Future Generation Computer Systems 113 (2020) 94–105

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

File- and API-based interoperability of digital twins by model transformation: An IIoT case study using asset administration shell

Marie Platenius-Mohr [a,*,1], Somayeh Malakuti [a,1], Sten Grüner [a,1], Johannes Schmitt [a,1], Thomas Goldschmidt [b]

[a] ABB Corporate Research Center Germany, Ladenburg, Germany
[b] ABB Digital, Ladenburg, Germany

ARTICLE INFO

Article history:
Received 23 January 2020
Received in revised form 29 June 2020
Accepted 4 July 2020
Available online 4 July 2020

Keywords:
Industrial Internet of Things
Internet of Things
Digital twin
Information modeling
Interoperability
Model transformation
Asset administration shell
Industry 4.0

ABSTRACT

By providing a consolidated access point to information of devices and systems, digital twins enable new, data-intensive use cases for the industrial internet of things (IIoT) like predictive plant and product design. A digital twin is a digital representation of an entity, which is sufficient to meet the requirements of a set of use cases. However, the lack of interoperability between the digital twins of different companies hinders use cases that require information exchange between different organizations. Achieving interoperability among digital twins requires transforming the included information to other formats. In this paper, we present requirements as well as a solution to enable interoperable digital twins by flexibly transforming their information models. The solution enables file- and API-based information exchange of comprehensive information models in a bidirectional way. We illustrate how we applied our solution to a real-world application example in an industrial context by transforming ABB Ability™ digital twins to the asset administration shell format. We show that our customizable transformation system paves the way for on-demand interoperability and thereby enables advanced use cases in the IIoT. This includes smart use cases that connect industries as well as artificial intelligence in industry.

1. Introduction

The Industrial Internet of Things (IIoT) is characterized by industrial components interacting with each other to become systems fulfilling complex industrial tasks, e.g., production or manufacturing systems. For a couple of years, *digital twins* have been designated as one of the technological trends for IoT as well as IIoT systems [1]. Initially, a digital twin was considered to be a high-fidelity mathematical model of a physical device, which could simulate the device as accurately as possible. As the importance of digitalization has increased for many companies, the definition of digital twin has accordingly evolved [2,3]. Based on the definition provided by Industrial Internet Consortium (IIC), a digital twin is a digital representation (attributes, behavior) of an entity, which is sufficient to meet the requirements of a set of use cases [4].

A digital twin can be seen as an interface to access information from various lifecycle phases of an asset.[2] Information may be provided by different organizations, e.g., manufacturing information of a device is provided by the manufacturer, whereas operational information of a device belongs to its customer. By providing a consolidated access point to various lifecycle information, digital twins enable new, data-intensive use cases to make IIoT systems smarter. Examples are predictive plant and product design, faster field device commissioning, etc. [3]. In particular, smart use cases that connect industries including customers and manufacturers as well as manufacturers among each other are enabled. Furthermore, digital twins can be viewed as an enabler for artificial intelligence in industry, by simplifying access to data and their meaning from various sources and lifecycle phases.

IIoT systems, such as a factory or a plant, consist of several thousand assets, which are often provided by different organizations. Those assets interact with each other to act as a system. To this aim, the involved devices and systems need to be interoperable. Interoperability is defined as “the ability for two or more systems or applications to exchange information and

* Corresponding author.
E-mail addresses: marie.platenius-mohr@de.abb.com (M. Platenius-Mohr), somayeh.malakuti@de.abb.com (S. Malakuti), sten.gruener@de.abb.com (S. Grüner), johannes.o.schmitt@de.abb.com (J. Schmitt), thomas.goldschmidt@de.abb.com (T. Goldschmidt).

[1] The authors were partially supported by German Federal Ministry of Education and Research in the scope of the BaSys 4.0 and BaSys 4.2 projects (01IS16022B and 01IS19022B, respectively).

[2] In this paper, we mainly focus on the information access part, while a digital twin in general could also be used to influence the behavior of an asset.

https://doi.org/10.1016/j.future.2020.07.004
0167-739X/

to mutually use the information that has been exchanged" [5]. Interoperability can be studied from different aspects, but in this work, we focus on interoperability of digital twins for information exchange.

Nowadays, most IIoT platforms and companies offer their proprietary and isolated digital twin solutions with information encoded in disparate ways. Thereby, a lack of interoperability emerges and hinders various use cases that require information exchange between organizations.

Achieving interoperability among digital twins requires transforming information in a peer-to-peer manner or to common (standardized) digital twin format(s) in the case where such format(s) exists at all. Either way, since an IIoT system may include devices and systems from various organizations and since these may change over time, we require means to (semi-)automatically perform the transformation so that the required time and effort to enable interoperability between organizations can be reduced.

In this paper, we study an approach to enable interoperable digital twins and outline a set of requirements that must be fulfilled by an appropriate solution. We propose a solution that enables engineers to flexibly define transformation rules, and apply them to source and target systems. The core of our solution is a customizable mapping model that is first generated and customized and later interpreted on-demand. The result can be accessed in a file-based and an API-based way. Thereby, online and offline information exchange of comprehensive information models is enabled in a bidirectional way. We illustrate how we applied our solution to a real-world application example in an industrial context by translating proprietary ABB Ability™ digital twins to the Asset Administration Shell (AAS) format [6] proposed by Plattform Industrie 4.0 as an agreed information exchange format. The example models describe a system consisting of an electric motor and an industrial drive.

We show that a customizable transformation system simplifies interoperable provisioning of asset information using digital twins in the IIoT. Our solution reduces the effort and time for companies to share digital twins with others to enable advanced IIoT use cases. Thereby, smart use cases that connect industries as well as – presently very desirable – applications of artificial intelligence in industry are enabled.

The remainder of this paper is structured as follows: The next section illustrates an example use case, which is utilized to derive requirements in Section 3. Section 4 describes our solution, a configurable IIoT model transformation approach. In Section 5, we discuss an example application. Section 6 addresses related work and Section 7 concludes and gives an outlook for future work.

This paper is an extended version of our earlier conference paper [7].

