Harmonization of heterogeneous asset administration shells

Nikolaos-Stefanos Koutrakis, Varun Gowtham, Wenzel Baron Pilar von Pilchau, Thomas Josef Jung, Julian Polte, Jörg Hähner, Marius-Iulian Corici, Thomas Magedanz, Eckart Uhlmann

Angaben zur Veröffentlichung / Publication details:

Koutrakis, Nikolaos-Stefanos, Varun Gowtham, Wenzel Baron Pilar von Pilchau, Thomas Josef Jung, Julian Polte, Jörg Hähner, Marius-Iulian Corici, Thomas Magedanz, and Eckart Uhlmann. 2022. "Harmonization of heterogeneous asset administration shells." *Procedia CIRP* 107: 95–100. https://doi.org/10.1016/j.procir.2022.04.016.

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 107 (2022) 95–100

www.elsevier.com/locate/procedia

55th CIRP Conference on Manufacturing Systems

Harmonization of Heterogeneous Asset Administration Shells

Nikolaos-Stefanos Koutrakis[a], Varun Gowtham[b], Wenzel Baron Pilar von Pilchau[c,*], Thomas Josef Jung[d], Julian Polte[a], Jörg Hähner[c], Marius-Iulian Corici[b], Thomas Magedanz[b], Eckart Uhlmann[a,e]

[a]*Fraunhofer Institute for Production Systems and Design Technology (IPK), Pascalstr. 8-9, 10587 Berlin, Germany*
[b]*Fraunhofer FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany*
[c]*University of Augsburg, Am Technologiezentrum 8, 86159 Augsburg, Germany*
[d]*Robert Bosch GmbH, Robert-Bosch-Campus 1, 71272 Renningen, Germany*
[e]*Technische Universität Berlin, Institute for Machine Tools and Factory Management (IWF), Pascalstr. 8-9, 10587 Berlin, Germany*

* Corresponding author. Tel.: +49 821 598 69263. *E-mail address:* wenzel.pilar-von-pilchau@uni-a.de

Abstract

In the era of digital transformation of the manufacturing and process industry, heterogeneity of assets is one of the most challenging issues towards digitally integrating components within the Industrial Internet of Things. In this context every participant relies on its proprietary digitalization approach envisioned by Plattform Industrie 4.0. To consolidate these heterogeneous data exchange interfaces, e.g communication protocols, data formats etc., an intermediate step of harmonization is required. Our contribution provides an architecture based on the Asset Administration Shell standard to bring heterogeneous Cyber-Physical-Systems together. We illustrate the functionality through an abstract use-case.

Keywords: Asset Administration Shell; Digital Twin; Virtual Sensor; Sensor Networks; IIoT; I4.0; MQTT; OPC UA

1. Motivation

In the course of digitalization the manufacturing industry is establishing digital representatives of physical assets such as sensors and all kinds of machines. Suitable concepts to do so are Digital Twins (DTs) [1] and Asset Administration Shells (AASs) [1] that allow to monitor and even control their physical counterparts. In the end, this digital transformation results in a digital factory with all its assets able to communicate with each other and the cloud. This, also known as the Industrial Internet of Things (IIoT) [2], follows the idea of the Plattform Industrie 4.0 (PI4.0)[1], which already delivers some well defined interfaces and standards to enable a unified design.

In a broader scope, several of these digital factories will work together and share all their data with each other. Here some problems arise, like privacy, data security, and legal issues [3]. Another important challenge that needs to be addressed is the combination of different underlying digitalization solutions that are implemented by the different stakeholders. The situation will be as follows: every data provider, e.g. factories, is implementing and is relying on its own proprietary digitalization approach using different data formats (e.g. binary, text) and interfaces (e.g. OPC UA, HTTP, MQTT). Nevertheless, to enable the cooperation of the different parties, an intermediate step is needed to harmonize the different data outputs and in the end offer a standardized and agreed upon format everyone can use.

This paper focuses particularly on the digital representation of physical assets and harmonization of data interfaces. The harmonization of data interfaces is an incremental process where the data from a physical device is abstracted from a set of possibly proprietary approaches to a generalized data format and uniform interface. We present a solution for this harmonization task in the form of an architecture based on the AAS standard that aims for the integration of heterogeneous Cyber-Physical-Systems. To do so we identified requirements that arise in the context of harmonization and then present a novel solution for each of them. We present both, an exemplary implementation

10.1016/j.procir.2022.04.016

in the project FAMOUS[2], and a general overview utilizing abstraction layers incorporated in our architecture. Different protocols can be used from data providers and we present the mapping of an AAS on two of them (OPC UA and MQTT) in detail and compare them in the end.

