PLATTFORM INDUSTRIE 4.0

DISCUSSION PAPER

Asset Administration Shell
Discovery in an Industrie 4.0 System Environment

Contents

I. Definitions and Abbreviations

For the purposes of this document, the following definitions and abbreviations shall apply.

a) Definitions

Asset Administration Shell (AAS)
standardized digital representation of an asset
[SOURCE: IEC 63278-1]

asset
physical, digital, or intangible entity that has value to an individual, an organization, or a government
[SOURCE: IEC 63278-1, BASED ON IEV 741-01-04, EDITORIAL CHANGES]

discovery
a process which provides information about what physical and logical storage entities have been found within the management domain
[SOURCE: ISO/IEC 24775-2:2021(EN), 3.1.16]

In the context of the AAS, discovery means to search for AAS and submodel identifiers via a defined set of characteristics.

Note
In the context of I4.0 Systems, discovery means identifying instances based on defined characteristics. It corresponds to the „plug“ level in Plug & Produce in I4.0.
While discovery corresponds to the “plug” and querying corresponds to the „produce“ level, which is not in the scope of this document.

service discovery
activities by which a service consumer can find services which meet their specific functional and/or non-functional requirements
[SOURCE: ISO/IEC 18384-1:2016(EN), 2.34]

Note
The desired discovery is the identification of instance identifiers of AAS Services for later lookup in the registry infrastructure.

query
In the context of the AAS, query means to request endpoints with search terms.

b) Abbreviations

Abbreviation	Description
AAS	Asset Administration Shell
AML	Automation Markup Language
API	Application Programming Interface
BaSys	Basissystem Industrie 4.0 (Basic System Industry 4.0)
BMBF	Bundesministerium für Bildung und Forschung (Federal Ministry of Education and Research
BOM	Bill of Material
CESMII	Clean Energy Smart Manufacturing Innovation Institute
CSS	Capability Skill and Service
DNP	Digital Nameplate
DPP4.0	Digital Product Passport 4.0
IDTA	Industrial Digital Twin Association
I4.0	Industrie 4.0
LNI4.0	Labs Network Industry 4.0
OPC UA	Open Platform Communication Unified Architecture
PCF	Product Carbon Footprint
PWI	Proposed Work Item
REST	Representational State Transfer
SDK	Software Development Kit
SME	Submodel Elements
SMIP	Smart Manufacturing Profile
SMT	Submodel Template,
WoT - TD	Web of Things - Thing description
W3C	World Wide Web Consortium
ZVEI	German Electro and Digital Industry Association (German: Zentralverband der Elektro und Digitalindustrie)

II. Introduction

One goal of this white paper is to define the purposes and mechanisms of a discovery for the AAS and its submodels. Based on that, this paper aims to specify the frame of interfaces for discovery, in addition to the defined discovery interface of the IDTA [1]. The white paper is aimed at companies that want to offer I4.0-compatible components, systems, or services as well as at standardization organizations such as IEC and ISO, but also at platforms made available by organizations such as Catena-X and Platform I4.0. Furthermore, this white paper is also intended for anyone interested in the further development and possible uses of the AAS.

The structure of the white paper is as follows.

Chapter 2 defines the problem statement, assumptions, and requirements.

Chapter 3 gives an overview of projects under consideration.

Chapter 4 analyzes their views on the tasks of a discovery and offers hypothesis for the purposes and mechanisms of discovery. It maps such views to the terminology of IEC 63278-1 whenever possible and necessary.

Chapter 5 summarizes the observations of chapter 4.

Chapter 6 offers conclusions and recommendations.

Chapter 7 compares a project-specific and a standardized approach for discovery and proposes a frame for interfaces of infrastructure supported discovery.

Chapter 8 proposes enhancements for discovery to the AAS metamodel and AAS interface specifications and reflects on some related work.

2.1 Problem Statement

This paper presents a selection of research projects, each describing its own view of discovery and its requirements for discovery. For this purpose, general questions such as

- What do we want to find?
- What do we want to achieve with the search?
- What are the contexts we want to search for?

are asked to identify frequently occurring problem patterns based on the answers from the different projects.

2.2 Assumptions

The objective of discovery is to find AAS that match defined characteristics as well as context information managed by AAS submodels or other information sources, that will be described in chapter 6.

An AAS discovery is used to discover and retrieve identifiers of AAS instances and submodel instances. Using the AAS discovery, a client can search for AAS instances based on various criteria such as the asset type, the asset location, or the asset owner - or other published information about the AAS. Once the AAS instance is discovered, with the AAS identifier the client can look up the required service endpoints in an AAS registry to access the AAS and listed AAS submodels and access the information contained within the AAS, such as the asset‘s properties, methods, and events. By providing a standard way to discover and access AAS instances, the AAS discovery allows applications to easily find the set of AAS instances that may provide the data relevant for the business function of the application and therefore supports collaboration between stakeholders having an interest in these AAS.

This paper aims to present a generalization of AAS discovery requirements and proposes a value proposition for AAS discovery. It also discusses the importance of a standardized protocol for AAS discovery and the need for future standardization efforts for defining AAS submodels/AAS metamodel elements that support specific types of discoveries with different qualities.

2.3 Requirements for Discovery

- The value acquisition (request & result of a discovery) shall follow a simple syntax and semantics – though the results have to be appropriate/efficient/effective.

- There shall be a distinction between discovery (identifier determination) and querying (value determination). Discovery does not require understanding the semantics of the question, while querying requires semantic resolution.

- The discovery process shall be the first step in a two-stage approach for AAS instance value processing:

  1) Identification of instances based on specific topics (functional, spatial, etc.), and then

  2) querying and processing their specific content-related information.

  Comment:

  Specified SMT help identifying AAS in a discovery process (e.g., DNP/ECLASS characteristics) – such as AAS metamodel elements do.

  o Future standardization efforts are required for defining AAS submodels/metamodel elements that support specific types of discoveries with different qualities (of scope and parameters to search for).

  o Discovery therefore uses/supports the generic engineering requirement "Classification" (see [11]).

- Submodel elements for a search can be found in AAS submodels, the AAS metamodel, or other contextual information sources.

- A discovery protocol definition should be open for future types of discovery.

- The discoverable characteristics shall be organized/implemented/stored in different technical ways (e.g., stored in common database, broadcasting request to instances of AAS etc.) with minimal constraints imposed by the discovery interface/protocol.

- Discovery types and their properties have to be standardized to be able to use the type in applications to answer business questions in applications.

- Access rights must be considered. AAS instances determine their visibility in specific types of discoveries and potentially must support corresponding mechanisms, e.g., creating active catalog entries with values of AAS submodels attached to the dedicated AAS instance.

- Specified SMT should be able to be used to support discovery by indicating the supported searches.

III. Considered Projects

3.1 BaSys, BaSyx, ÜberProd

The open-source Eclipse project BaSyx invented in the BaSys projects funded by the BMBF provides SDK in different programming languages (Python, C#, Java, C++) for creating and handling AAS. Beside the main project series, there are different satellite projects for the transfer of the concepts and technologies in real application scenarios with SME also funded by the BMBF (e.g., BaSys ÜberProd, BaSys4Brenner, BaSys4Forestry, BaSys4FluidSim, BaSys4SupplyQ). Regarding discovery, the discoverability of devices with respect to certain properties and capabilities was a particular issue. The problem was solved in a detoured way by either using a "TaggedDirectory" service, outsourcing the discovery mechanism to external software or providing it as functionality from a separate Asset Administration Shell. For the first solution, this means that AAS descriptors and submodel descriptors can be tagged. Using associated search queries, these descriptors can be found. For example, an AAS can be tagged with [„Mill", „VendorA", „OPC UA"]. Search queries can then be used to search for individual tags or tag combinations, which are then logically conjunct. A query for „Mill" returns all descriptors with the „Mill" tag, a query for [„Mill", „VendorA"] returns all descriptors that have both tags. This makes it possible to map binary properties. More complex queries, e.g., „Return me all product AAS created before 05/26/2023" cannot be mapped over it. An implementation of the concept was realized in the main projects as well as in some of the satellite projects, such as BaSys4SupplyQ. Due to the number of AAS used, a more precise filtering of results became necessary when implementing the use case. A component for filtering AAS can process application-specific filter criteria to find a selection of AAS. Two examples of filter criteria used within the projects are:

- assignment to a customer and
- creation within a defined time period.