2. Example use case

Among many definitions that exist for the concept of digital twins, we adopt the one provided by Industrial Internet Consortium (IIC): A digital twin is a digital representation (attributes, behavior) of an entity (device, system, etc.), which is sufficient to meet the requirements of a set of use cases [4]. This definition implies that digital twins are not limited/required to have real-time operational data of devices, but any other information about entities such as design models, engineering models, simulation models, sales and ordering information are valid content of digital twins. Also, there is no "the" digital twin for an entity; but the content of digital twins is determined by the use case for which the digital twins are designed.

An entity for which a digital twin is created has a unique identifier in the industrial system; this unique identifier can be, for example, the serial number of a device. For more complex entities, such as a plant, some agreed identifiers should be assigned. Likewise, in the cyber world, digital twins have their own unique identifiers. These identifiers facilitate the binding of digital twins to their underlying entities.

In this paper, as a use case, we consider an ABB ACS880 drive and an ABB electric 3-phase squirrel-cage motor that together are part of a powertrain system for one of ABB's customers. The drive and the motor are connected via an edge gateway to the ABB Ability™ cloud platform, which maintains operational as well as initial engineering parameters of the devices within the digital twins of these devices.

The customer aims to access a subset of the powertrain's current parameters from included operational models and to compare those to initially engineered set points from the engineering models. Examples are the minimum and maximum speed of the motor, but there are often several hundreds of such parameters. The current parameters of actually running devices are part of the customer's own digital twins. The current parameters often deviate from the default configuration as they are subject to optimization by service engineers and often have to be adapted to the specific environment of the devices during installation or maintenance. The ability to analyze the deviation of running devices and their default configuration from the engineering phase, allows the customer to potentially optimize their device orders as well as their (usually very expensive) maintenance activities. Since our example customer uses multiple cloud platforms and tools, they would like to receive information in a common (standardized) format in order not to deal with different digital twin formats. After receiving information, the customer merges it into their digital twin, alongside with other information (e.g., installation information) that they receive from other platforms.

An alternative use case is the manufacturer or also the customer planning to use analytics based on artificial intelligence, using the data of several lifecycle phases of their devices.

For all such use cases, which aim to connect several parties, interoperability of digital twins in different platforms becomes essential.

2.1. Challenges

We distinguish between two approaches to achieve interoperability: peer-to-peer transformation of digital twins and transformation to a common digital twin model, used as an intermediate model to transform into a company-specific one. While both approaches are valid and may be needed for different use cases, the former may not easily scale up if the number of participating platforms increases. The latter approach scales better, but it requires a standardized digital twin format, which does not exist currently but may appear in future. Alternatively, as it is the case for other areas, there might be multiple standards defined for one concept in future. Instead of having an internationally standardized format, there can also be currently agreed formats among multiple companies and consortia (e.g., AAS [6]).

This scenario motivates the need for means to (semi-)automatically transform digital twins from one format to another, let it be peer-to-peer transformation, a transformation to one or more international standards, or an agreed format. The need for an appropriate approach is greatly magnified, considering that companies invest a lot of money in digital twins [1]. The requirements for such an approach are addressed in Section 3.

2.2. Scope

Information contained in a digital twin is available in the form of *information models* [8] describing information from different

aspects of the asset. A digital twin may also include mathematical, analytical, and simulation models. In this paper, however, we limit our scope to object-oriented information models that consist of object structures, properties, methods, etc.

The IEC 21823 standard [5] defines five facets of interoperability for IoT systems. In this paper, we focus on two of them: syntactic and semantic interoperability. Syntactic interoperability refers to the case that the formats of the exchanged information can be processed by the participating systems. Semantic interoperability indicates that the meaning of the information model within the context of a subject area is understood. On the attribute level, this is usually enabled by annotating attributes in the information model with semantic references that link to semantic dictionaries like eCl@ss [9]. Examples are semantic references in AutomationML [10], dictionary references for OPC UA [11], or SemanticIDs and Concept Descriptions for AAS [6].

3. Transformation system requirements

In the following, we identify requirements to achieve interoperable digital twins coping with the challenges sketched and limited to the above-defined scope. The requirements have been derived by analyzing multiple real-world industrial use cases and have been refined during discussions with practitioners from the automation industry.

(R1) Supporting different formats: The most basic requirement is to be able to transform between different formats. We call the company's information models to be transformed the *source models* and the format they are specified in (i.e., a company's proprietary format or metamodel) the *source format*. Source models are to be transformed into *target models* following the *target format* (or target metamodel). Note that an application scenario may have multiple levels of sources and targets: Models defined in Company A's proprietary source format may be transformed into an agreed target format first. Later, (target) models in the agreed format can become source models to be transformed into Company B's proprietary target format.

(R2) Variety of information deployment: Source and target information models can be located either on devices themselves, on a cloud platform, or on edges that connect those devices to the cloud platform. The transformation system should be able to support all of those to read source information models and to propagate resulting target information models back.

(R3) Evolution of source and target formats: Formats for information models – even commonly agreed formats – may change over time. Any approach for transformation has to take the possible evolution of both source and target format into account.

(R4) Semi-automatic definition: Engineers must be able to define and maintain how syntax and semantics of source models are mapped to target models. That includes to define structure and naming of elements and also to determine which parts of the model are to be shared and which not, e.g., for data privacy reasons. However, at the same time, the approach should enable the engineer to get quick results without having to spend a lot of extra manual effort into the definition of the transformation rules by a high degree of automation and reusability.

(R5) Bidirectional mapping: Since information exchange can be bidirectional between companies, the transformation system should enable transformation in both directions between source and target models.

(R6) Composite structures: The transformation should also be able to cope with systems that consist of several assets, where composite structures of information models have to be handled. This means, not only flat lists of information, but also hierarchical/nested structures and relations are to be taken into account and treated appropriately.

(R7) Maintain semantics: In order to attach semantic definitions to pieces of information, information models often already contain references to semantic dictionaries which have to be transformed as well, if supported by the target format.

(R8) Make target models accessible: The system needs to provide other systems with the possibility to access the target models via proper APIs. Access should be possible in an offline way, e.g., per file download, but also in an online way, streaming live data from or to a device or a cloud.

