Overcoming Challenges in Integrating Legacy Devices with Asset Administration Shells - An OPC UA Case Study

Aaron Zielstorff, Dirk Schöttke,
Antonius Hohenhövel, Thomas Kämpfe, Stephan Schäfer
Hochschule für Technik und Wirtschaft (HTW) Berlin
Berlin, Germany
Email: stephan.schaefer@htw-berlin.de

Frank Schnicke
Fraunhofer IESE
Kaiserslautern, Germany
Email: frank.schnicke@iese.fraunhofer.de

2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA) | 979-8-3503-3991-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ETFA54631.2023.10275536

***Abstract*—The transition towards Industry 4.0 requires the modernisation of legacy systems. However, this transformation introduces complexity, mainly due to the variety of data formats and interfaces resulting from the heterogeneity of the used components. For seamless interoperability in Industry 4.0 applications, a harmonised data base is of vital importance. The Asset Administration Shell (AAS), as a standardised digital twin of assets, plays a key role in facilitating interoperable, data-centric solutions in the Industry 4.0 landscape. The prospect of automated data integration offers the potential to reduce errors and optimise the process of digitising legacy systems. This raises the question of what extensions and components within the AAS infrastructure are necessary to realise such automation. In response, this paper presents an architectural concept for integrating legacy systems into the AAS framework. Using an articulated robot as a tangible example, the process of interconnecting data points via the OPC UA protocol is illustrated. Additionally, a prototype is presented, capable of enabling vertical data integration via the BaSyx DataBridge, thus showcasing the notable advantages of automating the incorporation of legacy devices into the AAS. The shown solution retains flexibility and is readily applicable to a variety of systems and scenarios as required.**

***Index Terms*—Asset Administration Shell, Data Integration, OPC UA, Industry 4.0, DataBridge**

I. Introduction

As digitalization advances, the efficient utilization of data has become the focus of many industrial strategies. The capability to collect, analyse, and transform data into qualified information has emerged as a pivotal factor [1]. However, despite rapid advancements in information technology, enterprises in automation face significant challenges when integrating data from their legacy systems. This is due to the software solutions employed, which exert substantial influence on the flexibility and complexity of the system. Specifically, system interoperability and scalability present substantial challenges [2].

Based on the concept of the digital twin, the Asset Administration Shell (AAS) specification, developed by Platform Industry 4.0, creates a common understanding and guarantees interoperability through a defined metamodel [3]. The information provided by the AAS includes, among other things, documents, properties, and parameters [4].

In the industrial environment, where a variety of control components and communication protocols are utilised, the need to adapt the information provided for a unified understanding is crucial. This adaptation can be largely compensated for by using AASs and software adapters to exchange information with the assets. The use of software adapters to connect new and existing plants to the AAS is necessary because there is currently no direct integration of AASs on established industrial controls [5]. Consequently, in addition to the configuration of the controller, a configuration of the software adapters used is required. The open-source tool BaSyx DataBridge, discussed in this contribution, functions as a software adapter. The DataBridge enables connectivity to the AAS via various protocols and bus systems [6]. Depending on the use case, the extent of effort necessary in data provision and testing can vary significantly. This is largely due to manual setup requirements and the potential for overlooked mistakes. Currently, the project planning phase lacks necessary support, causing an increased complexity when configuring broad-scale solutions or adapting to altered circumstances. The research question of this paper is therefore: How can legacy devices be integrated into the AAS in a user-friendly and time-efficient manner?

To address this, an architectural concept is presented that focuses on facilitating the efficient provisioning of configuration data to the DataBridge while leveraging the existing AAS infrastructure. To illustrate this, an articulated robot use case is shown to demonstrate the transmission of data to the AAS using OPC UA.

The paper is structured as follows: Section II provides an overview of the benefits of using Digital Twins in the context of Industry 4.0 scenarios and presents existing approaches to data integration into the AAS. Section III discusses the current state of data integration for AASs, using the Eclipse BaSyx Middleware as an example. To illustrate the relevance of an extension concept, Section IV introduces an articulated robot as an OPC UA case study. Based on the insights gained, Section V derives requirements for an architectural extension for data integration. Section VI presents potential infrastructure components for a holistic approach to data integration. Building on this concept, Section VII introduces a prototype that demonstrates the partial implementation of the presented architecture focusing on the automated configuration of data provision. In Section VIII, the prototype is evaluated utilizing the defined use case scenario. A forecast on future extensions for the full realization of the concept is provided.

II. State of the Art

In the realm of manufacturing automation, a fundamental distinction is made between the domains of Information Technology (IT) and Operational Technology (OT). The OT encapsulates pertinent hardware and software components that enable the monitoring of devices, processes, and events. Industrial control systems, equipped with control programs following IEC 61131-3 standards, belong to this domain and exhibit a close correlation with the operating resources of production facilities and systems [5]. This provides significant advantages in the description, commissioning, and reconfiguration of plant components.

Nevertheless, despite these connections, there are also deficits in the use of industrial controls [7]. On one hand, there is a lack of consistent self-description, maintenance information, and representation of relationships to neighbouring systems and components. On the other hand, there are discontinuities in the consistent preparation, storage, and use of information during the engineering phases. When dealing with industrial controllers (especially legacy systems), comprehensive information provision is usually only possible via software adapters, connectors and gateways. For this, the communication protocol used and the data models of the partners must be known. However, if resources such as sensors only have an interface to the IT environment, which is not supported by the industrial control, the direct use of the relevant information on the part of the industrial control is not possible.