3.2 ZVEI Showcase PCF@ControlCabinet

The ZVEI showcase PCF@ControlCabinet demonstrator aims to collect practical experiences during the implementation of the DPP using the AAS and other I4.0 technologies. The demonstrator consists of a physical electrical cabinet with 90+ components from 15+ companies. Each component is identifiable by a standardized IEC 61406 identification link which is used to locate AAS provided by the device vendor, and in particular, DPP related submodels containing documentation, technical properties, and PCF. These SMT are developed by ZVEI and IDTA and are denoted as DPP4.0.

The demonstrator called "Level 2", which was presented during the Hannover Fair 2023, consists of distributed AAS servers sharing a common registry. The user can interact with the AAS of the control cabinet by scanning identification links (QR-Code) using a dedicated app, and influence the configuration of components within the cabinet, i.e., modifying the BOM submodel of the cabinet itself. Based on the contents of the BOM of the cabinet, a dashboard application calculates the cradle-to-gate carbon footprint of the cabinet and included components.

3.3 LNI4.0 AAS Testbed

The LNI4.0 AAS Testbed delivered the following use cases of the AAS:

1. **AAS for brownfield**
   The scope was to demonstrate how to apply the concept of the AAS submodel to integrate items of the installed base in a uniform way and make them available as I4.0 components represented by its AAS.

2. **Communication and Connectivity**
   The scope was to demonstrate how to use AAS to connect assets to central platforms which realized the control of production flow.
3. **AAS of a product**
   The scope was to demonstrate the configuration of a product using its AAS and use such configuration to derive the production flow.

4. **Interoperability – Demo distributed in different German locations**
   The scope was to demonstrate that the whole configuration of a production system can work if it is distributed over different locations by using the AAS as a standardized interoperability concept.

5. **Track and Traceability**
   The scope was to demonstrate the tracking and tracing of the product through its production using its AAS. This is also integrated internationally, with the USA and Korea as leading countries. The work on this use case is still in progress and a demonstrator with Japan is under discussion.

6. **Invitation to bid for process step engraving - international between Korea and Germany**
   The scope was to demonstrate the flexible choice of production capabilities to execute one specific production step dependent on different production goals and different assets which offer the capabilities required for that step via its AAS.

All these use cases have been demonstrated by a production scenario of a ball pen.

The use cases successfully demonstrated the advantages of the concepts of AAS and submodels to the extent of their goals, as described above. This was achieved by using the AAS type 2 (client-server) configuration, which uses AAS and submodels as a server for asset related information. All algorithms to integrate the assets into a production system as well as the logic to use the AAS of the product to produce were provided by an external platform in a project-specific way, this includes the functionality of discovery.

3.4 CESMI/LNI4.0 PCF Showcase

The CESMI/LNI4.0 PCF Showcase demonstrates the production scenario of a ball pen with an individual engravement and calculation of its PCF. All production equipment as well as the products have been represented as I4.0 components by its AAS. Discovery was realized by external platforms.

The PCF of the produced products was calculated out of information delivered online by the production assets. The concept of AAS submodels has been used to get such information via OPC UA from production assets and represent it in a uniform way (by means of energy measurements of the production machines or scope 2 and scope 3 calculations of the necessary material logistics for the productions).

The concept of AAS and submodels was used to showcase the integration of different standards, such as the CESMII SMIP interoperable with AAS submodels.

The advantages of the AAS as a unifying concept for access to information and representation of information were successfully demonstrated.

Like in the LNI4.0 demonstrator use cases, this was achieved by using the AAS type 2 (client-server) configuration, which uses AAS and submodels as a server for asset related information. All algorithms to integrate the assets into a production system as well as the logic to use the AAS of the product to produce it were provided by different platforms and tools in a project-specific way.

3.5 Documentation Maintenance for Device Replacement

In [2] it was shown how the AAS can be used to update the associated documentation semi-automatically after a device replacement in a production system. Here, the topic of discovery played a role in finding relationships to which the device to be replaced is connected in an AAS structure that simultaneously contains different views of the production system. Examples of these views include the mechanical and electrical structure of the production system, or the functional and location relationships between the device and the production system. When a device is replaced, the AAS of the device is retrieved and its relationships to other assets in the overall system are traced. The found AAS of the linked assets are then updated so that they are no longer related to the old asset‘s AAS, but to the new asset‘s AAS. In addition, the former relationships are moved from the old asset‘s AAS to that of the new asset, as the new asset has taken the place of the old asset in the production system.

The relationships in the AAS structure of the production system were implemented using the Relationship Elements and Reference Elements of the AAS metamodel [3]. The views, for locating a device in a specific context, were implemented through groupings using various AAS metamodel elements. With these solutions, a discovery was indirectly implemented that, after finding the device AAS, finds the relationships from the different views to other assets within the production system or even beyond. If this concept is applied to systems with a high level of complexity, the number of relationships with its different views increases, resulting in an increasing complexity in the AAS structure. A better solution would be to use the information on already existing AAS submodels. For example, the BOM submodel [4] can already be used to generate a view of the AAS structure for a production system. A discovery in this case has to realize the possibility to find the device to be replaced as an entity in the other AAS of the AAS structure of the production system. This would eliminate the need to implement additional Relationship Elements and Reference Elements. However, it must be noted that the entities must still be provided with the information on the views, e.g., the functional and physical aspects.

3.6 Verification of the Interoperability of Fluid Power Systems by the example of Plug-and-Produce (FL4)

FL4 was a research project that aimed to demonstrate the feasibility of plug-and-produce by realizing a specific fluid technology application based on I4.0-compliant components [5].

The project‘s core objective was the creation of I4.0-compliant fluid technology components that develop their full potential when providing other system participants with I4.0-compliant service and information interfaces in encapsulated form [6]. A profound understanding of fluid mechanics and information technology knowledge are prerequisites for new digital business models, such as predictive maintenance or automated commissioning.

Technical implementation of the components requires targeted modelling. Considering the regulatory framework under development, such as AAS, in the context of I4.0, it is ensured that the partial solutions can be combined into complex overall applications, such as commissioning [7]. The aim was to ensure that the technology systems can be put into operation in a simpler and largely automated manner compared to existing systems.

IV. Analysis of projects

4.1 Common problem patterns

This section tries to derive common problem patterns of the projects outlined in chapter 3.

As introduced in chapter 2.1 the following general questions are asked:

- What do we want to find?
- What do we want to achieve with the search?
- What are the categories we want to search for?

To structure the discussion, the following picture of IEC 63278-1 [8] is used.

It shows different roles and their interactions with the AAS. It is deemed in any case such interactions are performed by making use of an AAS user application which finally accesses the AAS interface(s). E.g., if the AAS responsible governs the AAS, it uses an AAS user application to manage the AAS submodels listed in the AAS and, among others, defines access rights and usage rights at his discretion.

Since all of the projects which are part of this paper use registries in one or another form to provide AAS interface(s) and to list submodels which belong to an AAS. It appears to be obvious that AAS user applications have a common need to search for AAS and submodels by different criteria. It seems to be beneficial that such search returns identifiers of AAS and submodels which can be used to search for interfaces provided by AAS through different means such as registries, belonging to such identifiers and used to access AAS and submodels.

4.2 Projects

4.2.1 BaSys, BaSyx, ÜberProd

BaSyx implemented a directory using tags to enable to search. In the structure of Figure 1 the AAS responsible governs the AAS by creating and adding such tags to it.