4. A customizable IIoT model transformation system

In the following, we give an overview of our approach according to the requirements from Section 3 and discuss architecture and main concepts of the proposed system.

4.1. Overview

There are multiple deployment alternatives for transformation systems and mapping models: the mapping can be done in the source system (as shown in Fig. 1), at the target system or also in an intermediate system. Where the transformation system is deployed is typically depending on which system is about to be integrated with another — in our example use case, the source system shall be integrated with the information demands of other target systems.

Fig. 1 shows an overview of the solution that we present in this paper. Within the scope of the `Source System`, there are many devices, e.g., drives, motors, and robots. Each device may be associated with a collection of information models, specified in the format used by the source system (`Information Models in Source Format`). These models capture the attributes and behavior of the devices and are part of their digital twins. The information models may be located directly on the device or a cloud platform connected to the devices or even be distributed in different locations. Information can also be stored on the edge, i.e., on a device located on-site, acting as a mediator between the devices and the cloud.

To provide controlled access to these information models or specific parts of them, we make them accessible in terms of the target format, i.e., as `Information Models in a Target Format`. However, we do not directly map the information models into the common format but use an intermediate model, that we call the `Mapping Model`. The mapping model can be viewed as a collection of customizable model transformation rules, that describes how the elements of the source format are mapped to the target format.

`Target Systems` can be other systems within the same company or of customers and partners that access the target models either directly like in the picture, or by using more transformations in between. Each of the companies on the side of the target systems may perform their own use cases based on the exchanged information, e.g., predictive plant design. The access can either be performed in an offline or online fashion: Offline access to information requires to transform the current snapshot of the information models into the target format and provide the target models as a download, which can be imported into the target systems. Alternatively, an API, e.g., a REST API, can be used

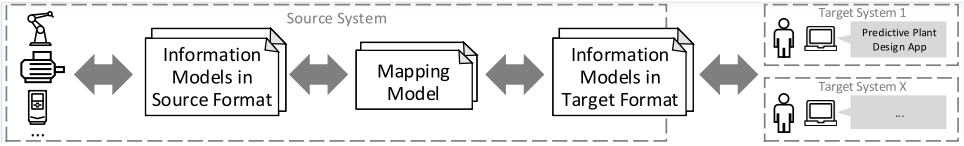


Fig. 1. Overview of the approach at runtime.

to pull the target models. Online access means that the accessor establishes a connection via a protocol that allows subscribing to changes in the information models, such that the transformation into the relevant target format concepts is performed continuously, in a push-fashion. Such a realization could be based, e.g., on OPC UA (IEC 62541) [12].

The scenario described here addresses the system at runtime, where the mapping model is already in place and continuously used. Before all that can happen, there needs to be also a setup phase, i.e., the phase where the system is setup at design time. In this setup phase, the mapping model is created by an engineer. More details about the creation of the mapping model follow in the next subsections.

4.2. Transformation system architecture

Fig. 2 shows the architecture of our transformation system. The components of the core transformation system are the `Mapping Model Builder` and the `Mapping Model Interpreter`. The `Mapping Model Builder` component takes source information models from the `IoT Facade` and generates a mapping model based on the structure of the input models. The mapping model may be customized afterward (see Section 4.3). The creation of the mapping model is performed at design time, before the translated model is used. `Mapping Model Interpreter` is executed at runtime, whenever the translated target model is requested. It takes the mapping model, accesses the source models in the way specified in the mapping model, and provides information in the required, translated format, following the mapping model.

For accessing the source models, both `Mapping Model Builder` and `Mapping Model Interpreter` use an `IoT Facade`, which represents an interface to the location where the models are stored. Depending on the location, different adapters can come into play here (R2), e.g., if the information to be accessed is from an IoT platform, a kind of "Cloud Adapter" or "Edge Adapter" is required. If the information is accessed from the device itself, a "Device Adapter" (which may be reusable for many different kinds of devices if they use the same or similar communication protocols) is used.

For providing live access to the translated target information model, the `REST API` is used. That means calls on the REST API are executed based on the target model and the consuming application that was built based on the target model can make use of the information natively. In each request made on the REST API, the bidirectional mapping capability of the mapping model is used to build up a response based on the underlying source model. This way, the REST API provides a response that represents the live state of the digital twin. Please note that we currently read/update the whole information model at once via the API in our system, mappings between APIs is subject to future work. The `Publisher` provides publish–subscribe based access towards other systems to support direct reaction to changes in the target model.

In Fig. 1, the system and the mapping model are deployed as part of the source system, but they could also be located on third-party cloud systems or with the target system. To provide the digital twin in an offline manner, e.g., using an exchange format and providing file downloads, an `Exporter` component is used. This component accesses the `Mapping Model Interpreter` directly or via a `REST API` to create a snapshot of the actual state of the target model. These components represent the interface to other companies that process the translated information (R8).

The `Importer` component is the counterpart to the `Exporter` component and can be used to import snapshots provided by others. Similarly, `Subscriber` is the counterpart to `Publisher` to automatically adapt to changes through live access to another system. Both components make use of back-transformations in terms of bidirectionality (R5). They make use of the mapping model to affect submodels and properties within the source model at the source system.

4.3. Mapping model

The mapping model can be seen as a set of model transformation rules that specify how the source model is translated into the target format. As known from model transformations, the rules are defined on the meta-level, i.e., on the level of the source and target format, while they are executed on instance level, getting a concrete source model as an input and returning concrete target models as an output. The concept of model transformations is well-established and has been applied many times and comes in many variations [13]. The innovation in our approach is the easy, but flexible creation of the transformation rules (R4), tailored to information models of digital twins in IIoT.

The main features of the mapping model are:

- The mapping model decouples the transformation of source models to target models such that it will be easier to change the transformation whenever either source or target formats evolve (R3).
- It provides access to all relevant information. It contains interpretable links to the endpoints that can be used to access the information that has to be transformed, e.g., data elements, and to all data associated with it, e.g., element's attributes like units (R2).
- Its content is specific to the target model format and defines everything the system needs to know to create a valid instance of that format (R1). It defines the structure the information should have after the transformation, e.g., grouping of elements into different substructures (R4).
- It can be used to filter out information that the company does not want to expose (R4) as defined by engineers customizing the model. The engineer customizing the mapping model fully controls which part of the information is translated and which is ignored.