In Industry 4.0 environments, cross-sectional use of data from both IT and OT domains aspires. Peer-to-peer communication between installed devices is desirable [8]. As a solution, AASs provide an essential foundation for the provision of data across device boundaries and enable the integration of data from multiple source systems. For this, a mapping of the respective endpoints to the properties of the AAS is necessary. There are already approaches for creating and using AASs to describe assets.

In the paper [9], the authors introduced an environment for the generic preparation of AASs, which was implemented within the scope of the "OpenBasys" project. In this environment, Type 2 AASs (reactive AAS) can be realized without programming knowledge. The AAS is largely automatically generated by the operator or on request, for example, by process control, and stored in a container. The implementation is based on the reconstruction of the OPC UA data structures on the part of the assets.

By using the established AASX Package Explorer, it is also possible to prepare active AASs with their structures and embedding of relevant communication relationships. This is largely done manually, with knowledge of the data structures and formats [10].

The extension of the AAS Manager [11] provides another opportunity for the automated preparation of AASs. The authors, among other things, illustrate the process of transferring the source model (here CSV file) into an AAS-specific target model using a transformation system, consisting of a transformation interpreter and definer. The configuration by the user takes place via a user interface.

With the AAS DataBridge, it is possible to configure communication relationships between assets and existing AASs offline. It supports a variety of protocols and can be manually adjusted via configuration files. The DataBridge also provides a multitude of options for the integration of data from legacy devices.

III. Status Quo of AAS Data Integration

This section explains the Eclipse Basyx DataBridge as a central component for connecting assets. First, a brief introduction to the AAS concept is given. This is followed by a description of the architecture blueprints relevant to the DataBridge, which enable the vertical integration of assets into the AAS. The configuration of the DataBridge is then explained.

A. Asset Administration Shell

An AAS is a standardised digital representation of a physical or logical object, the so-called asset [12]. For this reason, the AAS can also be described as a digital model of the asset, which, based on metadata, serves as a machine-readable self-description of the corresponding physical or logical component. According to [13], the information model of an AAS consists of the following elements (excerpt):

- Metainformation on AASs, assets, Submodels, etc.,

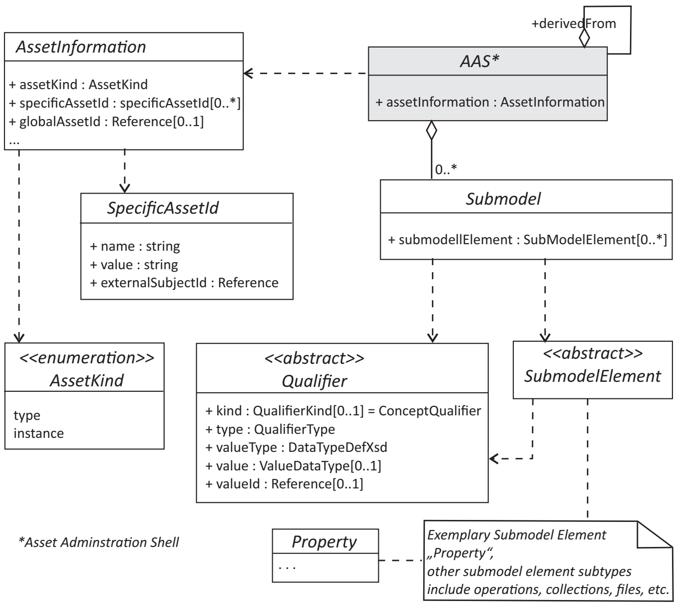

+derivedFrom
AssetInformation
+ assetKind : AssetKind
+ specificAssetId : specificAssetId[0..*]
+ globalAssetId : Reference[0..1]
...
AAS*
+ assetInformation : AssetInformation
0..*
SpecificAssetId
+ name : string
+ value : string
+ externalSubjectId : Reference
Submodel
+ submodellElement : SubModelElement[0..*]
<<enumeration>> AssetKind
type
instance
<<abstract>> Qualifier
+ kind : QualifierKind[0..1] = ConceptQualifier
+ type : QualifierType
+ valueType : DataTypeDefXsd
+ value : ValueDataType[0..1]
+ valueId : Reference[0..1]
<<abstract>> SubmodelElement
*Asset Adminstration Shell
Property
. . .
Exemplary Submodel Element „Property“, other submodel element subtypes include operations, collections, files, etc.

Fig. 1. Excerpt of a class diagram representing the structure of AASs [12].

- Asset-independent and asset-specific Submodels.

Submodels can represent certain aspects of the asset and contain both static and dynamic (or executable) elements. According to [14], an AAS comprises *Submodels* that are used to structure the information and functions of an AAS into distinct parts. Each submodel refers to a clearly defined domain or subject. Submodels can be standardised and thus become submodel types. An AAS can refer to a submodel type, resulting in that submodel instance containing a guaranteed set of properties and functions. The class diagram shown in Figure 1 gives a detailed impression of the structure of the AAS.

In the Industrial Digital Twin Association e. V. (IDTA), extensive efforts are currently being made to specify, test and subsequently standardise submodels for various use cases [15].

B. Architecture Blueprints for Data Integration

When it comes to integrating data into AASs, there are a variety of methods available. The paper [8] presents different scenarios for data provision. The authors distinguish between blueprints based on the frequency of data queries and data generation as well as the amount of data. The following blueprints were illustrated (excerpt):

- Delegation,
- Cyclical Updates,
- Event-driven Integration.

Figure 2 shows the communication relationships for the mentioned blueprints.