An interesting question is if there are examples for com-

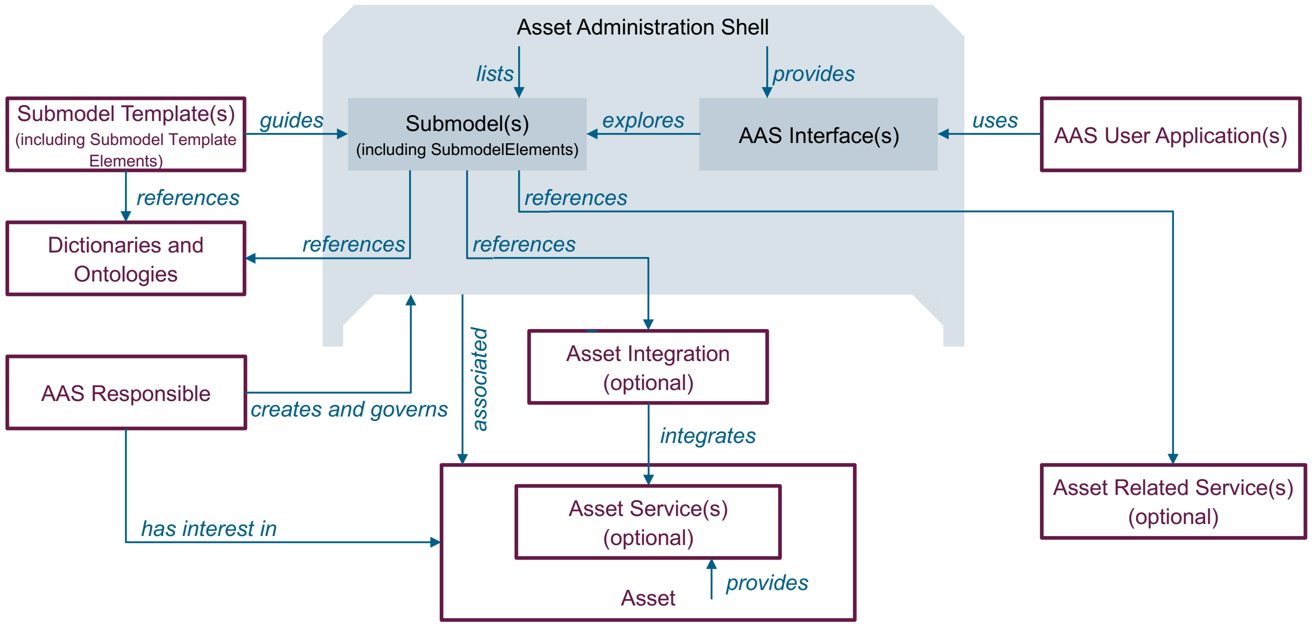

Figure 1: IEC 63278-1 Detailed overview of AAS and related entities

mon categories which are also needed in other projects, even if the application purpose is different.

Hypothesis 1:

“Mill” is an example for a category to describe an inner relation because searching for the tag “Mill” means, among other things, to search for AAS of assets which belong to “Mill”.

The same holds true for “VendorA” because it searches for AAS of assets “VendorA” is interested in, e.g., as a producer of a Mill

Hypothesis 2:

“OPC UA” is an example for a category to describe technical capabilities because to search for the tag “OPC UA” means, among other things, to search for AAS of assets which offer information made available via OPC UA.

Hypothesis 3:

BaSyx does not use the content of AAS or submodels for such kind of search. Although there are disadvantages to not being able to search for specific content, this has at least two advantages:

a) It makes search easier and less application dependent.

b) It decouples the search from access to AAS and submodels. No special access rights to investigate AAS for search are necessary.

Remark: The content of the tags could also be implemented as elements of AAS submodels. This would mean that the corresponding access rights must be available to investigate the AAS for the purposes of the search.

In BaSys, ÜberProd implemented scenarios AAS are used to implement plug and play in production to a certain extent. Devices offer properties and capabilities via their AAS which are needed for production steps. The purpose of discovery in such scenarios is to search at runtime for AAS which offer the capabilities needed at this point. Interestingly, this was not implemented by tagging as described in section 4.2.1 but detoured to other means.

Hypothesis 4:

One Level higher, there is no conceptual difference to search for specific capabilities as in Über-Prod or to search für “OPC UA” in BaSyx. In both cases, an AAS user application searches for AAS which offer information according to a well-known specification.

4.2.2 ZVEI Showcase PCF@ControlCabinet

One major goal of the ZVEI showcase is to calculate PCF values of a control cabinet out of the PCF values and some other information of the components the control cabinet is built of. A recalculation of the control cabinet’s PCF value shall be done if its configuration changes, e.g., by adding or replacing components. For this purpose, every component is represented by an own AAS which lists at least its belonging PCF submodel.

To express the components belonging to the control cabinet, the BOM submodel is used.

The advantage is that the project could make use out of an existing SMT.

The disadvantage is that it is very complex for the AAS responsible to govern the AAS of his components in a way that access to the BOM submodel is restricted to defined AAS user applications and in parallel to enable right PCF calculation even without access rights to the BOM submodel.

Hypothesis 5:

The usage of the BOM submodel in this case is another example to express an inner relation, namely to express the components belonging to the control cabinet at a certain point in time.

However, to use a submodel for the purpose to express discovery related information in this case an inner relationship has at least two disadvantages:

a) It requires every vendor to deliver a BOM submodel for its components, which might not be in the interest of the vendor.
b) It requires the AAS responsible to govern their AAS in a way that the access is restricted according to their business needs, e.g., their IP and in addition in a way to enable correct calculation of PCF values.

To govern access rights of AAS, submodels and submodel elements, the security SMT was used to create a security submodel for the AAS.

To indicate that an AAS submodel is a PCF submodel, a special PCF path in the repository per submodel kind was used.

Hypothesis 6:

This is an additional example to search for a technical capability in this case to deliver values according to a wellknown specification, in this case PCF.

4.2.3 LNI4.0 Demo-Factory Use Cases

The purpose of discovery in this project was to search and configure the asset to execute the next production step of ball pen according to its target specification, e.g., the text to be engraved.

Hypothesis 7:

This is an additional example to search for a technical capability according to a wellknown specification, in this case to engrave a defined text on a ball pen.

4.2.4 CESMI/LNI4.0 PCF Showcase

This showcase enhances 4.2.3 by calculation of the ball pen's PCF value using the energy consumption reporting of the production units.

The purpose of discovery in this case is to search and configure the used assets to execute the production of the ball pen and to retrieve the PCF relevant information while producing the ball pen.

Hypothesis 8:

This is an enhancement of hypothesis 6 by searching for an inner relationship, namely all assets used to produce the ball pen.

4.2.5 Documentation maintenance for device replacement

This project is very similar to Section 3.2. Instead of getting PCF values updated when configurations are changed, the documentation shall be updated in such case. The purpose of discovery was to find the assets belonging to a production system and to find all relationships to be updated when replacing a component of the system.

There are additional findings, e.g., the missing forward/backward references, which modify the system configuration complex. However, such findings are out of the scope of this paper.

Hypothesis 9:

The search for Relationship Elements is an additional example to search for different inner relationships, e.g., the mechanical structure of the system or the electrical structure of the system etc.

Hypothesis 10:

The search for AAS submodels which contain documentation is an additional example to search for a technical capability according to a wellknown specification, in this case to a set of documentations for an asset.

4.2.6 Verification of the Interoperability of Fluid Power Systems by the example of Plug-and-Produce (FL4)

FL4 focused on mapping the generalizable properties of the fluid components. To this end, submodels were created for each I4.0 component category, encapsulating and generalizing the technical aspects of the asset so that the functions of the fluid technology components could be accessed via the AAS interfaces [5]. In addition, the I4.0 system described in this way has been designed as a distributed system. A dedicated microservice was created for each individual instance of a submodel, which offers its SME, such as properties, operations and events, as endpoints in the system [6].

Hypothesis 11:

It must be possible to search and list AAS submodels with technical and functional features, e.g., in the fluid power domain like pressure build-up, the end position setup and the performance of leakage tests etc. The AAS features must be selected, aggregated and filtered.

The individual I4.0 components call each other via the defined interfaces, e.g., for plug-and-produce, in the distributed system [6]. This makes the relationships and dependencies between the individual assets extremely relevant to cooperative interaction. To map these, dedicated submodels were created to map the various aspects such as fluid connections, electrical connections, structural connections, organizational dependencies, etc. These relationships needed to be obtained. It had to be possible to list and find these dependencies [7].