An exemplary format of the mapping model is defined in Fig. 3(a) in the form of a meta-model formalized as a UML class diagram. Every `MappingModel` needs some information to identify the model and the asset that is addressed by this model (here: `modelId` and `objectId`). Furthermore, it refers to a list of `PropertyLinks` that define where to access which data element in the source model and to which element to map it to in the target model. `PropertyLinks` encode a large part of the transformation logic. `Mapping Model Interpreter` will use these links at runtime to get the latest value for the corresponding

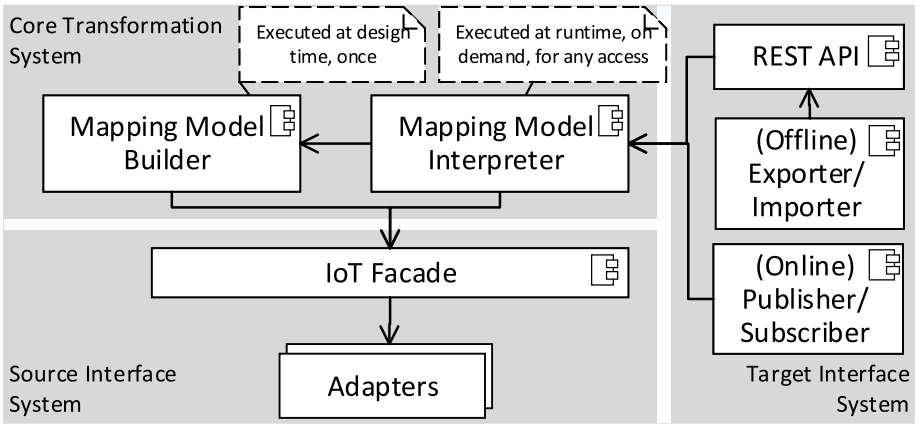


Fig. 2. Architecture of the transformation system.

property on-demand via the IoT Facade (cf. Fig. 2). This way, the mapping model avoids duplicating data and, thus, does not have to deal with synchronization issues. The mapping model also defines the `ContentStructure`, however, the concrete structure is specific to the target format. In our AAS example (see Section 5), content is structured using submodels, so this is reflected in the mapping model as well.

To create composite information models, i.e., information models of a higher-level system which may contain or refer to the information models of the assets it is composed of (e.g., like a powertrain is composed of several drives and motors), a `CompositeMappingModel` can be created (R6). A composite mapping model consists of links to the mapping models of its constituent parts.

Fig. 3(b) shows an excerpt of an exemplary mapping model from our example application (see Section 5). The `modelId` and the `objectId` are used to identify the asset that is described by the information model that this mapping model addresses. If the target format needs more header-like information, this can also be specified here. The `content` section defines the properties that describe the asset, structured in a certain way. In our example, the target model is structured in submodels containing elements that can be arbitrarily nested and the submodels are identified by their name (e.g., `operationalModel`) and their ID (`drive.submodel.operational`). If the target format requires more information per submodel, this information would be specified here as well.

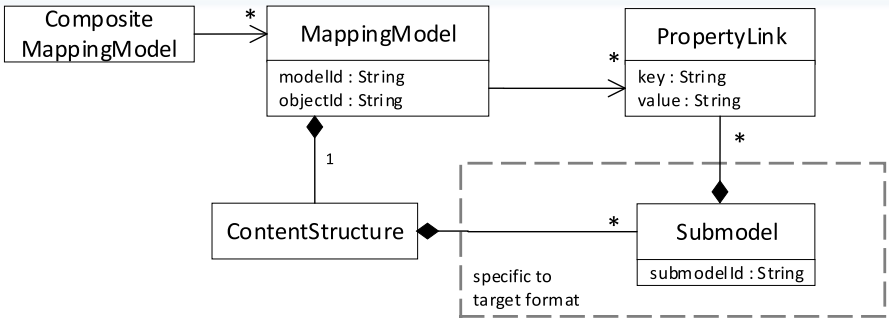
The actual data elements are displayed in the `propertyLinks` section. As shown in the figure, the mapping model does not contain any actual value for the properties but only the links to their location. Nesting of elements as required by the target model is currently handled by nesting of properties in the source model. However, the meta-model in Fig. 3(a) could be extended to add additional nesting hierarchies in order to give more control to the engineers that customize the mapping model. The property links are in the format: `name-of-property-in-target-model: @name-of-source-model/ path-to-property-in-source-model`. The properties in the source model can have different names as in the target model (e.g., "frequency" vs. "output_frequency"). Also, whole groups or substructures of properties can be linked, if the source format allows that. The example mapping model also shows that the structure of data in the target model can deviate from the structure in the source model: for example, the `engineeringModel` submodel in this example will include properties from different source models, "abb.ident" and "abb.drive.engineering".

4.3.1. Generation of initial mapping models

The `Mapping Model Builder` contains a collection of strategies for automatic generation of the initial version of the mapping model based on the source information models. This keeps the effort for defining the transformation low (R4). Examples are:

- The Structure-preserving Strategy: In this simple strategy, the mapping model is generated in a way such that the target model's structure resembles the structure of the source model. For example, the target model contains the same substructures, while properties might be different.
- The Min/Max-number-of-models Strategy: Here, the generated mapping model would produce target models that contain a minimum number of models (e.g., one) or a maximum number of models (e.g., one model per data element). For example, in the "min" case, three source models will be transformed into one target model which aggregates all properties from the three source models, without any additional structure.
- The Name-based-heuristics Strategy: The target models are generated based on domain-specific, customizable heuristics regarding the naming of the elements in the source models. For example, there may be rules like "Elements with a name beginning with 'Motor*' are always assigned to the 'motor' model" or "Elements containing '*service*' in their name always go to a 'maintenance' model" or "Elements named 'SerialNumber' always go into an 'identification' model, if existing". Such rules can be added by engineers and be loaded into the system at runtime.
- The Semantic-references-based Strategy: Semantic references linking to dictionaries like eCl@ss [9] can be used for the generation of a mapping model such that target models are based on structures defined in these dictionaries.