2. Related Work

The Industrie 4.0 (I4.0) movement aims to shift the existing processes into a digital domain, where the assets of an industry and the accompanying processes are digitized. This vision is a multi-disciplinary challenge that demands careful consideration of both, chosen technologies and approaches [4]. The success of the vision depends partly on amalgamation of the diverse domains of Cyber-Physical-Production-Systems [5] and use of suitable technologies [6]. It requires the mapping of a heterogeneous set of features collected from participating domains to a digital domain [7]. Accounting to the vast variety of aspects of physical devices, e.g. their communication interfaces and manufacturing practices, to name a few, there is an urgent requirement for a strategy for harmonization. One way of achieving harmonization is through lifting the abstraction level of assets and making the automation functions modular [8], leading the way for distributed DTs. To this end, a service oriented architecture for automation is possible [9]. The research community advocates AAS as one of the efficient methods in leading harmonization of heterogeneous environments [10]. In a hierarchical system, the level of abstraction of an entity is important. It adds a layer of morphism around the entity such that, at the given level in the hierarchy, the morphism exposes relevant details to the participating entities and hides non-essential details. For example, considering a sensor device as an entity, it is essential for the operator who physically controls the device (a lower level of abstraction) to utilize the information included in the sensor's data sheet. While an application interested in receiving the sensor values (at the higher level of abstraction), is interested about the interface through which data is exchanged with the sensor [11]. The AAS incorporates models [12] and makes it easier for services residing at higher levels of abstraction to access data and manage assets located in heterogeneous lower abstraction environments [13].

3. Requirements to achieve harmonization

To achieve harmonization in a way that meets the definition of the previous section, we identified several requirements that need to be satisfied. For the purposes of discussion we identified two abstract uses cases that are illustrated in Figure 1:

1. **Bottom-up:** Assets (e.g. sensors) are located in the physical domain on the lowest abstraction level. Owned by the data providers they are accompanied with a data sheet that includes all the relevant information of the asset (e.g. measured unit, measurement frequency, etc.) and an offered (proprietary) interface. The providers want to offer their data to the data consumers and therefore need to lift the abstraction level. This can be realised by transforming a physical asset into a digital representation (a DT) to achieve interoperability and promote usability.
2. **Top-down:** On the highest level of abstraction the service providers want to consume the data from the physical domain in a way that is simplified as much as possible and reduced to the relevant parts. In consequence a data provider knows the level of abstraction that is needed for its service to operate sufficiently.

In both use-cases arise the same two main questions:

1. **Q1:** How can this be achieved in general?
2. **Q2:** How can interoperability be achieved between providers and consumers?

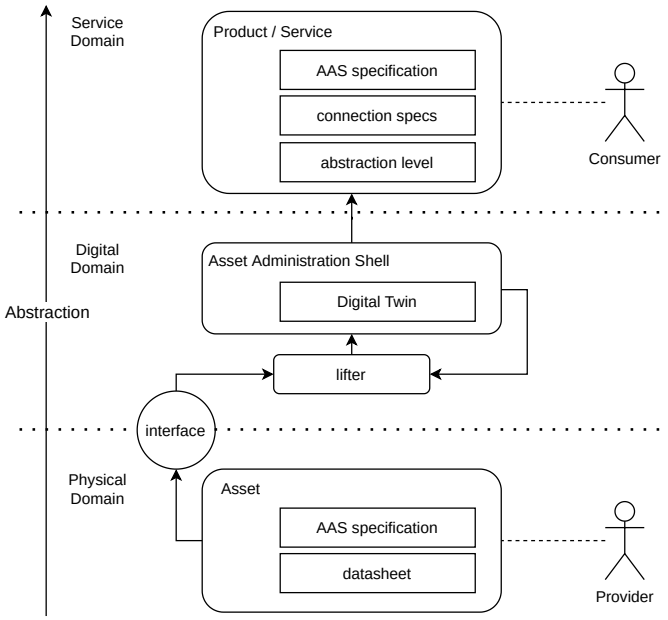

Fig. 1. A graphical illustration of the abstraction level elevation.