C. BaSyx DataBridge

The BaSyx DataBridge provides the two blueprints *Updater* and *Delegator* as core functionalities. It supports various communication protocols such as OPC UA, MQTT, Kafka, and various fieldbus systems via PLC4X [6]. It is provided as an off-the-shelf component as part of the Eclipse BaSyx middleware and can be tailored for specific use cases through configuration files without any programming effort.

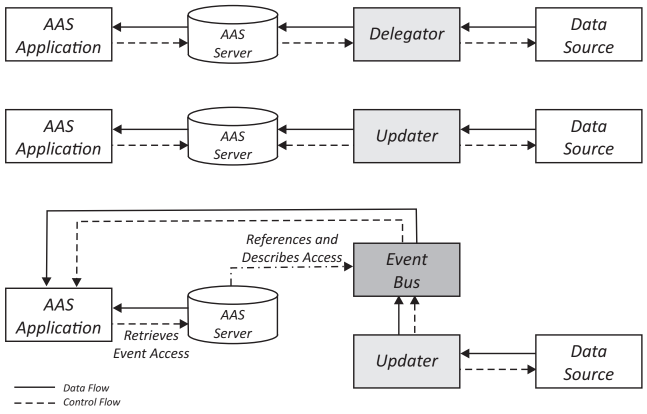


Fig. 2. Blueprints for data integration (from top to bottom: Delegator, Updater, event integration) [8].

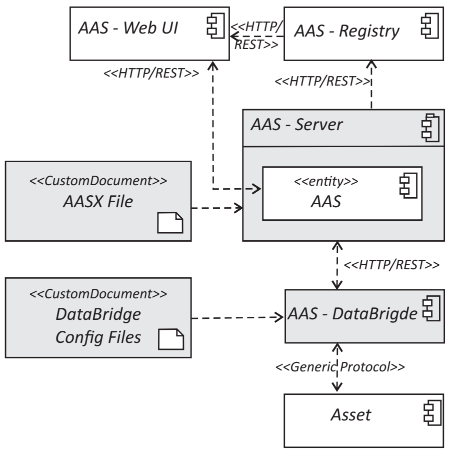


Fig. 3. Components of the BaSyx Infrastructure.

Live data integration requires a data source, in this case, one or more assets, and an active (Type 2 or Type 3) AAS in a runtime environment (BaSyx middleware). It is recommended to use additional compatible components from the BaSyx ecosystem. Type 2 AASs can be created and hosted by uploading previously modeled Type 1 AAS to the AAS server. The AAS Server automatically registers the endpoints of generated AASs in the AAS Registry. AASs referenced there can be visualised via the AAS Web UI. The described architecture component structure is illustrated in Figure 3.

Furthermore, the following prerequisites are provided for creating configuration files and commissioning the DataBridge:

- Knowledge about the communication protocols of the assets,
- Knowledge about the data structure of the assets,
- Accessibility of the assets over the network,
- Knowledge about the AAS structure,
- Accessibility of the AAS server over the network.

Once all prerequisites are met, the DataBridge can be commissioned by integrating specific configuration files. In general, the following parameters must be configured:

- Access information for the data source,
- Data transformation,
- Target submodel and target property of the submodel,
- Overall structure of the integration route.

This paper focuses on the configuration of the DataBridge for OPC UA[1].

IV. OPC UA Case Study

For a more detailed examination of the research question, a use case from the Industry 4.0 domain is considered. It concerns a handling system used in the automotive industry for the assembly of car parts. It consists of an articulated robot that is attached to the ceiling of the production hall via a joint.

[1]https://github.com/eclipse-basyx/basyx-applications/tree/main/opc2aas/DatabridgeDemo/DatabridgeConfig

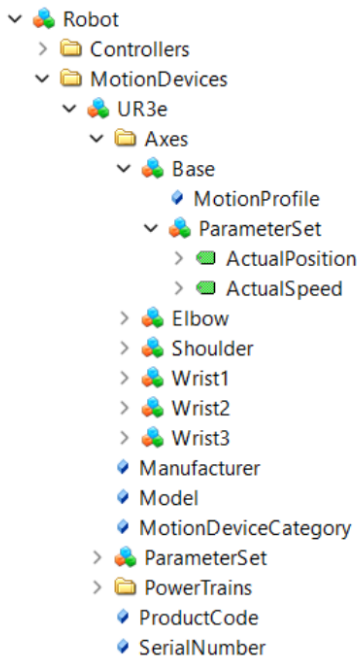
The industrial robot is equipped with a gripper that picks up vehicle parts and transports them to various assembly stations. Information about the handling system, such as the robot's joint positions, can be queried via a dedicated OPC UA server.

The data structure on the OPC UA server is based on the OPC Foundation's Robotics Nodeset. The nodeset defines a uniform structure for describing motion device systems and their components. The individual joints of the robot are defined as instances of *AxisTypes*. These contain not only static properties, such as the motion profile but also dynamic values. This includes joint positions and velocities obtained from the field level. An exemplary representation of the structure on the OPC UA server is shown in Figure 4. The server consists of a total of 88 nodes and includes string, float, integer, and boolean variables as well as complex data types.

To create an AAS for the handling system, several submodels are used whose modeling is based on the VDMA AAS for robots. The data model is built on the OPC UA Companion Specification 40010-1 [16]. In addition to the standardised submodels for asset identification (e.g. digital nameplate) and topology overview (bill of materials), the resulting AAS also includes non-standardised submodels to describe the kinematic structure.