The engineer can select between these strategies depending on the use case's requirements. For example, for a use case where the elements usually have very clear names, a target application could benefit from the Name-based-heuristics Strategy. However, if the target application is a machine learning algorithm that expects to process flat structures, the Min-number-of-models Strategy could be appropriate. More of such strategies can be added to the system at any time. Furthermore, combinations of such strategies can be applied iteratively. Also, note that the generated mapping model is always fully functional but may often need to be customized to optimize it to the special needs of the target use case.



(a) Meta-model of the mapping model

```
{“modelId“ : “ability.adminshell.mappingmodel“,
“objectId“ : “12345678-1234-1234-1234-123456789876“,
“content“: {
    “submodels“ : {
        “operationalModel“ : {
            “submodelID“ : “drive.submodel.operational“,
            “propertyLinks“ : {
                “frequency“ : “@abb.ability.device/variables/output_frequency“,
                “power“ : “@abb.ability.device/variables/output_power“,
                ... }
        },
        “engineeringModel“ : {
            “submodelID“  “drive.submodel.engineering“,
            “propertyLinks“ : {
                “serialNumber“ : “@abb.ident/properties/serial_number“,
                “minSpeed“ : “@abb.drive.engineering/properties/min_speed“,
                “maxSpeed“ : “@abb.drive.engineering/properties/max_speed“,
                ... }
        },  ...
```

(b) Excerpt of an example mapping model

Fig. 3. Mapping model.

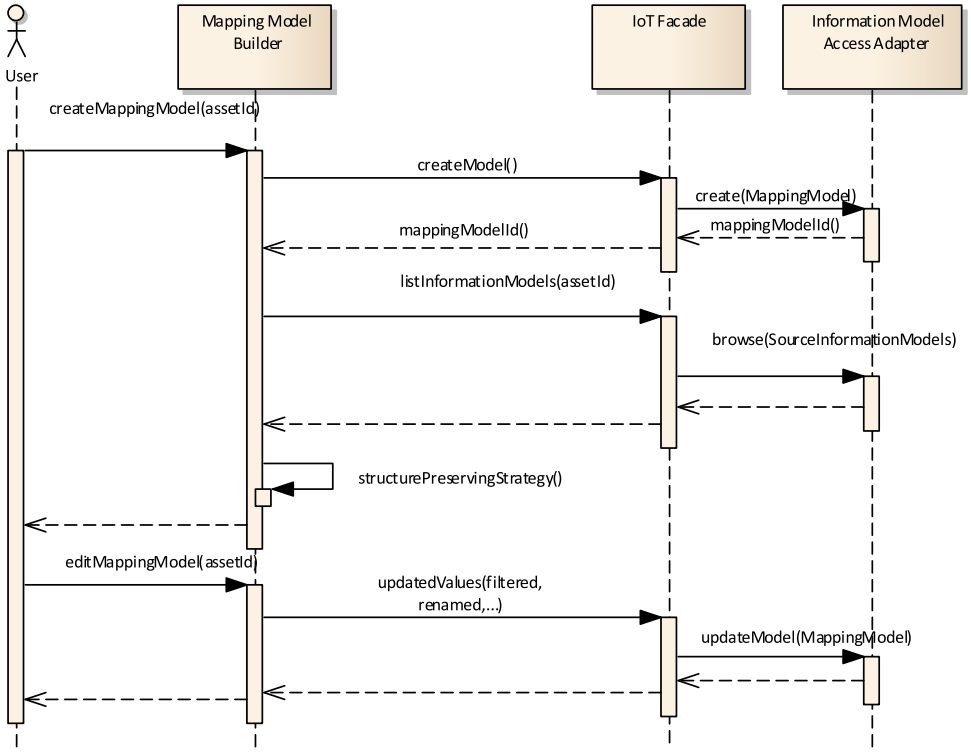


Fig. 4. Dynamic view of initial mapping model generation.

4.3.2. Dynamic view of initial mapping model generation

The dynamic view of the system during creation of a mapping model (i.e., the setup phase) is shown as a sequence diagram in Fig. 4. Lifelines represent system components found in Fig. 2. In addition, we detail a specific source system adapter which is accessed via the `IoT Facade`. Here, the `Information Model Access Adapter` is used to interact with the mapping model.

The creation of the mapping model is triggered by the engineer in case a data export for a given `assetId` to a target system is required. At first, the mapping model with a `modelId` is created. In the consecutive step, existing information models within the source system are queried according to the given `assetId`. The asset ID is linked or used as internal `objectId` that is used to reference models within the source system. Then, a structure-preserving initial mapping model generation strategy is applied as described in the last section.

Note that the handling of IDs, such as `objectId`, `modelId`, and `assetId` is a crucial part of the transformation. Since digital twin identifiers are internal to each party, we cannot assume that these internal IDs will be communicated among these parties and be incorporated directly in the model transformation process. Instead, the transformation system has to enable the target system to direct the information from the source system to the right target digital twin.

After a mapping model is created, the user (engineer) has the opportunity to edit the mapping model (R4). For example, elements can be removed from the mapping model in order not to be exported to the target system.

4.4. Import

By now, we put the most attention on one transformation direction: from the information models in source format to the information models in the target format. However, as stated in R5, there is also the need for bidirectionality. This means, also the import scenario, i.e., the transformation from target format to source format needs to be realized.

Since also this case benefits from customizability, the mapping model is used as an intermediate model here as well. Again, the mapping model decouples the model-specific transformation logic from the transformation itself. The mapping meta-model however is again specific to the target format, i.e., the source format, in this case. The procedure works analogously to what we described before (see, e.g., Fig. 4).

In addition to importing a snapshot of the target system, an alternative could also be a publish/subscribe mechanism to take online changes into account.