Figure 1 illustrates an abstract solution for Q1 in the following way: A physical asset needs to also provide an *AAS specification* as a composition of AAS meta-models [14] that includes all the required information of how to represent the asset in a digital form. This specification is a blueprint for the digital representation. In the digital domain, the AAS instance that is responsible for the harmonization process receives the AAS specification and uses this information to create a DT of the physical asset. The data exchanged from the asset uses its own (proprietary) interface and is received by a so-called lifter that is responsible for elevating the abstraction level. The lifted data is then received by the AAS instance and can be received from other digital assets if the current abstraction level is high enough. If this is not the case, the data can be sent through a lifter again. This procedure is repeated until the abstraction

[2] famous-project.eu

level meets the requirements. The data consumers in the service domain need to know, additionally to the required abstraction level, how they can connect themselves to the AAS instance and how the data will be provided (IP address and protocol). Additionally they can receive the AAS specification of the asset they want to consume from and with this information they are able to consume only the specific data they need for their services. Based on this abstract architecture we identified the following requirements:

1. **Standardized interfaces:** To enable the data exchange between assets (physical/virtual assets and services) it is required, that the data offered by the providers can be consumed through a standardized interface. Assets may provide their data through any (proprietary) interface, but after the harmonization only one standard interface should be in use.
2. **Standardized data formats:** Another important requirement is the usage of a standardized data format. Again, assets may deliver their data in any form, but after the harmonization the data should be provided in a standardized format for the consumers.
3. **Interoperability:** Relating to Q2, a requirement is the interoperability between the involved assets on different abstraction levels. The participating entities should therefore be able to communicate and understand the exchanged data formats. This point includes the two aforementioned requirements but does not stop there.
4. **Abstraction:** As already stated above, the consumers do not need all the data that comes with a physical asset. They require specific information like for example the acceleration measurement of a sensor that measures acceleration, level and speed. The sensor in the physical domain also provides data about measurement frequency etc. and therefore an elevation of the abstraction level is required, so that in the end the data meets the needs of the consuming service. The abstraction requirement also includes the abstraction of interfaces and data formats. Generally speaking, implementation details should be hidden by abstraction.
5. **Easy to use:** A more general requirement is the ability to make use of the harmonization in a way that is as straight forward as possible. Most preferred would be a simple plug-and-play solution, but if this is not possible, then the process should be assisted as much as possible. Next to a simplified usage, the data output in the end should be understandable and reduced to the important information. All of this can be achieved by a good usage of abstraction.

4. Towards Harmonization

4.1. Generic Architecture

Figure 2 shows a conceptual architecture of an envisioned AAS. The blocks in the figure are color coded to align to a specific layer in the Reference Architectural Model Industrie 4.0 reference architecture. Starting from the bottom, the physical domain represents physical assets. To operate the asset, the asset's attributes and data interfaces are relevant to a user. Depending on the design and capabilities, an asset has a set of heterogeneous attributes and interfaces. The user employs relevant technology to make the asset functional. For the purpose of modeling the asset in a digital domain, it is sufficient that the user abstracts the interfaces and attributes of the asset, through a combination of meta-model instances provided by the AAS standard. The AAS provides the facility to model the asset through AAS specification in the form of a digital document for example JavaScript Object Notation (JSON), which is used as default format in our discussion in the rest of the paper. The computing node runs an AAS agent capable of reading the JSON and transforming the embedded AAS specification into a digital model in the AAS environment, such that there is a one-to-one mapping between the attributes and interfaces of the physical asset to their digital counterparts in the form of model elements. Furthermore, through the medium of the AAS agent, the same AAS specification in the form of JSON guides the services in the service domain towards accessing the attributes and interfaces of the physical asset. If needed, the AAS interface located in the physical and the service domain translates to and from a proprietary protocol, into a generic standardized protocol of the digital domain. The approach of AAS makes it possible to build assets as modular and re-usable blocks. We can observe the following distinct points:

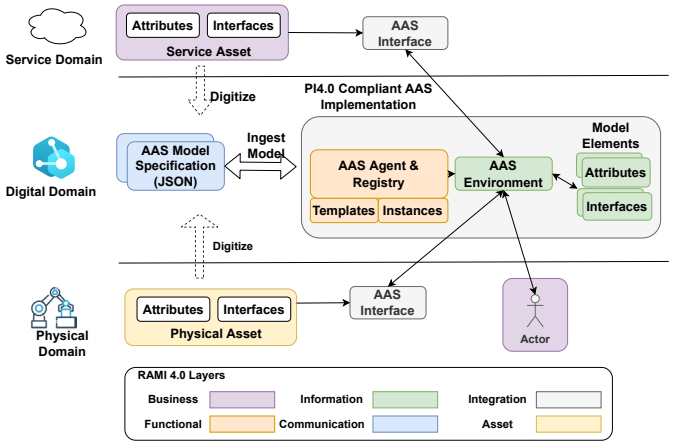

Fig. 2. Conceptual architecture of harmonization with AAS.

- A common AAS specification in the form of JSON elevates the abstraction level of physical assets and makes it reachable at all 3 domains.
- It's sufficient for users who intend to access the physical asset to communicate with the AAS agent in the digital domain with a standardized protocol. This way AAS relieves the user's burden of prerequisite expertise in lower abstraction technologies that physical asset originally uses.

Extending from Figure 2 we now focus on modeling the asset's attributes and data interfaces. The set of meta-models provided by the AAS is extensive. The general layout of an AAS

specification has 4 top level objects, where each top level object holds objects of a specific type. The field *idShort* in each object holds a unique identifier visible and referable under a given namespace.

- **assetAdministrationShells**: holds objects of type *assetAdministrationShell*. Each *assetAdministrationShell* references one or more objects of type *submodel* identified by *idShort*. Each object of type *assetAdministrationShell* represents an AAS as a kind of a re-usable *template* or an *instance*.
- **submodels**: holds objects of type *submodel*. Each *submodel* delineates an asset in the form of one or more hierarchically organized objects of type *submodelElement* Each *submodelElement* models a specific aspect of the asset. Both *submodel* and *submodelElement* object types can hold representations of *kind*, either template or instance. Together, they take the responsibility of modelling the asset.
- **assets**: holds objects of type *asset*. Each *asset* can be *kind*, either type or instance. Each *asset* object models certain necessary non-functional features of the asset.
- **conceptDescriptions**: holds objects of type *conceptDescription*. Each *conceptDescription* provides descriptions of extended concepts referred by other object types in the AAS to enhance the representation vocabulary.

Figure 3 gives a graphical representation of an AAS description of a hypothetical accelerometer sensor, which would be given as a JSON or XML document in reality. Starting from the outermost rectangle, *assetAdministrationShells[0]*, exposes the object of type *assetAdministrationShell* that represents an instance of an AAS with *idShort* = ”Sensor_A”. This AAS further refers to an instance of type *submodel* with *idShort* = ”Gyro”. Furthermore, the referred *submodel* holds the objects of type *submodelElement* identified by *idShort* = ”x” which holds *modelType* = ”Property” and ”event_x” which holds the *modelType* = ”BasicEvent”. ”event_x” holds the required details about the data interface used to receive events from the property ”x”. It is used as a common example in the following subsections.

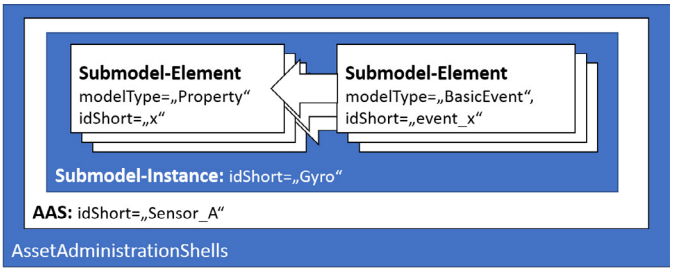

Fig. 3. Exemplary AAS description

In the following subsections we show, how abstract AAS descriptions are mapped to the concrete network protocols OPC UA and MQTT for interoperability in the different layers apart from offering an implementation specific data interface.