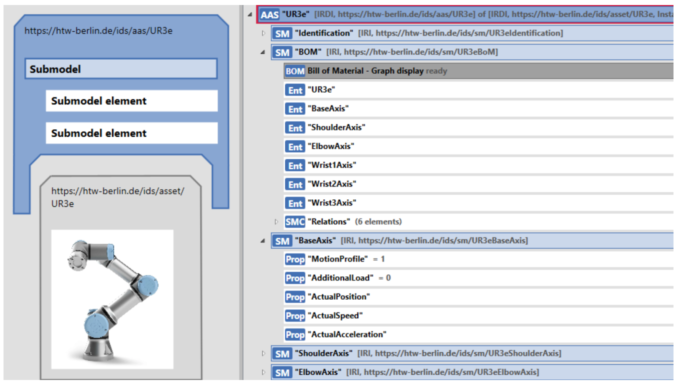
Figure 5 shows a section of the AAS in the AASX Package Explorer. The Bill of Materials (BOM) submodel and an Identification submodel can be seen. Dynamic joint positions are stored in non-standardised, dynamically filled submodels. These can be found at the bottom of the image.

The BaSyx DataBridge allows the integration of data from the OPC UA server into the AAS. As explained in Section III, the corresponding configuration files are populated for



```
Robot
  Controllers
  MotionDevices
    UR3e
      Axes
        Base
          MotionProfile
          ParameterSet
            ActualPosition
            ActualSpeed
        Elbow
        Shoulder
        Wrist1
        Wrist2
        Wrist3
      Manufacturer
      Model
      MotionDeviceCategory
      ParameterSet
      PowerTrains
      ProductCode
      SerialNumber
```

Fig. 4. Tree representation of the OPC UA Robotics Nodeset for an industrial robot.



https://htw-berlin.de/ids/aas/UR3e
Submodel
Submodel element
Submodel element
https://htw-berlin.de/ids/asset/UR3e

```
AAS "UR3e" [IRDI, https://htw-berlin.de/ids/aas/UR3e] of [IRDI, https://htw-berlin.de/ids/asset/UR3e, Insta
  SM "Identification" [IRI, https://htw-berlin.de/ids/sm/UR3eIdentification]
  SM "BOM" [IRI, https://htw-berlin.de/ids/sm/UR3eBoM]
    BOM Bill of Material - Graph display ready
    Ent "UR3e"
    Ent "BaseAxis"
    Ent "ShoulderAxis"
    Ent "ElbowAxis"
    Ent "Wrist1Axis"
    Ent "Wrist2Axis"
    Ent "Wrist3Axis"
    SMC "Relations" (6 elements)
  SM "BaseAxis" [IRI, https://htw-berlin.de/ids/sm/UR3eBaseAxis]
    Prop "MotionProfile" = 1
    Prop "AdditionalLoad" = 0
    Prop "ActualPosition"
    Prop "ActualSpeed"
    Prop "ActualAcceleration"
  SM "ShoulderAxis" [IRI, https://htw-berlin.de/ids/sm/UR3eShoulderAxis]
  SM "ElbowAxis" [IRI, https://htw-berlin.de/ids/sm/UR3eElbowAxis]
```

Fig. 5. AAS of an articulated robot in the AASX Package Explorer.

each route from the asset (here robot with OPC UA interface) to the AAS. To put this into practice, this means that a total of 264 JSON objects have to be created and mapped manually to commission the DataBridge. This results in a considerable amount of effort in the configuration and testing of data provision. In particular, this is due to the lack of support for creating configuration files and identifying potential errors.

To illustrate this, an automation engineer was tasked to configure the connection of the handling system using the Robotics Nodeset via OPC UA in a non-repeated attempt. Specifically, creating the DataBridge configuration took 3.5 hours. The corresponding submodel in the AAS was set up within a further 2 hours. Two errors occurred during the configuration, where a wrong AAS endpoint and NodeID were configured. These took another 1.5 hours to locate and fix. The total time required to integrate the robot into the AAS was therefore 7 hours.

The following obstacles for productive use were identified from the exemplary setup of the DataBridge:

- An AAS description must already exist,
- The manual configuration of the DataBridge is complex,
- There is no verification of the configured endpoints,
- There is also no verification of the AAS structure and content.

Thus, the DataBridge provides a purely technical solution for connecting assets to the AAS. Any adjustments on the part of the asset or the AAS require manual modification of the configuration files.

V. Requirements for a User-oriented Adaption

The challenges identified in the operationalisation of data integration via the DataBridge lead to the following requirements for architectural enhancements in the AAS environment for user-friendly configuration of data provisioning:

1) **Utilization of Existing Resources:** When implementing new systems and scenarios, existing and available assets should be suggested for deployment. This approach offers economic benefits.
2) **Automated Generation of AASs:** The creation of AASs should be automated. This particularly includes the

automatic generation of AASs from the configuration of data provisioning and the selection of relevant submodels. The selection is based on information from requirements engineering, vendor information about the asset and the views of relevant stakeholders on the data in the resulting AAS.

3) **Assisted Submodel Provisioning:** The user should be supported in the selection of suitable submodels for the use case specified by them. Also relevant is the automatic assignment of asset data points to technology-related submodel data models.
4) **Automated Configuration Tests:** Especially in manual configuration, conducting configuration tests to identify potential error sources is necessary. These tests should be performed automatically. A critical aspect is checking the accessibility of asset and AAS endpoints.
5) **Error Reporting and Handling:** In addition to automated testing of communication routes, error reporting and their correction must also be possible. This functionality is particularly relevant for extensive data models.
6) **User-friendly Configuration Interface:** The configuration of data provisioning should be as simple and intuitive as possible for the user. This includes the graphical preparation of the configuration interface and user support through assistant functions.