5. Example: Interoperability via asset administration shell

We have validated the proposed solution by implementing our customer's use case introduced in Section 2. The addressed example system had been exhibited by ABB at Hanover Fair 2019 and a part of the system can be seen on Youtube.[3]

Our solution is based on ABB Ability™,[4] which lays the foundation for ABB's digital services and solutions offerings based on the Microsoft Azure cloud platform. Among other features, ABB Ability™ offers an information model service that provides access to the information models. Ability information models are object-oriented models that, amongst others, describe static properties of a device as well as links to its historic time series data in the form of variables.

In our example application, we show how ABB Ability™ information models can be accessed in the format of AAS and thereby provide interoperability to other systems and platforms.

In the following, we provide some background on AAS, followed by details of our implementation taking ABB Ability™ models as source models and AAS as target format.

5.1. Asset administration shell

An AAS, as specified by the Plattform Industrie 4.0 consortium, provides a digital representation of an asset in order to enable interoperability across a value chain where multiple organizations may be involved [6]. An asset is a physical or logical object having a value to the organization, e.g., IIoT devices, systems composed from such devices, product types.

Amongst others, an AAS contains the following parts:

- *Submodels*: Submodels are used to structure an AAS into distinguishable parts. Each submodel refers to a well-defined domain or subject matter (e.g., operational or engineering model) of an asset. Submodels are expected to become standardized, e.g., a documentation submodel.
- *Submodel elements*: Each submodel consists of a set of elements. Submodel elements further describe the asset, including properties (e.g., a temperature value) or files (e.g., a documentation manual), and references (referencing to another element within the same or another administration shell).
- *Identification*: An AAS addresses unique identification of administration shells, submodels, and submodel elements, e.g., using IRIs and URIs.
- *Semantic references*: Each element of an AAS can be annotated with semantic references, e.g., linking to eCl@ss properties [9] and thereby attach a standardized, semantic definition to each element.
- *Instance and Type levels*: An AAS can either refer to an asset instance (e.g., a concrete drive located in Ladenburg) or a product type of an asset (e.g., the ABB ACS880 drive type).

For a complete description of the AAS meta-model, please refer to [6]. In [6], also security aspects, especially access control, are discussed.

Plattform Industrie 4.0 also specified a packaging format "AASX" based on the Open Packaging Conventions to enable the offline exchange of AAS. Amongst others, the AASX package contains the content of an AAS as an XML file or a JSON or RDF file alternatively. For online transfer, mappings to other protocols, especially OPC UA, are in progress[5] (however, main aspects of the mapping are already available in [6]).

5.2. Solution

In the following, we illustrate our example application based on the use case presented in Section 2 in more detail. We refer to the requirements from Section 3 and show to which extent our solution fulfills them.

Fig. 5 depicts an overview of our example application. The example devices that we need to collect information from are an ABB motor and an ACS880 drive, which together form a (simplified) ABB powertrain.

The information on these devices is collected via an edge and propagated to ABB Ability™. Our example implementation establishes a connection to the ABB Ability™ cloud (R2), which again receives information from the devices via the edge, and pulls and

[3] https://youtu.be/AXQ0yIonNxk.

[4] https://new.abb.com/abb-ability/.

[5] https://opcfoundation.org/markets-collaboration/i4aas/.

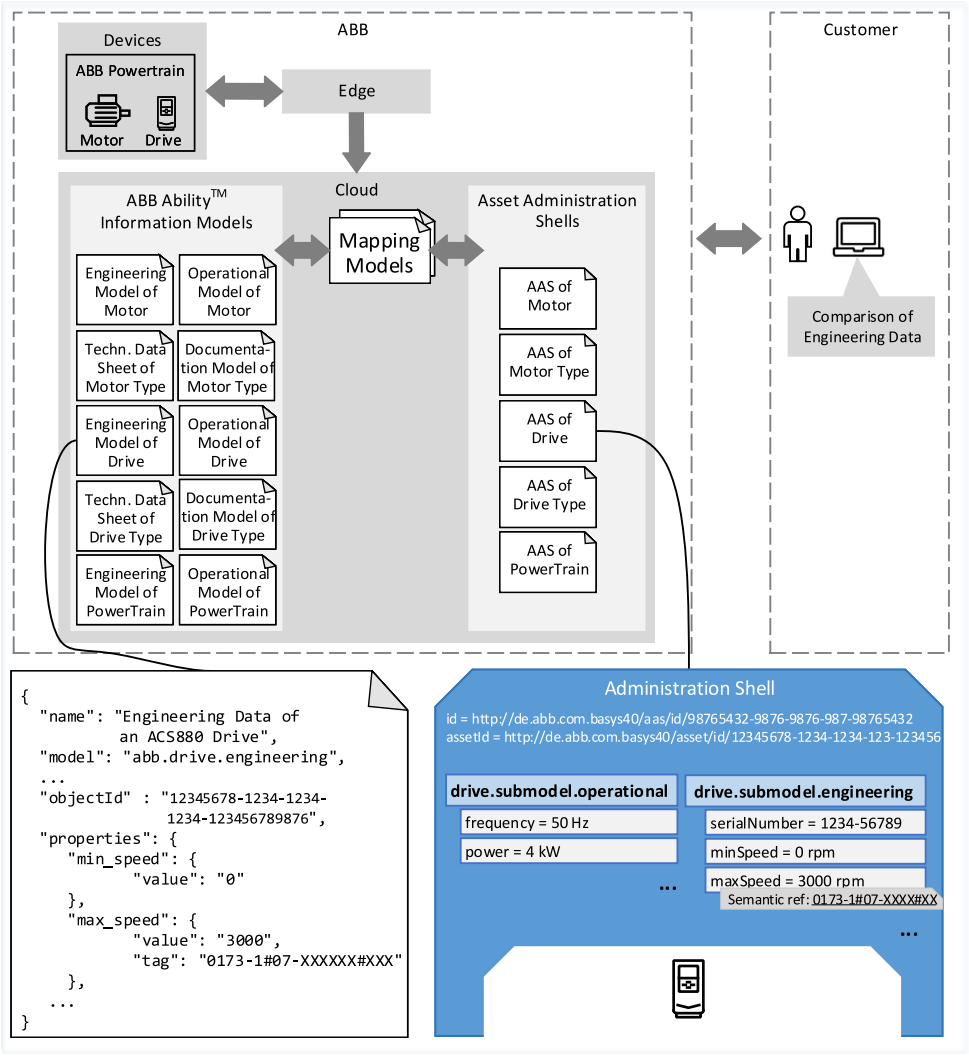


Fig. 5. Overview of the example application.

pushes information from and to it. In our example scenario, we consider 10 source information models: the engineering model as well as the operational model of both the motor and the drive, a technical data sheet model and a documentation model for the motor type and the drive type, and an engineering model and an operation model for the powertrain. The defined mapping models contain links to the ABB Ability™ information models and specify how to translate them into the AAS format. For example, the mapping models in Fig. 5 must have been specified with the knowledge that the property `min_speed` in the source model is to be mapped to the property `minSpeed` in the target model.