Hypothesis 12:

To collect and return the complex relationships and topological dependencies between the different components of I4.0 (e.g., in fluid power), it is necessary to query and correlate numerous AAS submodels and their relationships. Same as in Hypothesis 11 it is an example to search for different inner relationships.

V. Observations

All projects have shown that it is possible to implement the functionality needed to fulfil their goals by using AAS and submodels as a basis. The advantages to use the AAS and submodels are mainly achieved by using standardized access to information by using the AAS interface, as well as standardized representation of information by using AAS submodels.

However, a significant step to lower implementation and engineering effort is to a wide extent not achieved. Many tasks to configure systems remain the same as of today but done with other methods than in classical engineering. The basic structure of AAS and submodels is used to integrate components into a system but in project-specific ways.

Examples to express and search for inner relationships

- BaSyx uses tagging.
- ZVEI showcase uses the BOM submodel.
- LNI4.0 uses manually configured relations of related endpoints (addresses).
- Documentation maintenance uses Relationship Elements of the metamodel in a project-specific way.
- FL4 used dedicated submodels to map various aspects, e.g., organizational dependencies.

Examples to express well-known specifications

- BaSyx uses tagging.
- ZVEI showcase uses a directory-oriented structuring for submodels.
- LNI4.0 does not use an explicit method but assumes technical compatibility to be assured by the configuration of related endpoints.

Examples to express restriction of use

- Most projects restrict the access to AAS and submodels by restriction of access to the places the relevant data are stored, e.g., to registries or AAS repositories.
- ZVEI showcase uses a dedicated submodel to express further restriction of use of AAS, submodels and submodel elements.

VI. Conclusions and recommendations

This paper aims to clarify the meaning of discovery and provide recommendations for it as mentioned in chapter 2.1 following the guiding questions:

- What do we want to find?
- What do we want to achieve with the search?
- What are the aspects we want to search for, and is it beneficial to have some common categories of such aspects?

According to the results of the analysis of the considered projects, we would like to offer the following answers and to propose a value proposition for discovery.

6.1 Answers to guiding questions

a) What do we want to find?

For AAS user applications, it is important to be able to search for AAS and for submodels listed in AAS. Such entities are identifiable by worldwide unique identifiers. Those identifiers enable the search for interfaces of AAS. Independent on how and where to find such interfaces and where they are finally located it is recommended that discovery returns AAS identifiers and submodel identifiers.

b) What do we want to achieve with the search?

From a usage point of view, there is a big variety of search goals. Currently, there is no distinction between project-specific search goals and search goals which have a more general character. That's why, as of today, all search goals are expressed project-specific, and a more general discovery is missing.

It appears to be beneficial by lowering implementation efforts, improving interoperability, and increasing flexibility to identify search goals which have a general character. This means that the specific goal of the search from a one level above perspective shares the same category of search goal with others. Such search categories could be used for discovery.

c) What are the aspects we want to search for?

There are three aspects of search goals which seem to be common in all projects discussed in this paper.

I. The search for AAS and submodels an AAS user application has access rights and usage rights. The guiding search goal is to get AAS identifier and submodel identifier the searching AAS user application is allowed to interact with. It is proposed to name this the category "Restriction of use".

II. The search for AAS and submodels an AAS user application can interact with because both support the same wellknown specification. It is proposed to name this the category "Classification".

III. The search for AAS and submodels an AAS user application restricts the search results according to an inner relationship. It is proposed to name this the category "Context".

6.2 Conclusions

From the experiences of the projects discussed in this paper, it would be beneficial if an AAS responsible would be able to govern AAS and submodels by attaching expressions of the categories proposed above to AAS and submodels. This would enable AAS user applications to search for combinations of such expressions. To implement services which offer such search as service of a common infrastructure, the categories must become harmonized. The expressions may remain project-specific even if their harmonization would improve interoperability and flexibility in addition.

It is proposed to name a search for AAS identifiers and submodel identifiers according to a combination of expressions assigned to a harmonized set of search categories and attached to AAS and submodels "AAS discovery".

6.3 Value proposition of AAS discovery

AAS discovery is a critical process in I4.0 systems, enabling clients to discover and retrieve identifiers of AAS instances. The importance of a standardized approach for AAS discovery and the need for future standardization efforts for defining submodels/metamodel elements that support specific types of discoveries with different qualities have been discussed in this paper. The value proposition for AAS discovery is to provide a standardized way to discover and access AAS instances, which allows applications to easily find the set of AAS instances that may provide the information relevant for the business function of the application. A standardized approach to AAS discovery will facilitate collaboration between stakeholders having an interest in these AAS and enable plug-and-produce functionality in I4.0 systems.

6.4 Relation to (ISO/IEC) JTC1/SC41/AG31 External Liaisons collaboration “IoT/Digital Twin”

ISO/IEC JTC1/SC41/AG31 addresses discovery in the context of data spaces to be used in IoT Applications and for Digital Twins. This is reflected in its principles, namely principle four and thirteen, as shown below. These principles will guide the integration of IoT and Digital Twins in data spaces.

- **Principle 4: data interoperability is enabled by a common languag**e.
  A common language for data Interoperability and discovery is required. It requires the exchange of meta-data based on ontologies and semantic information. It is used for knowledge discovery in the data space.

- **Principle 13: Integrated data management**
  An integrated data management platform that enables the full breadth of integrated data management capabilities including discovery, federated governance, curation, and orchestration

Both principles would be supported by AAS discovery, as proposed in chapter 6.1. I4.0 assets are often things in IoT. The AAS is a digital representation of such assets. Insofar, AAS discovery offers an answer on how to implement the ISO/IEC JTC1/SC41/AG31 principles in a standardized way.

The intention is to facilitate and foster the international standardization for a discovery methodology on Digital Twins through an PWI inside ISO/IEC JTC1/SC41/WG6 to be facilitated.

VII. Projects specific and standardized discovery

This section describes different approaches for discovery and differentiates between discovery which is supported by a standardized infrastructure and discovery solely supported in a project-specific way. Therefore, the following documents are discussed:

[3] Industrial Digital Twin Association, Hrsg.: „Asset Administration Shell Specification - Part 1: Metamodel Schema“. 2023, March.

[9] Industrial Digital Twin Association, Hrsg.: „Asset Administration Shell Specification - Part 2: Application Programming Interfaces“. 2023, June.

[10] Andrei Ciortea, Simon Mayer, Florian Michahelles: “Repurposing Manufacturing Lines on the Fly with Multiagent Systems for the Web of Things”, AAMAS 2018, July 10-15, 2018, Stockholm, Sweden

[11] Plattform Industrie 4.0: Functional View of the Asset Administration Shell in an Industrie4.0 system environment. Discussion Paper https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/Functional-View.html

The chapter describes an example for a project-specific discovery with high flexibility of production systems to adapt such systems on the fly, e.g., to different production goals.

It will make a proposal on how to use AAS to structure and standardize the process of discovery necessary to achieve such flexibility in a standardized way.

Therefore, proposals to adapt and use the AAS metamodel and the AAS interface are made.

7.1 Project-specific Discovery

Ciorteas, Mayer and Michahelles discuss in [10] a flexible and selfre-configuring production system. They show that it is possible to generate manufacturing workflows to execute manufacturing tasks at run time if all necessary capabilities to perform a manufacturing task are available in a production system. There is no extensive upfront engineering necessary, and the system can find different solutions for different boundary conditions such as "fastest production", "cheapest production", "best energy-efficient production" etc. by reasoning on the capabilities of the artifacts accordingly.

The discussed scenario addresses several aspects of I4.0 especially:

- To lower engineering effort and increase flexibility
- To support plug & produce

The central picture with the main elements of the system and related roles as well as the general approach are shown below.