VI. Concept for a dynamic Asset Integration Architecture

In this section, an extension concept for the already existing BaSyx AAS infrastructure is presented. By integrating five new architectural components, this concept meets the identified requirements for the efficient commissioning of data provision from existing systems or generally from dynamic assets.

The new components are:

- Asset Catalogue,
- Submodel Template Database,
- Submodel Template Integrator,
- Data Conditioning Service,
- Configuration User Interface.

A complementary component diagram, which shows both the existing infrastructure and the new components shaded in gray, is presented in Figure 6. Figure 7 displays a sequence diagram that visualises the configuration workflow for asset integration. Upon completion of the configuration, the setup is made available for productive operation (see Figure 8). The new components and their functionalities are described in the following subsections.

A. Asset Catalogue

The Asset Catalogue aims to fulfill requirement 1 from Section V. It acts as a management tool for already existing assets and associated information. These include the communication interfaces of the respective asset, associated submodels (e.g., via the *SemanticID*), and tags for asset classification to facilitate search. Classifiers are for example the asset type, the manufacturer, or provided skills represented by already existing submodels, AASs and AutomationML files (e.g. from asset vendors). Advantages of using this component arise particularly from sustainability through the reuse of assets and from the reduction of configuration effort resulting from the presence of asset-specific metadata.

B. Submodel Template Database

The Submodel Template Database is responsible for managing submodel templates, aligning with requirement 3. Here, a distinction is made between two types of submodels: Intra-Company-Submodels, which are specifically developed for assets within a company (e.g., a specific industrial robot), and

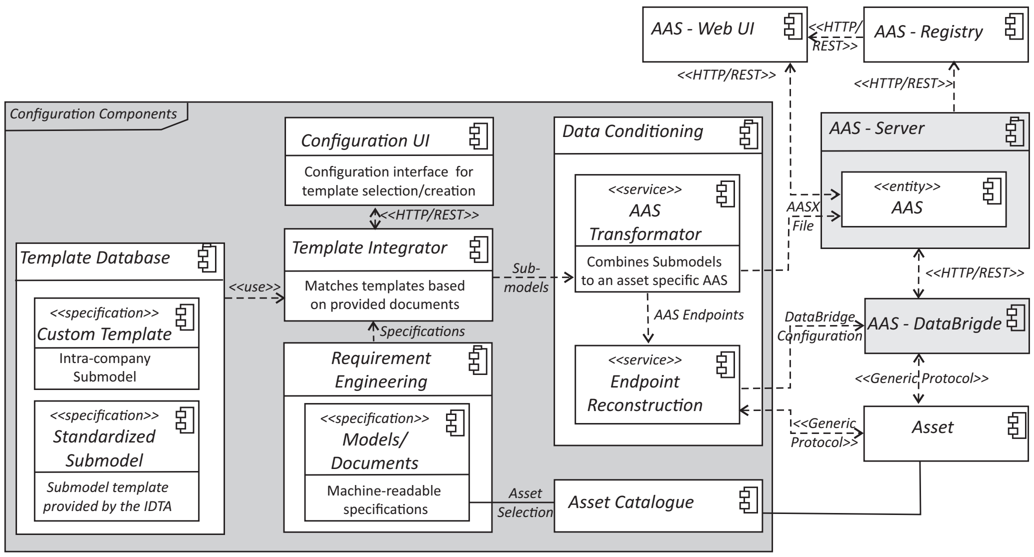


Fig. 6. Component Diagram of the Asset Integration Architecture using existing BaSyx Components.

standardized submodels (e.g., Digital Nameplate), provided by the IDTA.

C. Submodel Template Integrator

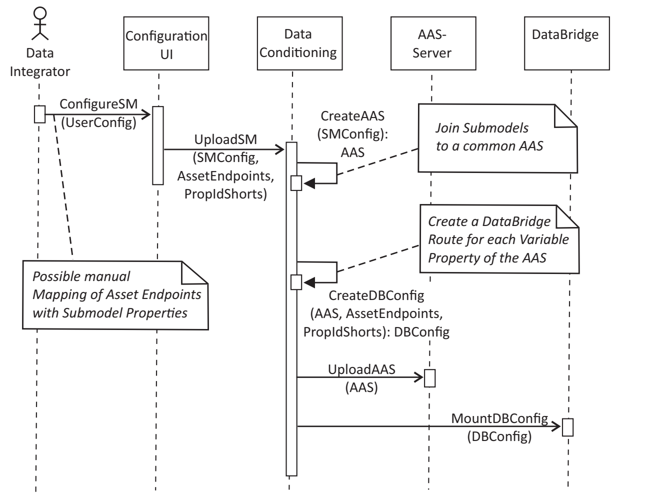
The Submodel Template Integrator enables the generation of submodels from the templates provided by the Submodel Template Database. This is achieved by matching submodel templates with asset information from the Asset Catalogue or with the specific data from the asset. An assistant service can guide the selection of submodels with the help of tags, semantic categorisations, and interface descriptions (see requirement 3). If no suitable submodel is available, there is the option to create a new one and store it in the Submodel Template Database for future uses. Furthermore, the Submodel Template Integrator acts as a backend for the Configuration User Interface. It provides information about both the existing submodel templates and the selected assets.