The process results in several AASs to cover the composite nature of the powertrain (R6). Accordingly, we get five AASs as an output: one for the motor, one for the motor's product type, one for the drive, one for the drive's product type, and one for the powertrain system. For each AAS, we created two submodels, which makes in total 10 submodels: for the motor, the type, and the powertrain, we transformed an operational and an engineering model for each, while the AASs of the drive's and the motor's product types contain both a submodel with technical data sheet information and a documentation model. The bottom right part of Fig. 5 shows a schematic view of an excerpt of one of the result AASs, the AAS of a drive, following the mapping model from Fig. 3, which has been explained in Section 4.3. Attributes in the source models that were tagged with eCl@ss references, were transferred to corresponding semantic references in the AAS (R7). These references can be, for example, semantically defined properties of the drive type like its height, weight, or environmental protection class. The model can be easily extended to cover further semantic references towards their definition in the standardized dictionaries.

All mapping models for the involved devices were initially generated based on the structure-preserving strategy because the use case required us to distinguish between the operational model and engineering model, which were both already there in the source system and no standardized submodels for them exist so far, such that no restructuring was needed (R4).

5.3. Validation and results

We validated this setting using a prototype of an AAS-based customers' applications shown in Fig. 5. The customer application is implementing an offline data exchange via an offline package following the use case described in Section 2. As mentioned above, the use case was to compare the engineering data with the actual data of a device. Using our solution, the customer can access the needed information in the form of the AAS and develop

an application to read them and to do the comparison. The offline export is described in Section 5.3.1 in more detail.

Another AAS-based application that was used for validation is the AASX Package Explorer which is an open-source AAS processing tool with many AAS-based features.[6] In particular, this application displays content of the AASX package without interpreting the semantics of elements with the AAS. It does not compare current and planned parameters for the drive engineering use case. In addition to the import possibility of offline AASX packages, the package explorer tool offers a RESTful HTTP client enabling an online AAS access which is described in Section 5.3.2.

5.3.1. AASX package export

To make the AAS accessible, we implemented the offline export (R8) for an AASX package, which means, each time the transformation is triggered, it transforms the current snapshot of the information models including the latest values of operational parameters obtained from the devices.

The details of this export are shown in Fig. 6. Here, the AAS export is triggered by the user based on the existing mapping model. The model's `objectId` is used as the `assetId` in the domain of the asset administration shells. At first, the mapping model is obtained by the `Information Model Access Adapter` via the `IoT Facade`. The mapping model is then resolved iteratively by the `Mapping Model Interpreter` component. Static meta-information for each AAS submodel element like element's name and semantic identifier is already contained within the mapping model. The latest values of the elements need to be additionally read from the `Live Value Access Adapter`. Finally, the `Mapping Model Interpreter` returns an AAS model that needs to be packed as AASX package and provided for download to the user.

Also, an import scenario, where a third-party AAS was given and its information was transformed back into the ABB Ability™ system, was successfully implemented within the prototype (R5).

5.3.2. AAS access using a REST API

The AAS concept also targets at live access to the content of AAS. Besides the fact that after translation the target model can be queried via the `REST API` as shown in Fig. 2, we integrated the AAS REST API on top of the Mapping Model Interpreter. As result, the AASX Package Explorer tool can be used to browse the content of the AAS. On each request on the AAS REST API the mapping model is used to map parts of the source model to the target model. Fig. 7 shows that, whenever the user requests a property within the AAS (`getAASProperty()`), the property in the source model is identified using the mapping model and then the latest value is returned and shown as an AAS property within the customer's AAS browser. Depending on which target representation is required, the RESTful API server can provide the data accordingly, e.g., in JSON.

5.4. Composite AAS

AAS also includes concepts for composite components to model hierarchical relations between models [6]. As described in Section 4.3, also our transformation approach is able to deal with composite structures. Using a composite mapping model, a composite AAS has been realized for the powertrain, referring to child mapping models for the motor, the motor type, the drive, and the drive type.

[6] https://github.com/admin-shell/aasx-package-explorer.

5.5. Limitations

All in all, our example application showed that we were able to specify the mapping model such that all involved information models could be used and the example use case could be fulfilled due to its configurability supporting arbitrary structures of information (R4). The example application, however, only shows ABB Ability™ information models as source models and AASs as target models. The mapping model concepts are flexible such that also all kinds of other model formats can be supported (R1), but detailed case studies for other model formats are subject to future work. Also, evolving model formats (R3) need to be validated, even though the mapping model serving as an intermediate model enables decoupling between source and target formats such that only parts of the mapping model have to be adapted if one of both changes. The prototype was used to validate the mapping model concepts considering a transformation from ABB Ability™ information models as source models and AASs as target models. This means the implemented mechanisms for transformation are limited to the set mechanisms that are required for this validation. As mentioned in Section 4.3 the mapping rules are defining mapping on the level of meta-models, e.g., supporting selection of information based on structural aspects. As far as the meta-model supports a type system, the mapping model could support the definition of mapping rules also based on types to enable better and more fine granular re-use of mapping rules. Also, the mapping model is not supporting the transformation of the telemetry data (e.g., transforming property values).

A comprehensive concept to include role-based access control (R8) within the transformation concepts is still subject to future work as well. However, in our approach, information can be filtered and not everything is exposed, which is important, as – e.g., for security reasons – companies may not want to give direct access to their digital twins. Also, the target format in our example application, the AAS, contains more features regarding this matter, but we need to take into account more sophisticated mechanisms.