4.2. Asset Administration Shell via OPC UA: Mapping AAS elements to nodes

OPC UA is a Machine To Machine (M2M) communication protocol. The AAS standard provides rules for mapping the AAS specification syntactically onto OPC UA nodes. A comprehensive discussion on the rules of mapping between AAS and OPC UA is given in [15, 16]. The hierarchical arrangement of OPC UA nodes under the node *AASRoot* holds all other AAS nodes. Each node’s unique *browseName* derived from *idShort* of the encapsulated AAS object uniquely identifies the node under a given namespace. The node’s *displayNode* derived from *idShort* of the AAS provides a human readable name to the node. Each node thus defined, has a specific type definition depending on the object type of the *AAS* node refers to. Referring to Figure 3, the following points provide an overview of our mapping implementation:

- General hierarchy: The OPC UA server works in tandem with an AAS agent that holds the digital models. Translation rules map digital models present in the AAS agent on the OPC UA server. The *AASRoot* is the top-level node that holds all other AAS nodes. Each unique AAS node, an analog of the *assetAdinistrationShell* object in JSON, holds the encapsulated *submodel* in yet another unique node. The node of the *submodel* then holds the nodes of *submodelElements* of *Property x* and *BasicEvent event_x*.
- Relationship: The node of *BasicEvent event_x* references the *Property x*.
- Data Exchange: Among other details, the node of *event_x* holds a variable node responsible in holding the value of data the clients can write to or read from. The clients that write to the variable node lift the abstraction level of data received from assets to a standardized interface of OPC UA such that any client subscribed to the variable node can receive the abstracted data without worrying about the underlying topology or implementation details of the asset.

4.3. Asset Administration Shell via MQTT: Mapping AAS elements to topics

MQTT is a lean Pubsub communication protocol well established in the IIoT-world. Further details are given in [17].

Basic mapping-approach is: Each individual AAS occupies a root topic in the form `<namespace>/<aasIdShort>`. The *namespace*-element is a generic item used to group a set of AASs together, e.g. a company’s name. The AAS’ short ID in the second position is used to identify the concrete AAS instance. As short ID must contain only ”letters, digits and underscores” [14], this fits the MQTT topic constraints [17]. The AAS’ elements are published at sub-topics below this root topic.

4.3.1. AAS self description

Accordingly the AAS’ self description in the form of a JSON-document is published at the topic: `<namespace>/<aasIdShort>/aas`. The third element *aas* was

chosen in analogy to the REST interface described in [18]. Self description is published as *retained* message, so that a new subscriber receives it immediately when subscribing.

4.3.2. Submodel properties

In an AAS specification, a submodel property consists of the property's meta data and the property's value. In BaSyx [18], when offering AAS via HTTP/REST, a client can choose exactly the section of an AAS' description it is interested in, e.g. by querying `GET <host>/<aasIdShort>/submodels/<submodelIdShort>/submodelElements/<elementIdShort>/value`: Via the URI the client can choose between a property value, a full property description or even a complete submodel.

Transferring this concept directly to MQTT or similar would result in a huge number of topics: Each single element of an AAS specification had its own topic. Even worse: When updating just one Property value, in addition to the value's topic's message another separate message to every topic corresponding to each parent element would have to be sent.

A lean solution inspired by [19] would be: Leave out the *value* attribute in the AAS description. Instead the value is sent at an implicitly defined topic in the form: `../submodelElements/<elementIdShort>/value`. This might align with the idea of the *ValueOnly*-serialization [20], but does not seem to be covered by the standard as of today. This is why we propose the usage of AAS-Events:

```
{
  "idShort": "event_x",
  "observed": {
    "keys": [{
        "type": 25,
        "idType": 2,
        "value": "<urn of submodel>",
        "local": true
      }, {
        "type": 15,
        "idType": 4,
        "value": "Prop_X",
        "local": true
      }]},
  "modelType": {"name": "BasicEvent"},
  "messageTopic": "<namespace>/Sensor_A/submodels
      /Gyro/submodelElements/event_Prop_X",
  "kind": 0}
```

The value field in the AAS specification is left out. In addition to the property of concern the submodel contains a model element of type "BasicEvent" [20], referring to the property using the "observed"-attribute. It names the concrete MQTT topic for the value updates in its *messageTopic*-attribute. Note: Although in the example the event's name relates to the property's name, an arbitrary name is possible.

