASs for a fluid processing unit equipped with pressure and flow sensors and a pump. The results show that our approach reduces the time and effort required to create and maintain active AASs.

Keywords: Asset Administration Shells; IO-Link; Industry 4.0

1 Introduction

The onset of the fourth industrial revolution has led to the emergence of smart factories, where intelligent field devices generate vast amounts of data that can be used for tasks such as monitoring and predictive maintenance. However, this technological advancement has also brought about a surge in interfaces, data streams, and documentation, posing navigation challenges for companies and their engineers [Gr20],[Re17]. Questions about which devices are installed where, which manuals correspond to specific devices, and other similar concerns have become daunting challenges in the modern industrial landscape.

In response to these challenges, the concept of digital twins has emerged as a promising solution[Kr18]. These digital replicas of physical assets act as repositories for a wealth of data and documents that might otherwise be scattered and elusive. While digital twins have shown great potential, there is a growing recognition that a standardised approach is needed to harness their full benefits. This is where the Asset Administration Shell (AAS) comes into play, representing not only a shift towards standardisation but a transformative leap for modern industrial devices in their entirety.

[1] FH Aachen, Institut für angewandte Automation, Goethe Straße 1, 52064 Aachen, Deutschland benjamin.evans@alumni.fh-aachen.de

[2] FH Aachen, Institut für angewandte Automation, Goethe Straße 1, 52064 Aachen, Deutschland chavez-bermudez@fh-aachen.de

[3] FH Aachen, Institut für angewandte Automation, Goethe Straße 1, 52064 Aachen, Deutschland wollert@.fh-aachen.de

AASs offer a framework that allow companies to employ consistent file standards and communication platforms across various field devices, fostering seamless integration and management. However, the challenge lies in the creation and configuration of AASs to efficiently gather and encapsulate this critical information.

This paper is dedicated to addressing this fundamental challenge and presents a proof of concept that demonstrates a solution based on IO-Link. We delve into the intricacies of AAS creation and setup, showcasing a practical approach to streamline the management of digital twins for contemporary industrial devices. In doing so, we aim to pave the way for more efficient, standardised, and interoperable industrial systems in the era of Industry 4.0.

2 Background

AASs have gained prominence in the German industrial landscape, promoted as the standard for digital twins by Plattform Industrie 4.0. This initiative, backed by Germany's Ministry of Economic Affairs and Climate Action and Ministry of Education and Research, has set the stage for AAS adoption [PI23].

The fundamental concept behind AASs is to function as digital repositories, encapsulating all the data associated with the physical assets they represent. This is accomplished through a hierarchical structure known as "submodels", which serve as the internal organisational system for AASs. Each submodel comprises various submodel elements, such as the asset's product name, manufacturer details, or dynamic operational parameters. The composition of elements within a submodel is specific to the submodel in question. While there are no rigid rules governing the structure of a submodel or the submodels an AASs must encompass, best practices often involve utilising the templates provided by the "Details of the Asset Administration Shell Part 1"[Ba22] or those offered by the Industrial Digital Twin Association [IT23] on their GitHub page.

This constitutes the core essence of an AAS; however, discussions surrounding AASs often imply the existence of an AAS runtime environment or server.

2.1 Different types of AASs

AASs can be split into three different groups, as described in [Sc22]:

- Type 1 (Passive AAS): AAS and submodels are serialised files
- Type 2 (Reactive AAS): AAS is a runtime instance accessible via a standardised API
- Type 3 (Proactive AAS): AAS has the ability to actively negotiate with other AASs

It is important to note that each successive type encompasses the functionalities of the preceding types. This research project primarily concerns itself with the generation of basic reactive AASs, necessitating serialised AASs and supplementary runtime instances to facilitate this process.

It's worth noting that there is still limited research in the domain of fully automated generation and deployment of reactive AASs. This area warrants further exploration and investigation.

3 Related Work

Asset Administration Shells (AASs) have witnessed a surge in research interest in recent years. These research efforts span various facets of AAS functionalities, with a substantial number of researchers directing their focus towards Proactive AASs and the intricate digital infrastructures essential for enabling such use-cases [SLl22], [Ye21].