The approach is summarized in the paper as below:

"Following our analysis of relevant related work, we integrated these recent developments into an approach to design scalable and flexible agent-based manufacturing systems. The novelty of our approach is two-fold: (i) it integrates PRS-like BDI agents with first-principles planning for Web-based artifacts, and (ii) it applies semantic multi-agent organizations for repurposing manufacturing lines on the fly. Figure 1 depicts an overview of the layers of abstraction used in our approach. The manufacturing systems use the Web as an application architecture, which induces scalability and evolvability (cf. Section 2.2, see also [21]). Then, the various programming abstractions (e.g., agents, artifacts, organizations), which were inspired by the JaCaMo meta-model for MAOP [4], further enhance the modularity of the systems. Together, modularity, BDI reasoning, and automated planning enable the flexible pursuit of manufacturing goals, while the use of high-level programming abstractions (e.g., organizations) and automated planning allows engineers to repurpose the systems on the fly (see also Section 4.1.2). In the following, in Section 3.1 we first discuss considerations for designing the agent environment and single-agent planning. In Section 3.2, we elaborate on using semantic multi-agent organizations for the dynamic reconfiguration of manufacturing lines." [10]

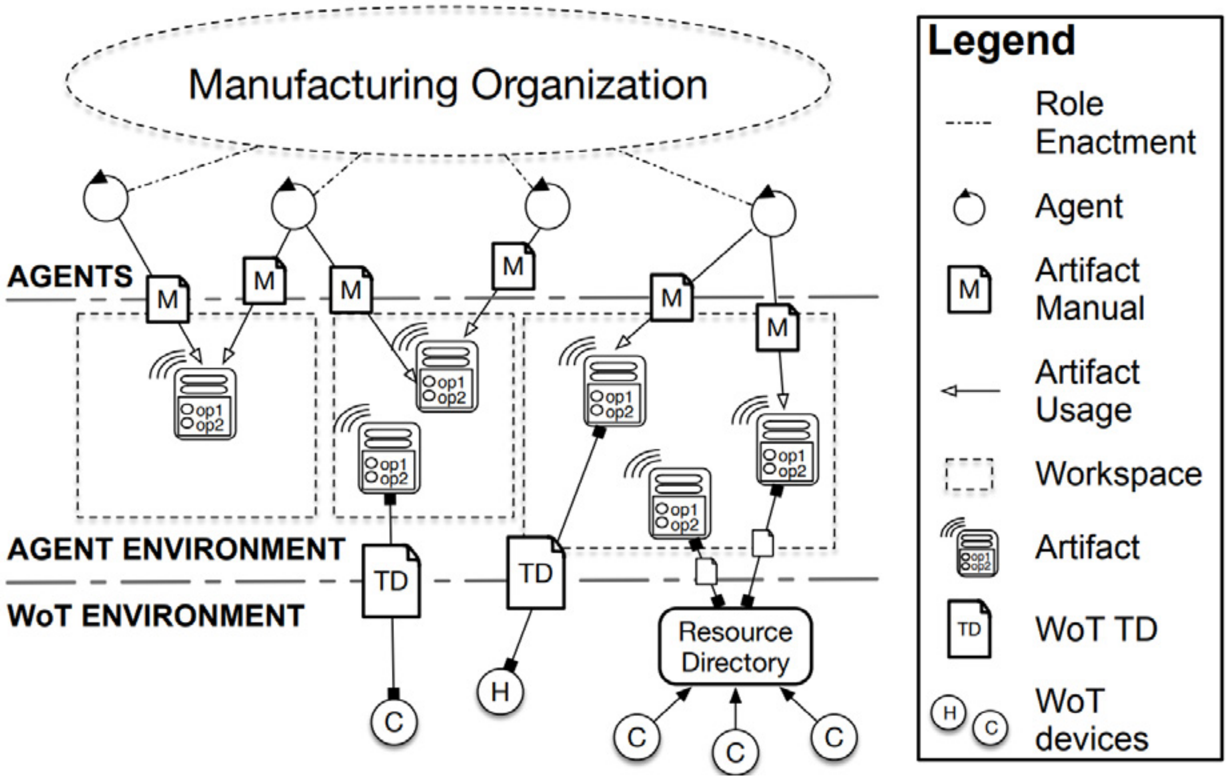

Figure 2: Copy of the central Figure 1 from [7]

7.2 Standardized AAS discovery

The project as explained in section 7.1 uses a classical project-specific discovery approach characterized by the following facts:

- It works for the project only it is built for.
- Semantics is a project-specific one.
- Plug & produce is implemented by project-specific agent-based search and a project-specific matching of capabilities.

However, it is an early and very promising attempt to structure systems in a way to achieve plug & produce and to showcase how technologies can be combined in an implementation.

It shows a way forward to a standardized infrastructure supported AAS discovery.

This is the case because the project-specific abstraction maps on a conceptual level very nicely with IEC 63278-1.

In a project-specific way:

- Agents are used to search for artifacts which integrate different devices into an application in a flexible and on the fly manner.
- The configuration of the search characteristics of the agents is done by a manufacturing organization in a project-specific way, e.g., by ontologies.
- Artifact manuals are used to express all artifacts (e.g., functions) belonging to a device. Artifact manuals are the digital representation of a device.
- Agents return endpoints of artifacts to be interacted with when using a device. Such artifacts harmonize functions of devices even if the device's interfaces are different. W3C WoT TD are used to make device functionality usable via its artifacts.

This structure is very similar if not the same as the I40-Component's structure.

The search of the agents is from a functional perspective the same as AAS discovery.

In a standardized way:

- AAS discovery uses standardized characteristics in form of the three categories, as proposed in chapter 6.1 and 6.2. to search for submodels which integrate different devices into an application in a flexible and on the fly manner.
- The configuration of the AAS discovery characteristics is done by an organization by configuring AAS and submodels in a standardized way with expressions of the three discovery categories.
- AAS is used to list all submodels associated to an asset (in this case, a device). AAS is the standardized digital representation of an asset.
- AAS discovery returns standardized identifiers to be looked up in an infrastructure, e.g., a registry to get endpoints of submodels to be interacted with when using an asset. Such submodels harmonize and standardize functions of assets, even if the asset's interfaces are different. Asset integration is used to make asset services (in this case, device functionality) usable via its submodels.

Standardized plug and play become possible by using a standardized AAS discovery.

The conceptual mapping to IEC 63278-1 is shown in Figure 3.