4.4. Common data format for OPC UA and MQTT

Due to large overlap between our requirements and the JSON-based Production Performance Management Protocol (PPMP) of the Eclipse Unide project [21], we chose it as data exchange format. In detail we use the Measurement Message, Version 3. For our use-case we decided to make two elements mandatory, which are optional in the standard: Each TimeMeasurement must contain a Context object and: The Context object must contain at least a non-empty value in the "unit"-field, as it is used as minimum semantic description of the data.

5. Evaluation

This section provides a general comparison between implementing AAS through OPC UA and MQTT. Keeping the JSON representation of Figure 3 as a common point, we try to superimpose the implementations of MQTT and OPC UA to compare. The section first addresses the experiences relating to each requirement listed in Section 3 and then makes a general comment on the use-cases.

5.1. Experiences and comparison: AAS on OPC UA and MQTT

1. **Standardized interfaces**: Both MQTT and OPC UA elevate the proprietary data interfaces of the assets to a standardized protocol. The clients connecting to the implementation may use a plug and play approach to receive data.
2. **Standardized data formats**: Detailed in Section 4.4, both implementations act as an intermediate medium between the producer and consumer of data. The agreed contract of the data format [21] between the producer and consumer is kept intact by the implementations.
3. **Interoperability**: The message exchange protocols of MQTT and OPC UA are interoperable thanks to the standardization and also the ecosystem of products and services available around them. OPC UA is a de-facto standard for M2M communication in industries. MQTT is widely prevalent in the field of IIoT.
4. **Abstraction**: OPC UA and MQTT focus different scenarios. While OPC UA was designed to be the all-encompassing solution for IIoT, MQTT was developed as a simple, lean, general-purpose solution. Correspondingly one will make the choice depending on the operational scenario. It made sense to us to use MQTT in the physical layer, supporting low resource devices and highly dynamic network environments. Analogously, it made sense to use the client-server-based OPC UA in the service layer, as it offers all features services require to search and query the specific portion of data they need.
5. **Easy to use**: The AAS implementation benefits users by providing a standardized method of modeling their assets through the meta-models provided by the AAS standard of PI4.0. OPC UA and MQTT are two different technologies that can hold AAS representations. Together with interoperability and abstraction as outlined earlier, they can address the challenge of incorporating digital models for automation. The discussion further reinforces the vision of a common AAS specification that can be reused between physical, digital and service domains as shown in Figure 2.

5.2. Benefits to the use-case

To come back to the two abstract use-cases from section 3, we have demonstrated how they benefit from the presented harmonization approach. In the bottom-up case the data providers are now able to abstract their assets through the meta-model that is provided by the AAS standard. Consequently, they communicate interfaces and attributes to higher abstraction levels in a clear and easy manner. On the other side, the top-down perspective benefits from the elevated abstraction level to standardized interfaces which enables for the users the possibility to focus on the development of their applications that can then be used by plug-and-play.

6. Summary and Future work

Digitalization will be an important part of modern industry and concepts like AAS and DT represent state-of-the-art solutions for this task. To combine several proprietary digitalizations on different levels (physical, digital and service level) a harmonization approach is necessary that ensures interoperability. We presented a definition of what can be understood as harmonization and provided some examples for related work in this domain. With a definition on hand we were able to identify some major requirements that need to be fulfilled in order to deliver a satisfying harmonization approach. To illustrate these requirements we used two abstract use-cases in an exemplary digitalization architecture. Then we presented a more concrete architecture developed in the FAMOUS project that can be mapped on different communication protocols. We investigated this mapping in more detail for MQTT and OPC UA and compared them. Using superimposition we were able to point out the benefits and disadvantages of the two interfaces and concluded the comparison. The presented MQTT mapping only handles submodel elements of type Property. Mapping rules for other submodel element types have to be developed in the future. Also, the AAS standard is an evolving thing and at the moment of writing the so-called *events* are not covered yet. As soon as this is ready, the presented AAS mappings need to be adjusted.

Acknowledgements

Research for this paper is funded by the Federal Ministry of Education and Research (BMBF) project FAMOUS (01IS18078). https://famous-project.eu/

References

[1] Constantin Wagner, Julian Grothoff, Ulrich Epple, Rainer Drath, Somayeh Malakuti, Sten Grüner, Michael Hoffmeister, and Patrick Zimermann. The role of the industry 4.0 asset administration shell and the digital twin during the life cycle of a plant. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2017.