A noteworthy trend is the widespread adoption of the AASX Package Explorer [AP23] as a tool for creating AASs in research projects. This tool has found application in multiple publications such as those by Pribiš et al., Sakurada et al. and Ocker et al. [PBD21], [SLl22], [Oc21]. It serves as a practical solution when the scope of a research project involves the creation of a limited number of AASs. However, this approach proves less suitable for industrial applications, where the demand for AAS generation is substantial. An ideal solution lies in the automated generation of the requisite AASs. Nonetheless, this task presents significant complexity beyond initial expectations.

Initial proposals have surfaced, such as the concept suggested by Miny et al., which explores automated AAS creation through domain-specific language elements [Mi20]. Additionally, Xia et al. have developed a method for generating AASs using neural language models and semantics [XJW22]. This approach, leveraging neural language models, demonstrates promising initial results, although it can encounter challenges related to data mappings, it exhibits versatility beyond specific communication protocols. Moreover, research has been conducted on the conversion of AutomationML to AAS serialisation, as outlined by Lüder et al. [Lü20]. Nevertheless, this approach necessitates the presence of asset information in an AutomationML format before serialisation, rendering it less suitable for a "plug and produce" scenario.

4 Methods

Our research highlighted a practical challenge regarding AASs in industrial adoption. Manual AAS creation using tools like the AASX Package Explorer, while valuable for beginners and understanding the AAS structure, is not feasible for manufacturers or engineers integrating AASs into production lines. The required technical knowledge is substantial.

To address this challenge, we initiated the development of an automated AAS creation tool designed specifically for IO-Link sensors. This tool simplifies the AAS creation process and ensures continuous updates of dynamic sensor information within the AAS. The following sections outline the methods employed to achieve these objectives.

5 Architecture Overview

An overview of the components and their connections within this architecture is depicted in Fig. 1. The architecture employed comprises distinct modules, each being essential to facilitate the automatic AAS generation. The process begins with the generation of a passive AAS, followed by the transmission of operational data to the newly deployed AAS runtime instance.

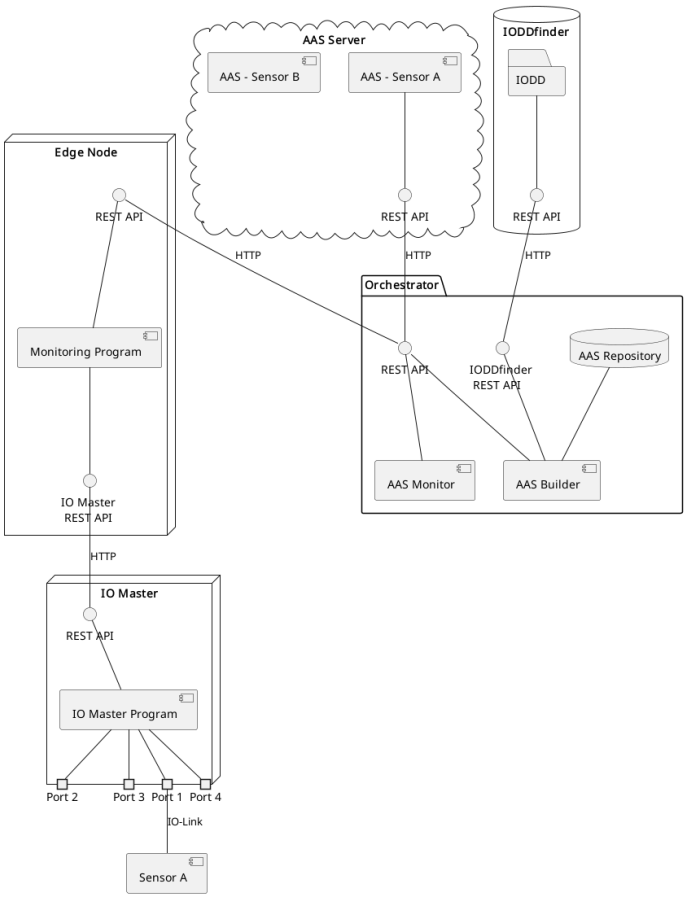

Fig. 1: Component Diagram of presented architecture

5.1 Generation of the passive AAS

The cornerstone of AAS generation in this architecture is the Edge Node, alternatively referred to as an Edge Adapter. Its primary role is to establish the communication protocol between the asset and the broader AAS infrastructure. The specific Edge Node used is determined by the asset's communication protocol. In this project, we designed the Edge Node to work with the IO-Link protocol.