D. Data Conditioning Service

The Data Conditioning Service is responsible for assembling submodels to generate AASs (see requirement 2). Additionally, it is in charge of creating configuration data for the DataBridge. This includes constructing the endpoint description from the data structure of the submodels according to the communication protocol used, as well as generating descriptors for automated testing of the DataBridge's communication routes (see requirement 4).

E. Configuration User Interface

The Configuration User Interface provides a user-friendly configuration interface for data provision from existing systems, aiming to fulfill requirement 6. This User Interface is the front-end application for the Submodel Template Integrator, assisting the user in the selection of submodels, the configuration of communication routes, and the assignment of asset data points to submodel properties. Moreover, it is capable of displaying errors in the configuration, for instance, after the automated testing of communication routes via the generated descriptors (see requirement 5).



Data Integrator
Configuration UI
Data Conditioning
AAS-Server
DataBridge
ConfigureSM (UserConfig)
UploadSM (SMConfig, AssetEndpoints, PropIdShorts)
CreateAAS (SMConfig): AAS
Join Submodels to a common AAS
Create a DataBridge Route for each Variable Property of the AAS
Possible manual Mapping of Asset Endpoints with Submodel Properties
CreateDBConfig (AAS, AssetEndpoints, PropIdShorts): DBConfig
UploadAAS (AAS)
MountDBConfig (DBConfig)

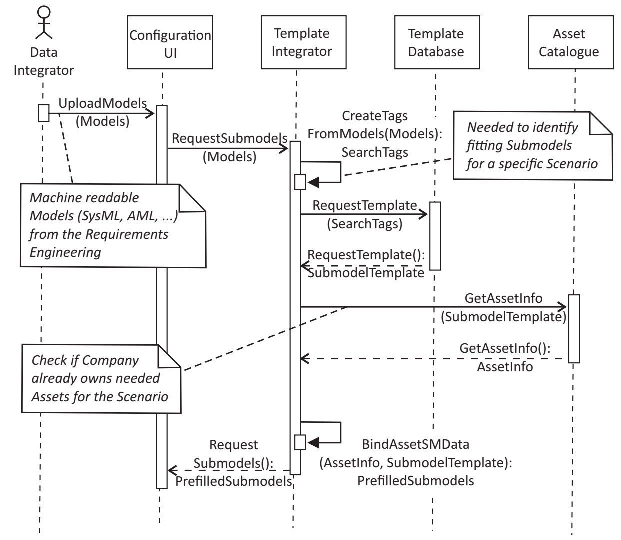
Fig. 8. Sequence Diagram of the AAS deployment based on configured Submodels (Excerpt).

VII. Prototype for Asset Integration

Based on the preceding architectural concept, a prototype is presented in the following section that enables the integration of data from the use case introduced in Section IV. The prototype focuses on the Data Conditioning Service and its use for the OPC UA protocol. Within the context of this contribution, this component is of particular relevance as it facilitates the automatic configuration of asset connection. Therefore, the functionalities of the other four components are not further considered.

Since the presented robot provides data via OPC UA, a data structure already exists on the OPC Server, which can be mapped into an AAS. Thus, it is possible to implement and evaluate the Data Conditioning Service separately. Instead of using standardised and Intra-Company submodels, the OPC UA data structure is used to create AASs. Therefore, the prototype generates the AAS, which represents the same data model as present on the server.

Figure 9 depicts the prototype of the Data Conditioning Service, subsequently referred to as *OPC2AAS*[2], and its integration into the existing BaSyx infrastructure. The OPC2AAS component is itself an AAS that provides its functionality via an Operation-SubmodelElement. The AAS consists of two submodels: The first one includes the operation to start the service, whereas the second one contains automatically created DataBridge configuration files and the generated AAS as File-SubmodelElements.



Data Integrator
Configuration UI
Template Integrator
Template Database
Asset Catalogue
UploadModels (Models)
RequestSubmodels (Models)
CreateTags FromModels(Models): SearchTags
Needed to identify fitting Submodels for a specific Scenario
Machine readable Models (SysML, AML, ...) from the Requirements Engineering
RequestTemplate (SearchTags)
RequestTemplate(): SubmodelTemplate
GetAssetInfo (SubmodelTemplate)
GetAssetInfo(): AssetInfo
Check if Company already owns needed Assets for the Scenario
Request Submodels(): PrefilledSubmodels
BindAssetSMData (AssetInfo, SubmodelTemplate): PrefilledSubmodels

Fig. 7. Sequence Diagram visualizing the Submodel Generation from Templates based on provided Engineering Models (Excerpt).

[2]https://github.com/eclipse-basyx/basyx-applications/tree/main/opc2aas

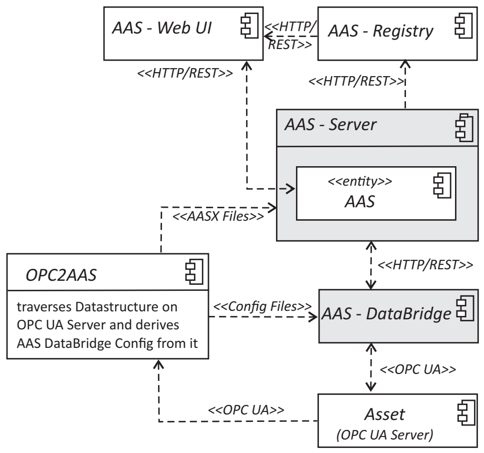


Fig. 9. Component Diagram for the OPC2AAS Prototype.