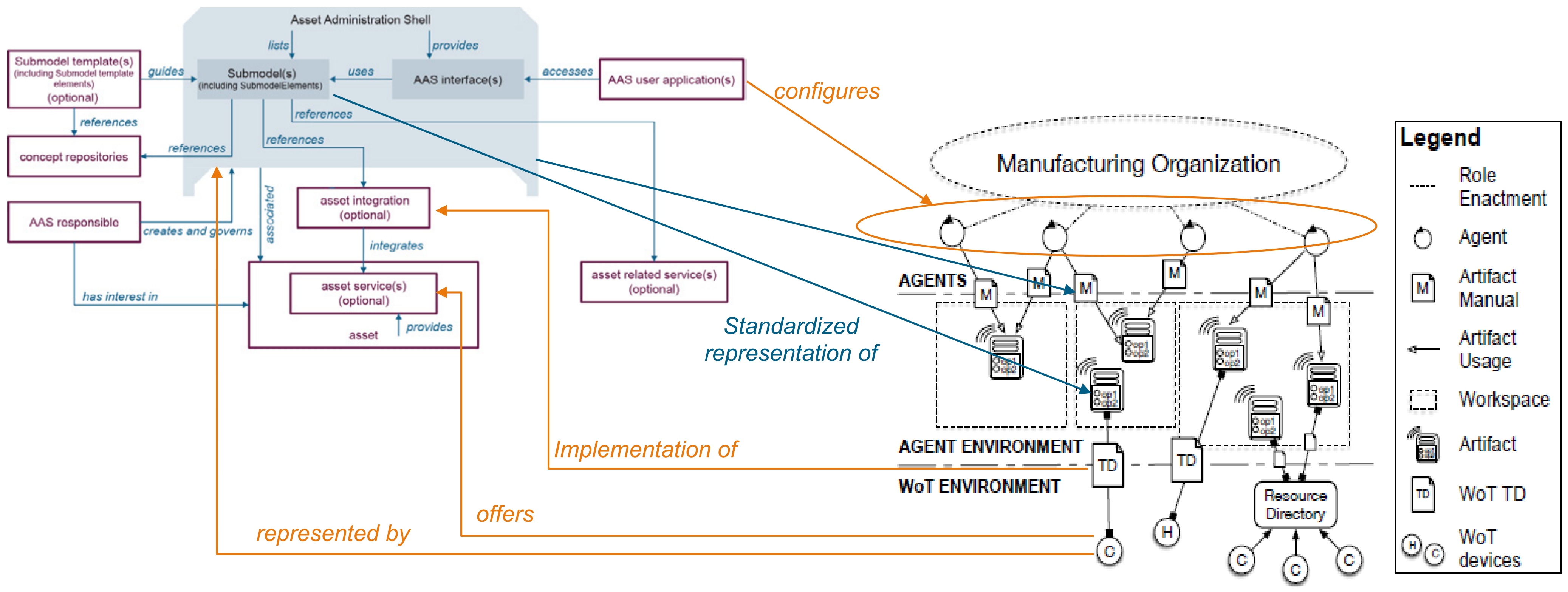

Figure 3: Conceptual analogies to IEC 63278-1

Figure 3 does not show the ontologies used to describe the capabilities of the artifacts.

Since they, in fact, describe their capabilities in a semantic way, the conceptual analogy to IEC 63278-1 is the concept repository.

[11] discusses the AAS infrastructure in depth and separates infrastructure services from such software services, which are application components. Infrastructure services are systemically relevant, characterized by the fact that they are syntactically and semantically uniform to all AAS user applications, using the infrastructure. In contrast to this, application services are syntactically and semantically uniform to one or more AAS user applications. This is shown is Figure 4.

This would enable the definition of infrastructure services to discover AAS and submodels by using combinations of expressions out of these categories. Therefore, the infrastructure does not need to know the application-specific meaning of the expressions and would search such by matchmaking of expressions in categories.

Note:

Figure 5 shows all concepts of services as proposed in [11]. However, the scope of this discussion paper is solely on the analysis of projects regarding discovery and a proposal of standardized discovery and the therefore necessary parts of meta information management.

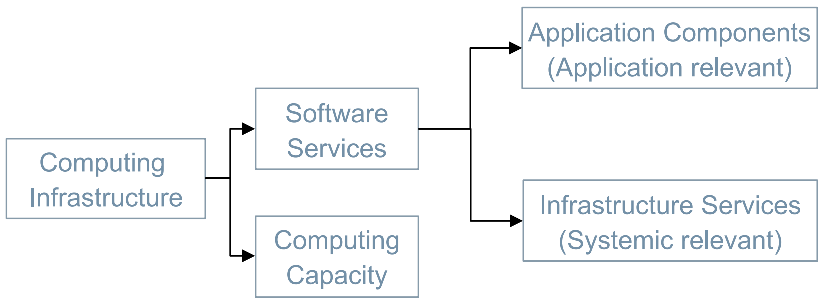
Computing Infrastructure
Software Services
Computing Capacity
Application Components (Application relevant)
Infrastructure Services (Systemic relevant)

Figure 4 Infrastructure Services and Application Components

The paper proposes the usage of meta information applicable to AAS and submodels and to group them in categories which are syntactically and semantically uniform to every AAS user application using the infrastructure. The values of such categories may remain application-specific.

In this sense, the paper proposes an infrastructure supported discovery which could be achieved, e.g., by a to be standardized usage of the concepts described in [10] mainly by standardization of interfaces and APIs.

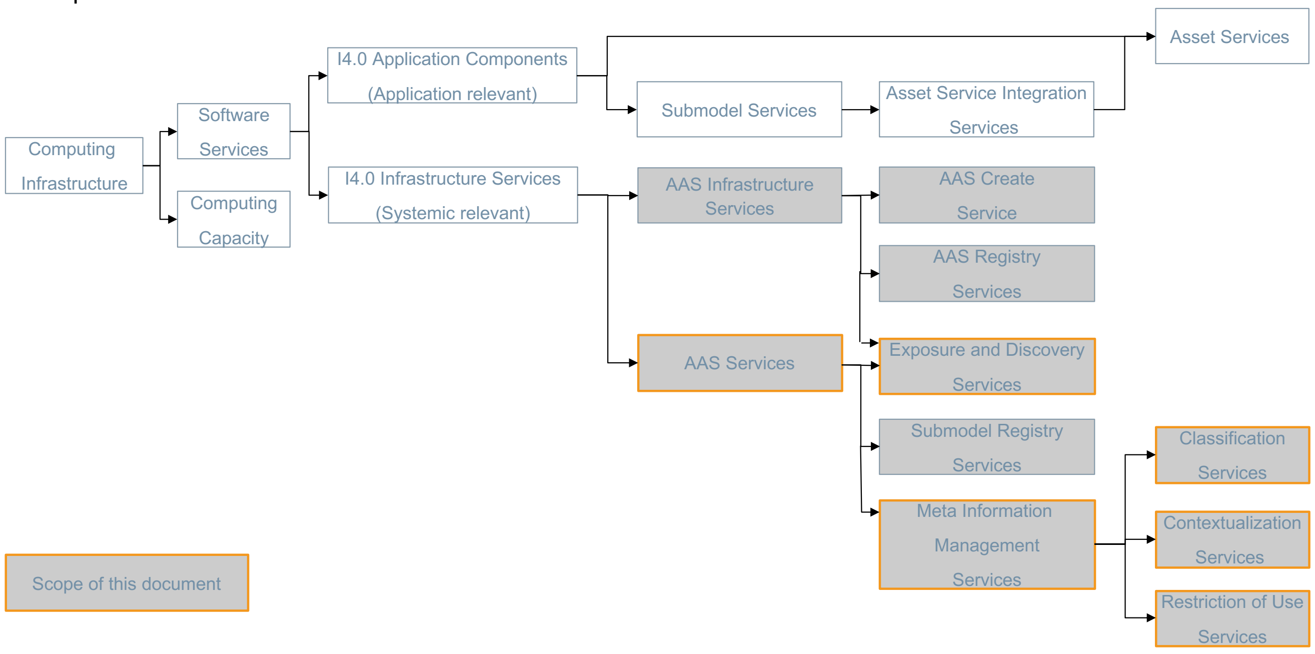
Computing Infrastructure
Software Services
Computing Capacity
I4.0 Application Components (Application relevant)
I4.0 Infrastructure Services (Systemic relevant)
Asset Services
Submodel Services
Asset Service Integration Services
AAS Infrastructure Services
AAS Create Service
AAS Registry Services
AAS Services
Exposure and Discovery Services
Submodel Registry Services
Meta Information Management Services
Classification Services
Contextualization Services
Restriction of Use Services
Scope of this document

Figure 5: Scope of the Functional View of the AAS with AAS discovery as part of it

VIII. Adaptations of existing specifications for AAS discovery

This chapter makes proposals on adaptations which are minimal invasive to the existing specification.

Requirements:

RQ 1: The adaptations of the AAS metamodel shall be as less as possible.

RQ 2: All adaptations of the AAS metamodel shall be backwards-compatible with version 3.0 of the metamodel specified in [3].

RQ 3: All adaptations of the AAS API shall be backwards compatible with version 3.0 of the API specified in [9].

8.1 Adaptation of the AAS metamodel - Extension of qualifier kind

Proposed Solution:

To enable the search for suitable AAS and their submodels, it is proposed to use tags for the search. In order not to adapt the AAS metamodel too much, it is recommended to introduce a new QualifierKind called „DiscoveryQualifier." This is a backward-compatible change, and the QualifierKind is currently only experimental, so this change should not have a significant impact.

To use the new DiscoveryQualifier, various types, analogous to the categories in the Functional View (e.g., Context Information or Classification Information), should be specified in the annex of the AAS metamodel specification. These types are described as an incomplete list. String is specified as the valueType, and for each type, a list of possible values is also recommended to make searches more efficient. These values, like the types, are non-normative and should therefore be located in the annex of the AAS metamodel specification.