It's important to note that while passive AASs, represented as Type 1, do not require direct communication with the asset, an Edge Node becomes necessary from Type 2 onwards. This holds true unless the sensor itself offers the requisite interface directly to the AAS runtime instance. In our architecture, the Edge Node's significance extends to the generation of passive AASs. This is accomplished by harnessing the information provided by IO-Link's I/O Device Descriptions (IODDs). In the initial phase of the program, the first task is to acquire information about the status of the IO-Master's ports. Once the Edge Node determines which ports are in use by responsive devices, it proceeds to conduct individual polls on these devices. The polled information includes details such as the device manufacturer, product ID, serial number, hardware revision, and firmware revision. This data, in conjunction with the IODD, gathered by the orchestrator, is essential for the creation of an AAS Nameplate submodel. This communication can also be seen in Fig. 2.

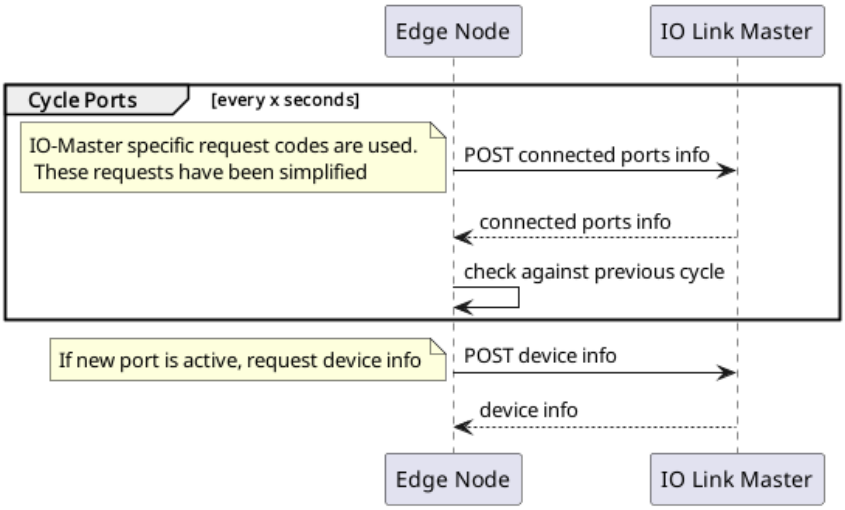

Fig. 2: Communication with IO-Link master

Following the collection of device data, this information is transmitted to the orchestrator via its REST API.

While this approach is tailored for IO-Link, it's essential to highlight its adaptability. The Edge Node used here, specifically designed for IO-Link data transmission, can be substituted with an Edge Node suited to alternative communication protocols for transmitting the asset's operational data. However, it's crucial to note that the creation of a passive AAS for the asset requires the provision of a device profile. Proceeding to the next step, the orchestrator initiates a request for the IODD files from IODDfinder, an online IODD database run by the IO-Link community. Subsequently, the orchestrator cross-references the information obtained from the Edge Node with its locally stored lookup table of known assets. Fig. 3 shows this communication.