The OPC2AAS Service is initiated via the operation *CreateAASFromOPC*. This operation accepts as parameters the name of the AAS to be generated and the address of the OPC UA Server. The sequence diagram in 10 illustrates the process of automatic configuration generation.

Initially, the data structure of the OPC UA Server is read. This is followed by a recursive reconstruction of the data model, starting after the root node of the server. The top level of the AAS to be generated is represented through submodels. Deeper levels use SubmodelElementCollections as a collection of subordinate elements. The deepest level of a branch is resolved through properties.

Currently, most of the basic OPC UA data types are supported, as well as complex data types such as structures and enumerations. This allows, but is not limited to, mapping OPC UA companion specifications. Custom structures and data types are also supported. At this time, OPC UA events and methods are not considered for this approach.

For each property, the required JSON objects for the respective Databridge configuration files are created. After all data points of the OPC UA Server have been mapped, all previously created submodels are merged into an AAS. This is then serialized into an AASX file and provided to the data integrator as a File-SubmodelElement. The Databridge configuration files are also created via the serialization of the corresponding configuration objects and are stored as downloadable files.

By uploading the generated AAS to the AAS Server, a live instance of it is initiated. Subsequently, the DataBridge can be put into operation by incorporating the created configuration files.

In the environment of the AAS Web UI[3], it is now possible to examine the AAS along with its dynamic data from the asset (in this case, the industrial robot). For instance, users can view the changing joint positions of the robot.

[3]https://github.com/eclipse-basyx/basyx-applications/tree/main/aas-gui

VIII. Conclusion & Outlook

In this paper, we've underscored the significant role of Digital Twins, with a particular emphasis on Asset Administration Shells (AAS). Integrating data from legacy devices into the AAS surfaced as a notable challenge. In response, we've chosen to leverage the BaSyx DataBridge, a technical solution devised for asset connection. It lends support to an array of communication protocols, including but not limited to OPC UA. However, we've identified two major obstacles in its practical implementation: the lengthy commissioning process and the vulnerability to errors during manual configuration.

To overcome these obstacles, we've proposed an architectural concept that extends the BaSyx AAS infrastructure. Automatic configuration for asset connectivity has been identified as a vital requirement for the new architecture. Additionally, this paper introduces a prototype that implements a segment of the

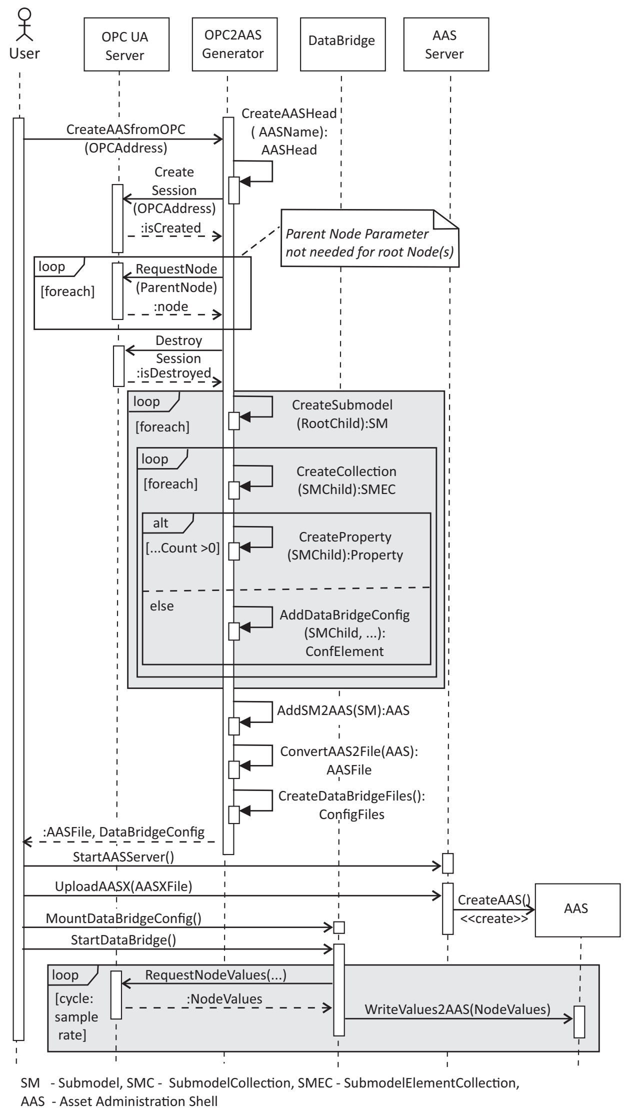

User
OPC UA Server
OPC2AAS Generator
DataBridge
AAS Server
CreateAASfromOPC (OPCAddress)
CreateAASHead (AASName): AASHead
Create Session (OPCAddress)
:isCreated
Parent Node Parameter not needed for root Node(s)
loop
[foreach]
RequestNode (ParentNode)
:node
Destroy Session
:isDestroyed
loop
[foreach]
CreateSubmodel (RootChild):SM
loop
[foreach]
CreateCollection (SMChild):SMEC
alt
[...Count >0]
CreateProperty (SMChild):Property
else
AddDataBridgeConfig (SMChild, ...): ConfElement
AddSM2AAS(SM):AAS
ConvertAAS2File(AAS): AASFile
CreateDataBridgeFiles(): ConfigFiles
:AASFile, DataBridgeConfig
StartAASServer()
UploadAASX(AASXFile)
CreateAAS()
<<create>>
AAS
MountDataBridgeConfig()
StartDataBridge()
loop
[cycle: sample rate]
RequestNodeValues(...)
:NodeValues
WriteValues2AAS(NodeValues)
SM - Submodel, SMC - SubmodelCollection, SMEC - SubmodelElementCollection,
AAS - Asset Administration Shell

Fig. 10. Sequence Diagram for a Prototype supporting OPC UA as Asset Communication Protocol.

proposed architecture, enabling the automatic generation of AASs and DataBridge configuration files.