[2] Hugh Boyes, Bil Hallaq, Joe Cunningham, and Tim Watson. The industrial internet of things (iiot): An analysis framework. *Computers in Industry*, 101:1–12, 2018.

[3] Matthias Eckhart and Andreas Ekelhart. *Digital Twins for Cyber-Physical Systems Security: State of the Art and Outlook*, pages 383–412. Springer International Publishing, Cham, 2019.

[4] Martin Hoffmann, Somayeh Malakuti, Sten Grüner, Soeren Finster, Jörg Gebhardt, Ruomu Tan, Thorsten Schindler, and Thomas Gamer. Developing industrial cps: a multi-disciplinary challenge. *Sensors*, 21(6):1991, 2021.

[5] Stamatis Karnouskos, Luis Ribeiro, Paulo Leitao, Arndt Luder, and Birgit Vogel-Heuser. Key directions for industrial agent based cyber-physical production systems. In *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, page nil, 5 2019.

[6] Xun Ye and Seung Ho Hong. An automationml/opc ua-based industry 4.0 solution for a manufacturing system. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, page nil, 9 2018.

[7] Santiago Soler Perez Olaya and Martin Wollschlaeger. The role of comprehensive function models in the management of heterogeneous industrial networks. In *2019 15th IEEE International Workshop on Factory Communication Systems (WFCS)*, page nil, 5 2019.

[8] Juergen Jasperneite, Thilo Sauter, and Martin Wollschlaeger. Why we need automation models: Handling complexity in industry 4.0 and the internet of things. *IEEE Industrial Electronics Magazine*, 14(1):29–40, 2020.

[9] Thomas Kuhn, Frank Schnicke, and Pablo Oliveira Antonino. Service-based architectures in production systems: Challenges, solutions & experiences. In *2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K)*, page nil, 12 2020.

[10] Miguel A. Inigo, Alain Porto, Blanca Kremer, Alain Perez, Felix Larrinaga, and Javier Cuenca. Towards an asset administration shell scenario: a use case for interoperability and standardization in industry 4.0. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, page nil, 4 2020.

[11] Fabian Burzlaff and Christian Bartelt. I4.0-device integration: A qualitative analysis of methods and technologies utilized by system integrators: Implications for enginering future industrial internet of things system. In *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 27–34, 2018.

[12] Salvatore Cavalieri and Marco Giuseppe Salafia. A model for predictive maintenance based on asset administration shell. *Sensors*, 20(21):6028, 2020.

[13] Andreas Deuter and Sebastian Imort. Product lifecycle management with the asset administration shell. *Computers*, 10(7):84, 2021.

[14] Plattform Industrie 4.0. Details of the Asset Administration Shell - Part 1: The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC01). https://bit.ly/3phSUYB, November 2020. Accessed: 2021-11-15.

[15] Salvatore Cavalieri and Marco Giuseppe Salafia. Insights into mapping solutions based on opc ua information model applied to the industry 4.0 asset administration shell. *Computers*, 9(2):28, 2020.

[16] Cornelis Bouter, Monireh Pourjafarian, Leon Simar, and Robert Wilterdink. Towards a comprehensive methodology for modelling submodels in the industry 4.0 asset administration shell. In *2021 IEEE 23rd Conference on Business Informatics (CBI)*, 9 2021.

[17] Organization for the Advancement of Structured Information Standards. MQTT Version 5.0. https://bit.ly/37OnnWs, 2019. Accessed: 2021-11-17.

[18] Constantin Ziesche. BaSyx Asset Administration Shell HTTP REST-API. https://bit.ly/3C7KOIV, 2021. Accessed: 2021-11-15.

[19] Daniel Ewert, Thomas Jung, Timur Tasci, and Thomas Stiedl. *Assets2036 – Lightweight Implementation of the Asset Administration Shell Concept for Practical Use and Easy Adaptation*. Springer, Berlin, 2021.

[20] Plattform Industrie 4.0. Details of the Asset Administration Shell Part 2: – Interoperability at Runtime – Exchanging Information via Application Programming Interfaces. https://bit.ly/3phSUYB, November 2020. Accessed: 2021-11-15.

[21] The Eclipse Foundation. Production Performance Management Protocol (PPMP). https://www.eclipse.org/unide/, 2021. Accessed: 2021-11-15.