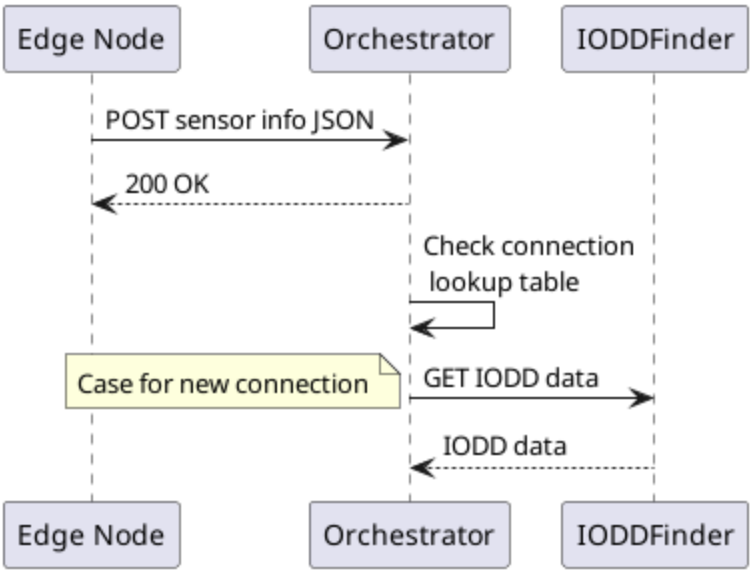

Fig. 3: Retrieval of IODD

If the device has previously undergone AAS generation, and its nameplate information is already available in the orchestrator's records, there is no need to retrieve this information from the IODD. In such cases, the orchestrator launches an AAS runtime instance (AASX Server docker image [AS23]) using the corresponding AASX file stored in its repository. However, if the orchestrator does not find an entry for the asset in question, it proceeds to request the necessary IODD files via the IO-Link IODDfinder API. Subsequently, the orchestrator parses these IODD files to extract the required information. This step can be seen in Fig. 4.

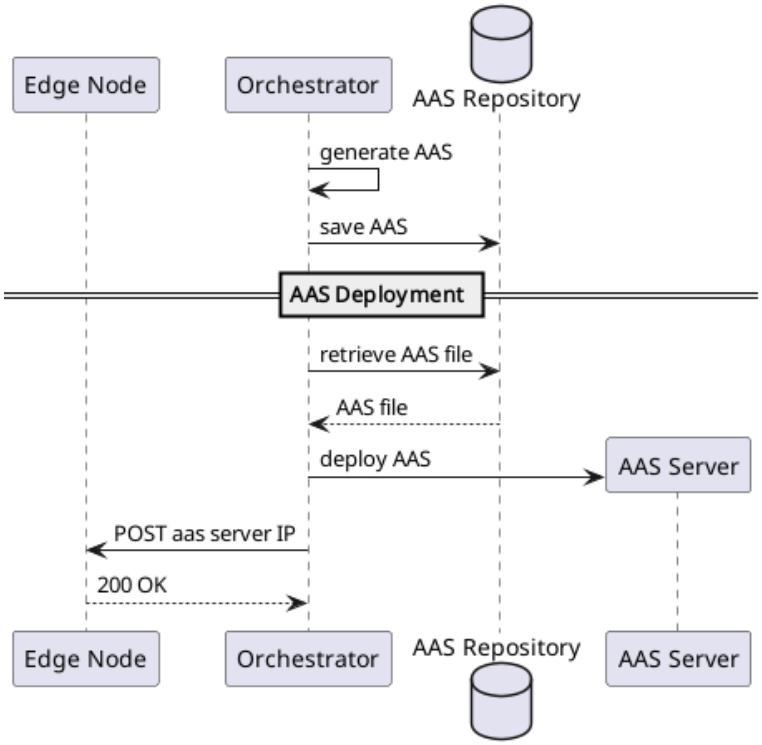

Fig. 4: Generation and deployment of AAS Server

For a comprehensive list of submodel elements used in the Nameplate submodel, along with their corresponding IODD variables or sensor parameters, please refer to Tab. 1.

AAS Nameplate submodel element	IO-Link[a]	
idShort	IODD attribute	Index Service Data Unit[b]
URIOfTheProduct	-	Index 21, Subindex 0 (Serial No.) & Index 19, Subindex 0 (Product Id)
ManufacturerName	DeviceIdentity •vendorName	Index 16, Subindex 0
ManufacturerProductDescription	DeviceVariantCollection •DeviceVariant •Description	-
ContactInformation	-	-
ManufacturerProductRoot	DeviceVariantCollection •DeviceVariant •Description	-
ManufacturerProductFamily	DeviceIdentity •DeviceFamily	-
ManufacturerProductType	DeviceIdentity •DeviceName	Index 19, Subindex 0
OrderCodeOfManufacturer	-	-
ProductArticleNumberOfManufacturer	DeviceIdentity •DeviceName	Index 19, Subindex 0
SerialNumber	-	Index 21, Subindex 0
YearOfConstruction	-	-
DateOfManufacture	-	-
HardwareRevision	-	Index 22, Subindex 0
FirmwareRevison	-	Index 23, Subindex 0
SoftwareVersion	-	-
CountryOfOrigin	-	-
CompanyLogo	DeviceIdentity •VendorLogo	-
Markings	-	-
AssetSpecificProperties	DeviceFunction	-

[a] For IODD specification v1.1, other IODD specifications are also viable
[b] Can be dynamically retrieved from the IO-Link device itself

Tab. 1: IO-Link to AAS mapping (“-” stands for no information available)

5.2 Transmission of operational data

Once the orchestrator has assembled the requisite information, generated the AAS, and deployed a runtime instance, it proceeds to notify the asset's Edge Node about the IP address of the AAS runtime. This information enables the asset's Edge Node to establish direct communication with the AAS runtime's API.