During the productive utilisation of the prototype in the presented OPC UA case study, it was observed that the time required for data provisioning configuration was significantly reduced, from several hours to just a few minutes. The integration of data from the showcased handling system only took 3 minutes, as opposed to the previous 7-hour process. Notably, errors in the configuration were completely avoided.

The presented prototype marks the first step towards efficient data integration in the utilization of AASs. Future work will focus on refining the components outlined in the presented architectural concept. This includes the development of a user interface for assistant-guided selection of submodel templates, the capability for automatic or graphical configuration of the DataBridge for protocols beyond OPC UA, and the integration of testing components to verify generated AASs and DataBridge configurations.

IX. Acknowledgement

This research work was funded in the research project "Basys4Transfer" by the Federal Ministry of Education and Research BMBF under the number (01IS22089G). The authors would like to thank the support of the BaSys4Transfer team.

References

[1] R. Sahal, J. G. Breslin, and M. I. Ali, “Big data and stream processing platforms for industry 4.0 requirements mapping for a predictive maintenance use case,” *Journal of Manufacturing Systems*, vol. 54, pp. 138–151, 2020, ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2019.11.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612519300937.

[2] A. Zeid, S. Sundaram, M. Moghaddam, S. Kamarthi, and T. Marion, “Interoperability in smart manufacturing: Research challenges,” *Machines*, vol. 7, no. 2, 2019, ISSN: 2075-1702. DOI: 10.3390/machines7020021. [Online]. Available: https://www.mdpi.com/2075-1702/7/2/21.

[3] S. R. Bader and M. Maleshkova, “The semantic asset administration shell,” in *Semantic Systems. The Power of AI and Knowledge Graphs*, M. Acosta, P. Cudré-Mauroux, M. Maleshkova, T. Pellegrini, H. Sack, and Y. Sure-Vetter, Eds., Cham: Springer International Publishing, 2019, pp. 159–174, ISBN: 978-3-030-33220-4.

[4] M. Azarmipour, H. Elfaham, C. Gries, and U. Epple, “Plc 4.0: A control system for industry 4.0,” in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, 2019, pp. 5513–5518. DOI: 10.1109/IECON.2019.8927026.

[5] S. Cavalieri and M. G. Salafia, “Asset administration shell for plc representation based on iec 61131–3,” *IEEE Access*, vol. 8, pp. 142 606–142 621, 2020. DOI: 10.1109/ACCESS.2020.3013890.

[6] Schnicke, Frank and Danish, Mohammad and Espen, Daniel, *BaSyx DataBridge*. [Online]. Available: https://github.com/eclipse-basyx/basyx-databridge.

[7] B. Wallner, T. Trautner, F. Pauker, and B. Kittl, “Evaluation of process control architectures for agile manufacturing systems,” *Procedia CIRP*, vol. 99, pp. 680–685, 2021, 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15-17 July 2020, ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2021.03.091. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221282712100384X.

[8] F. Schnicke, A. Haque, T. Kuhn, D. Espen, and P. O. Antonino, “Architecture blueprints to enable scalable vertical integration of assets with digital twins,” in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–8. DOI: 10.1109/ETFA52439.2022.9921728.

[9] S. Schäfer, D. Schöttke, T. Kämpfe, O. Lachmann, A. Zielstorff, and B. Tauber, “Migration and synchronization of plant segments with asset administration shells,” in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–8. DOI: 10.1109/ETFA52439.2022.9921595.

[10] Ristin, Marko and Orzelski, Andreas and Hoffmeister, Michael, *AASX Package Explorer*. [Online]. Available: https://github.com/admin-shell-io/aasx-package-explorer.

[11] I. Garmaev, T. Miny, T. Kleinert, A. Schüller, and P. Bitterlich, “Verwaltungsschalen aus excel?: Automatisierte erstellung von verwaltungsschalen aus bestandsdaten aus excel-tabellen,” *atp magazin*, vol. 65, pp. 80–86, Mar. 2023. DOI: 10.17560/atp.v65i3.2636.

[12] S. Bader, E. Barnstedt, H. Bedenbender, B. Berres, M. Billmann, and B. Boss, “Details of the Asset Administration Shell: Part 1: The exchange of information between partners in the value chain of Industrie 4.0,” Plattform Industrie 4.0, Tech. Rep., 2022.

[13] H. Bedenbender, J. Bock, B. Boss, *et al.*, “Verwaltungsschale in der Praxis,” German, Plattform Industrie 4.0, Tech. Rep., 2020.

[14] France: Ministry of Economy and Finances, Germany: Federal Ministry for Economic Affairs and Energy (BMWi), Italy: Ministero dello Sviluppo Economico, “The Structure of the Administration Shell: TRILATERAL PERSPECTIVES from France, Italy and Germany,” Plattform Industrie 4.0, Tech. Rep., 2018.

[15] Industrial Digital Twin Association e. V., *IDTA - working together to promote the Digital Twin*. [Online]. Available: https://industrialdigitaltwin.org.

[16] OPC Foundation, *OPC 40010-1 Robotics - Vertical Integration*. [Online]. Available: https://opcfoundation.org.