Implications and necessary changes

For each tag, a qualifier must be created, since DataTypeDefXsd does not allow arrays. To access the qualifiers, they should get a local identifier. Moreover, it is also possible to extend the valueType to allow arrays of strings, so that not a qualifier must be created for each value.

For use in the API, it should be recommended that qualifiers are individually accessible, meaning that specific operations must be added. It is recommended to add the following operations: Create, Delete, and Query Qualifiers. Updating qualifiers should not be possible, as they only have one value each, making deletion and creation sufficient. If the valueType is extended to also use arrays of strings, an operation for Modifying the Qualifiers is needed.

Current status of the metamodel

In Figure 6, Figure 7 and Figure 8 the qualifier class in UML and Table format as well as the qualifier kind enumeration from [3] is given. In Figure 9 the adapted metamodel with the DiscoveryQualifier as a new enumeration element in QualifierKind is presented.

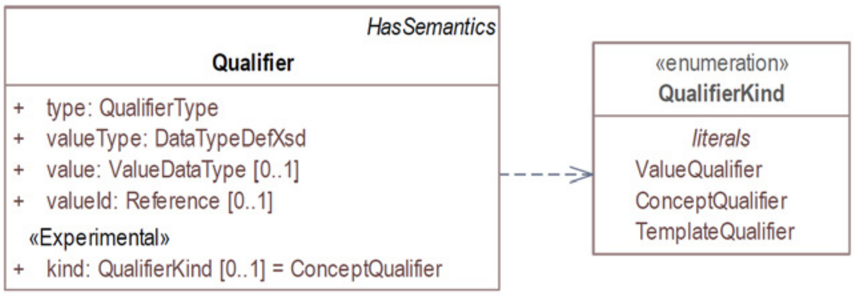

Figure 6: Copy of Metamodel of Qualifiers [3]

Class:	Qualifier		
Explanation	A qualifier is essentially a type-value-pair. Depending on the kind of qualifier, it makes additional statements • w.r.t. the value of the qualified element, • w.r.t the concept, i.e. semantic definition of the qualified element, • w.r.t. existence and other meta information of the qualified element type. Constraint AASd-006: If both, the value and the valueId of a Qualifier are present, the value needs to be identical to the value of the referenced coded value in *Qualifier/valueId*. Constraint AASd-020: The value of Qualifier/value shall be consistent with the data type as defined in *Qualifier/valueType*.		
Inherits from:	**HasSemantics**		
Attribute	Explanation	Type	Card.
Kind <<Experimental>>	The qualifier kind describes the kind of qualifier that is applied to the element. Default: ConceptQualifier	QualifierKind	0..1
type	The qualifier type describes the type of qualifier that is applied to the element.	QualifierType	1
valueType	Data type of the qualifier *value*	DataTypeDefXsd	1
value	The qualifier value is the value of the qualifier.	ValueDataType	0..1
valueId	Reference to the global unique ID of a coded value *Note:* it is recommended to use an external reference.	Reference	0..1

Figure 7: Qualifier Class and its attributes [3]

Enumeration:	QualifierKind
Explanation	Enumeration for kinds of qualifiers
Set of:	--
Literal	Explanation
ValueQualifier	Qualifies the value of the element; the corresponding qualifier value can change over time. Value qualifiers are only applicable to elements with *kind="Instance"*.
ConceptQualifier	Qualifies the semantic definition (*HasSemantics/semanticId*) the element is referring to; the corresponding qualifier value is static.
TemplateQualifier	Qualifies the elements within a specific submodel on concept level; the corresponding qualifier value is static. *Note:* template qualifiers are only applicable to elements with kind="Template". See constraint AASd-129.

Figure 8: Qualifier Kind [3]

8.2 Adaptation of the AAS Interfaces

This chapter proposes a technology neutral specification of operations of the AAS interfaces to achieve AAS discovery functionality. These enhancements can be mapped to various technologies, such as the existing REST API specified in [9].

The interface specification consists of interface operations, their functionality as well as input (I) and output (O) parameters as listed in Table 1.

For the sake of simplicity, the following assumptions are made:

- All environmental identifiers are omitted at this point, e.g., ID of registries and repositories. If necessary, such ID must be added when mapping the interface operations to API operations.

- Categories (CAT) must support to have an unlimited number of values (VAL). The way to implement this (e.g., as a variable supporting a set of values or a field of CAT and VAL pairs with identical CAT) is not discussed here.

- Discovery rules (RULE) are expressed by a selection of finitely many discovery criteria (CAT and VAL pairs) and their logical relations.
  Examples of logical relations are Boolean equations of discovery criteria applying e.g., the Boolean operations "EQUALS", "AND", "OR", "IN", "NOT".

- Discovery operations (DISC) shall return lists of identifiers of AAS (AASID) or submodels (SMID) conformant to the configuration of the Restriction of use category e.g., if an AAS user application has no access rights and no usage rights of a submodel then the SMID of the submodel shall not be part of the list of submodels returned by DISC.

- These identifiers shall be used by AAS user applications to lookup for AAS and submodels.

Note: Leaving RULE empty means not to limit the DISC methods by a discovery rule.

This would mean that, e.g., method DISC AAS SM would return LSMID containing all SMIDs of submodels listed in AAS with id AASID and AAS user application has access rights and usage rights for.

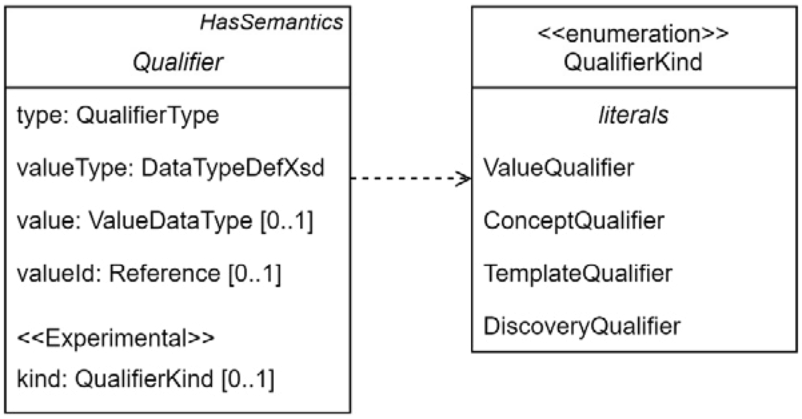

Figure 9: Adapted Metamodel of Qualifiers

Operation	Parameter	I/O	Functionality
ADD CAT AAS	AASID CAT VAL Success	I I I O	The operation adds the category CAT with value VAL to the AAS AASID It returns Success. *Note:* CAT can have the value "Context", "Classification" or "Restriction of use"
ADD CAT SM	AASID SMID CAT VAL Success	I I I I O	The operation adds the category CAT with value VAL to the submodel SMID listed in AAS AASID. It returns Success. *Note:* CAT can have the value "Context", "Classification" or "Restriction of use"
RMV CAT AAS	AASID CAT Success	I I I O	The operation removes the category CAT and all of its values VAL from AAS AASID It returns Success. *Note:* CAT can have the value "Context", "Classification" or "Restriction of use"
RMV CAT SM	AASID SMID CAT Success	I I I O	The operation removes the category CAT and all of its values VAL from the submodel SMID listed AAS AASID. It returns Success. *Note:* CAT can have the value "Context", "Classification" or "Restriction of use"
DISC AAS	RULE LAASID Success	I O O	The operation searches for AAS which comply with discovery rule RULE. It returns Success. It returns a list of AAS ID LAASID. LAASID contains all AASIDs of AAS compliant with RULE.
DISC SM	RULE LSMID Success	I O O	The operation searches within for submodels which comply with disco-very rule RULE. It returns Success. It returns a list of SM ID LSMID. LSMID contains all SMIDs of submodels compliant with discovery rule RULE of all AAS accessible in a defined environment, e.g., in an AAS repo-sitory.
DISC AAS SM	AASID RULE LSMID Success	I I O O	The operation searches for submodels which comply with discovery rule RULE. It returns Success. It returns LSMID. LSMID contains all SMIDs of submodels compliant with discovery rule RULE and listed in AAS AASID.