This step is pivotal as it facilitates seamless communication between the Edge Node and the AAS, bypassing the orchestrator. In larger systems, this approach prevents the orchestrator from becoming a potential bottleneck. Subsequently, the Edge Node periodically updates the operational data from the asset via the IO-Master and directly transmits this data to the AAS runtime instance using its REST API. This communication cycle can be seen in Fig. 5.

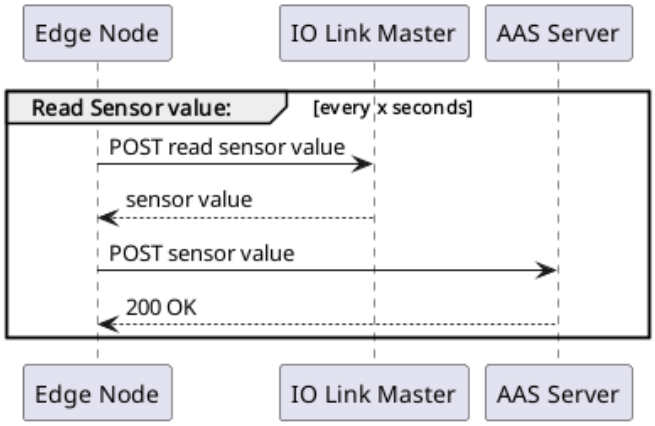

Fig. 5: Dynamic asset data update cycle

6 Validation

To evaluate the effectiveness of the proof of concept, we conducted tests on a sensor testbed equipped with multiple IO-Link sensors. Fig. 6 shows the testbed used. The testbed served as a comprehensive platform for assessing the capabilities of the presented architecture with a diverse array of sensors. Our proposed architecture demonstrated its ability to seamlessly communicate with all sensors within the testbed. Moreover, it successfully generated AASs with Nameplate submodels for these sensors. For a visual representation of one such AAS, please refer to Fig. 7

Fig. 6: Sensor Testbed

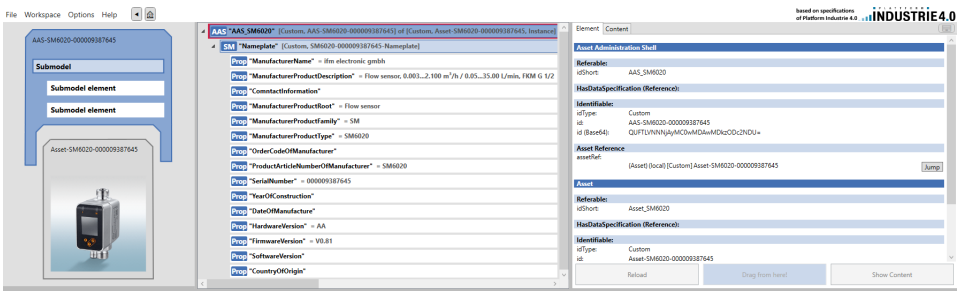

Fig. 7: Generated AAS from the Testbed opened in the AASX Package Explorer

7 Discussion & Outlook

The architecture presented in this study exhibits promise for automating the creation of digital nameplates within AASs. This capability could prove increasingly valuable in the context of potential future requirements for digital product passports (DPPs) [EU23], which may become mandatory for manufacturers in the future. Initiatives are already underway to establish AASs as the de-facto standard for DPPs [DP23], and manual AAS generation is unlikely to remain a feasible option. However, it's crucial to acknowledge that this architecture has certain limitations. One notable limitation is the reliance of edge nodes on existing device descriptions for assets, which are subsequently reformatted into a compatible AAS structure. As detailed in Tab. 1, there are instances where certain AAS nameplate submodel elements lack equivalents within the IODD or ISDU. This limitation is likely to extend to other communication protocols not specifically designed with AASs in mind. Nonetheless, this architecture excels in delivering a true "plug and produce" functionality, demonstrated by its capacity to seamlessly generate basic reactive AASs.

Bibliography

[AP23] Industrial Digital Twin Association e. V., 2023, URL: https://github.com/admin-shell-io/aasx-package-explorer.

[AS23] Industrial Digital Twin Association e. V., 2023, URL: https://github.com/admin-shell-io/aasx-server.

[Ba22] Bader, S.; Barnstedt, E.; Bedenbender, H.; Berres, B.; Billmann, M.; Boss, B.; Braunisch, N.; Braunmandl, A.; Clauer, E.; Diedrich, C.; Flubacher, B.; Fritsche, W.; Garrels, K.; Gatterburg, A.; Hankel, M.; Heppner, S.; Hoffmeister, M.; Jänicke, L.; Jochem, M.; Ziesche, C.: Details of the Asset Administration Shell. Part 1 -The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC02), Mai 2022.

[DP23] Industrial Digital Twin Association e. V., ZVEI e. V., 2023, URL: https://dpp40.eu/.

[EU23] European Health and Digital Executive Agency (HaDEA), 2023, URL: https://hadea.ec.europa.eu/calls-proposals/digital-product-passport_en.

[Gr20] Grabowska, S.: Smart Factories in the Age of Industry 4.0. Management Systems in Production Engineering 28/2, S. 90–96, 2020, URL: https://doi.org/10.2478/mspe-2020-0014.

[IT23] Industrial Digital Twin Association e. V., 2023, URL: https://industrialdigitaltwin.org/.

[Kr18] Kritzinger, W.; Karner, M.; Traar, G.; Henjes, J.; Sihn, W.: Digital Twin in manufacturing: A categorical literature review and classification. IFAC-PapersOnLine 51/11, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018, S. 1016–1022, 2018, ISSN: 2405-8963, URL: https://www.sciencedirect.com/science/article/pii/S2405896318316021.

[Lü20] Lüder, A.; Behnert, A.-K.; Rinker, F.; Biffl, S.: Generating Industry 4.0 Asset Administration Shells with Data from Engineering Data Logistics. In: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). Bd. 1, S. 867–874, 2020.

[Mi20] Miny, T.; Thies, M.; Epple, U.; Diedrich, C.: Model Transformation for Asset Administration Shells. In. S. 2207–2212, Okt. 2020.

[Oc21] Ocker, F.; Urban, C.; Vogel-Heuser, B.; Diedrich, C.: Leveraging the Asset Administration Shell for Agent-Based Production Systems. IFAC-PapersOnLine 54/1, 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021, S. 837–844, 2021, ISSN: 2405-8963, URL: https://www.sciencedirect.com/science/article/pii/S2405896321009563.

[PBD21] Pribiš, R.; Beňo, L.; Drahoš, P.: Asset Administration Shell Design Methodology Using Embedded OPC Unified Architecture Server. Electronics 10/20, 2021, ISSN: 2079-9292, URL: https://www.mdpi.com/2079-9292/10/20/2520.

[PI23] Plattform Industrie 4.0, 2023, URL: https://www.plattform-i40.de.

[Re17] Ren, G.; Hua, Q.; Deng, P.; Yang, C.; Zhang, J.: A Multi-Perspective Method for Analysis of Cooperative Behaviors Among Industrial Devices of Smart Factory. IEEE Access 5/, S. 10882–10891, 2017.

[Sc22] Schnicke, F.; Kuhn, T.; Klausmann, T.; Grüner, S.; Porta, D.: Architecture Blueprints for the Application of the Industry 4.0 Asset Administration Shell. In. S. 1–8, 2022.

[SLl22] Sakurada, L.; Leitao, P.; la Prieta, F. D.: Agent-Based Asset Administration Shell Approach for Digitizing Industrial Assets. IFAC-PapersOnLine 55/2, 14th IFAC Workshop on Intelligent Manufacturing Systems IMS 2022, S. 193–198, 2022, ISSN: 2405-8963, URL: https://www.sciencedirect.com/science/article/pii/S2405896322001938.

[XJW22] Xia, Y.; Jazdi, N.; Weyrich, M.: Automated generation of Asset Administration Shell: a transfer learning approach with neural language model and semantic fingerprints. In: 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA). S. 1–4, 2022.

[Ye21] Ye, X.; Hong, S. H.; Song, W. S.; Kim, Y. C.; Zhang, X.: An Industry 4.0 Asset Administration Shell-Enabled Digital Solution for Robot-Based Manufacturing Systems. IEEE Access 9/, S. 154448–154459, 2021.