Figure 9: Adapted Metamodel of Qualifiers

8.3 Related work

8.3.1 The Interoperability Vector - a Standardized Approach to express Values of the Classification Category

The interoperability vector is a self-declaration for AAS and contains information about three main interoperability aspects that are transport, syntax and semantic. Transport provides information about the used transport mechanisms for assets, e.g., http/https, MQTT, OPC UA and security. Syntax contains the implemented serialization schemas like JSON, XML, RDF etc. and semantic describes the used AAS metamodel elements, both with version information of the used AAS specifications. [12]

The interoperability vector can be implemented as a submodel. Classification services or restriction of use services like in [11] can be used for AAS to search for interoperable AAS with discovery mechanisms.

The goal of the interoperability vector, which is described in [12], is

- to provide means to support interoperability evaluations during run time of AAS applications and their infrastructure
- to investigate during run time if there are dependencies between defined potential interoperability levels and application cooperation
- to identify which AAS related specifications are necessary for interoperable interactions
- and to investigate the scope of the particular test subset of an AAS.

8.3.2 The Information model for capabilities, skills and services – a standardized approach to implement according to a defined value for classification

The Platform I4.0 published a discussion paper about the information model for capabilities, skills, and services (CSS model) [13]. With the model different use cases for the CSS model like assessing the principle producibility of a certain product as a production planer, to find production services via a marketplace or to schedule production executions are described. The model was developed independently of implementation technologies, so that beside AAS standards like Module Type Package, OPC UA, AML or semantic web technologies can be used.

In the case of using AAS, most of the described uses cases in [13] need mechanisms to find assets with certain capabilities, skills, or services in a large pool of AAS. For this, it is not wise to browse every AAS for the searched element for every request rather than have discovery mechanisms. To find, for example, machines with a certain capability it is useful to classify the machine capabilities, e.g., with the classification service of [11] and to find the right machines by using the discovery described in this paper.

8.3.3 The generation of Digital Twins for Exchanging Information via Application Programming Interfaces

In the easyASSET research project, various northbound and southbound AAS connections have been analyzed and developed. The generative approach of AAS-CORE is used and further developed [14]. In the rapidly evolving landscape of Industry 4.0 and digital twins, generative AAS is a fundamental building block. This design and development is presented as an innovative approach to automatically generate an Application Programming Interface (API) for AAS based on a formalized specification [15]. Until now, the convention has been to manually specify the API for AAS by tedious translation of the abstract specification published in book form. We propose instead to formalize the abstract specification and automatically translate it into an API. In this way, we streamline the development process, ensure efficiency and minimize the risk of errors in the representation of digital twins, which is otherwise inherent in the manual procedure [16]. Our proposed approach places great emphasis on automating the development of AAS APIs, using the common and widely used YAML/OpenAPI as a serialization format due to its clarity and simplicity. In addition, the paper examines other interface descriptions such as Protobuf, SOAP, GraphQL and WSDL, highlighting the importance of well-defined interfaces for efficient AAS API usage. Our methodology ensures scalability and flexibility, particularly in terms of SDK generation, where different AAS interfaces are generated to support smooth integration within the ever-changing Industry 4.0 landscape. This comprehensive approach facilitates the entire digital twin development lifecycle, from API generation to SDK development, to ensure robust and adaptable solutions, independent of specific serialization formats.

IX. References

[1] Plattform Industrie 4.0: Usage View of the Asset Administration Shell. Discussion Paper. https://www.plattform-i40.de/PI4.0/Redaktion/DE/Downloads/Publikation/2019-usage-view-asset-administration-shell.html

[2] M. Stolze, C. Diedrich, und M. Riedl: "Automatisierte Dokumentationspflege bei Gerätetausch durch Verwaltungsschalen - Entwurf von Teilmodellen zur Beschreibung von Verwaltungsschalen-Strukturen", atp magazin, Nr. 10/2023, 2023, October.

[3] Industrial Digital Twin Association, Hrsg.: "Asset Administration Shell Specification - Part 1: Metamodel Schema".2023, March.

[4] Industrial Digital Twin Association, Hrsg.: "IDTA 02011-1-0 - Hierarchical Structures enabling Bills of Material".2023, April.

[5] H. Schweizer, N. Braunisch, R. Alt, K. Schmitz, und M. Wollschlaeger: "Prozesskomposition in verteilten Automatisierungssystemen: Orchestrierung und Choreografie von Inbetriebnahme-Prozessen in verteilten Automatisierungssystemen mittels proaktiver Verwaltungsschalen und B2MML", - Autom., Bd. 69, S. 242–255, 2021 March, doi: 10.1515/auto-2020-0118.

[6] R. Alt, H. Schweizer, M. Wollschlaeger, und K. Schmitz: "Interoperable information model of a pneumatic handling system for plug-and-produce", 2020, June. doi: 10.25368/2020.65.

[7] H. Schweizer, N. Braunisch, und M. Wollschlaeger: "Prozessorchestrierung in verteilten Automatisierungssystemen", 2020, February.

[8] IEC: "IEC 63278-1:2023 - Asset Administration Shell for industrial applications - Part 1: Asset Administration Shell structure".2023, December.

[9] Industrial Digital Twin Association, Hrsg.: "Asset Administration Shell Specification - Part 2: Application Programming Interfaces". 2023, June.

[10] Andrei Ciortea, Simon Mayer, Florian Michahelles: "Repurposing Manufacturing Lines on the Fly with Multiagent Systems for the Web of Things", AAMAS 2018, July 10-15, 2018, Stockholm, Sweden

[11] Plattform Industrie 4.0: Functional View of the Asset Administration Shell in an Industrie4.0 system environment. Discussion Paper https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/Functional-View.html

[12] Christian Diedrich: Concept Interoperability, AG "Referenzarchitekturen und Standards" Plattform I4.0, 2022, August

[13] Christian Diedrich et al.: "Information Model for Capabilities, Skills & Services - Definition of terminology and proposal for a technology-independent information model for capabilities and skills in flexible manufacturing", Plattform Industrie 4.0, Discussion Paper, 2022, October

[14] N. Braunisch, M. Ristin-Kaufmann, R. Lehmann, M. Wollschlaeger, und H. W. van de Venn: "Empowering Industry 4.0 with Generative and Model-Driven SDK Development", in Industrial Electronics Society (IECON), 2023, S. 1–6. doi: 10.1109/IECON51785.2023.10312302.

[15] N. Braunisch, M. Ristin, R. Lehmann, U. Schmidt, M. Wollschlaeger, und H. W. van de Venn: "Generation of Digital Twins for Exchanging Information via Application Programming Interfaces", in Industrial Cyber Physical Systems (ICPS), 2024.

[16] N. Braunisch, M. Ristin, R. Lehmann, M. Wollschlaeger, und H. W. van de Venn: "Generation of Digital Twins for Information Exchange Between Partners in the Industrie 4.0 Value Chain", in Industrial Informatics (INDIN), 2023.

AUTHORS
Alexander Gordt (objective partner AG), Anja Simon (Labs Network Industrie 4.0 e.V.), Bernd Vojanec (WITTENSTEIN SE), Detlef Tenhagen (HARTING Stiftung & Co. KG), Dominik Rohrmus (Labs Network Industrie 4.0 e.V.), Guido Stephan (Siemens AG), Igor Gramaev (RWTH Aachen), Jens Vialkowitsch (Robert Bosch GmbH), Melanie Stolze (ifak Institut für Automation und Kommunikation e.V.), Nico Braunisch (Technische Universität Dresden), Sten Grüner (ABB AG Forschungszentrum Deutschland), Torben Miny (RWTH Aachen University)

THANKS FOR THE SUPPORT
Kai Garrels (ABB Stotz-Kontakt GmbH), Sebastian Bader (SAP SE)

THIS PUBLICATION IS A RESULT OF THE TASK FORCE “DISCOVERY” OF THE ASSET ADMINISTRATION SHELL SUBWORKING GROUP OF THE WORKING GROUP 1 “REFERENCE ARCHITECTURES, STANDARDS AND NORMS” OF PLATTFORM INDUSTRIE 4.0.