6. Related work

Interoperability of IoT systems, in general, is an important topic and a subject of various standardization activities [5]. Although several researchers study data fusion from various sources into digital twins [14], to the best of our knowledge, there is no other work that studies the application of model transformations to enable digital twin interoperability.

Existing commercial proposals for digital twins (e.g., by GE or Microsoft) are proprietary to companies. In [15], AutomationML is used for modeling digital twins and achieving interoperability. AutomationML can be seen as one standardized model in our system, to which different proprietary digital twins can be transformed. However, as illustrated by AAS, our system applies to more models.

Data flow platforms, e.g., Node-RED [16], are used in the IoT community to interconnect devices and systems. These platforms provide building blocks to create workflows that are initiated by certain events. Within a flow, data transformations are typically described imperatively, e.g., using JavaScript. In contrast, our system uses a declarative mapping model that is decoupled from the event flow.

Xiao et al. [17] presented a "user interoperability framework" for IoT systems to deal with heterogeneous devices. They propose a device transformability model, however, they do not deal with information models, which is needed to address the interoperability challenge for digital twins.

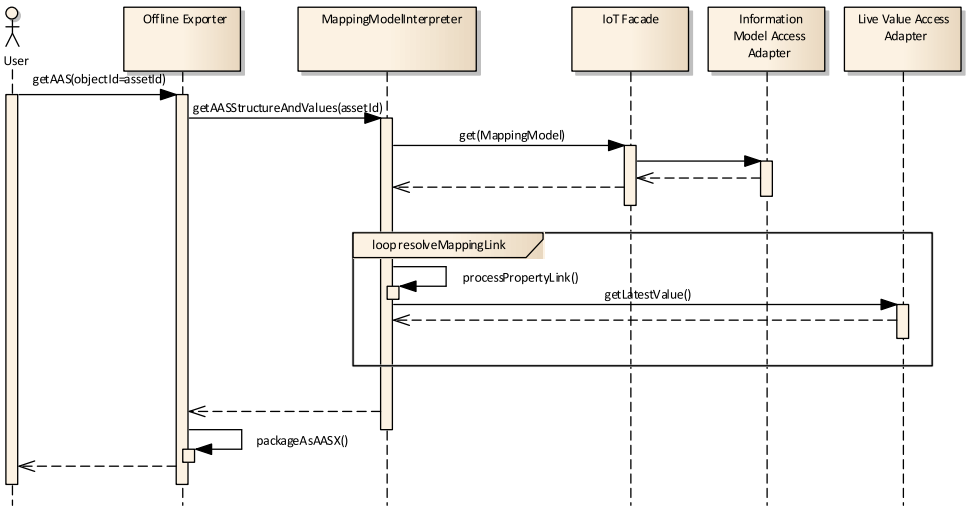


Fig. 6. Dynamic view of AAS package export.

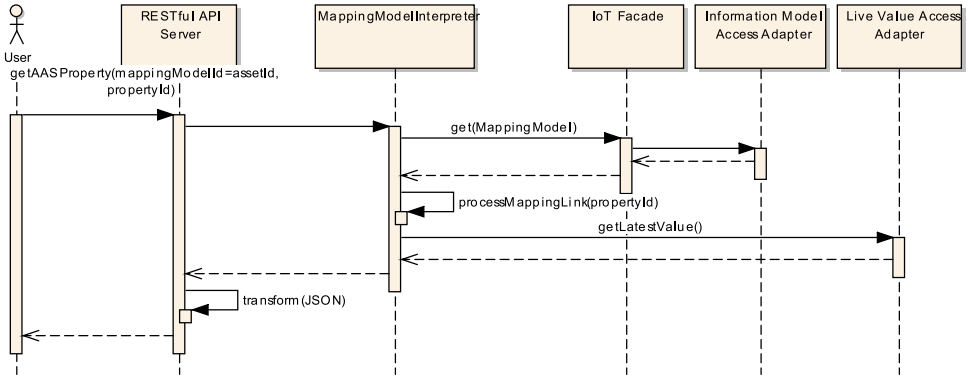


Fig. 7. Dynamic view on RESTful access to single AAS properties.

In [18], experiences and best practices for information modeling in OPC UA were presented. One use case referred to in the paper was the mapping of arbitrary protocols and implied information models to OPC UA. A manual mapping was selected due to the lack of a generalized mapping language. Our proposed system including a descriptive mapping model can be one possibility to fill this gap.

In [19], we proposed an architectural pattern for digital twins, which enables receiving information from multiple sources. However, we have not studied the necessary model transformation to enable interoperability in that context.

Model transformations, in general, have been studied a lot (see, e.g., [13]). We did not use existing model transformation languages (e.g., QVT, ATL) to keep the complexity low and to simplify engineers the handling of the transformation rules as much as possible. However, existing transformation languages come with many features that could be useful in the future, e.g., to take into account conversions of property values using more complex expressions and formulas.

Nguyen et al. [20] presented a model-driven approach to develop IoT applications. As part of a model-driven approach, model transformations are used as well, however, they are not used to make systems interoperable, but to develop them in a homogeneous fashion in the first place, which is not realistic in the scenarios our approach deals with.

In the domain of Web of Things (WoT), there are generic approaches to describe *Things* using the *WoT Thing Description* [21] which uses JSON-Linked Data (JSON-LD) [22] as default encoding. JSON-LD in turn can be used to link up Resource Description Framework (RDF) based semantic descriptions in a world-wide semantic web. The WoT Thing Description could be used (similar to semantic dictionaries like eCl@ss [9]) to link up several data sources with a common semantic. Therefore, approaches to generate RDF from heterogeneous data sources (cf. [23,24]) can be applied. Our approach is not bound to RDF as a target format. Furthermore, as already mentioned above, we intend to simplify the handling of transformation rules as much as possible and to reuse the structure of the source model if possible (i.e., by applying the structure-preserving strategy).

7. Conclusion

In this paper, we presented a system to enable interoperable digital twins and discussed how we applied it to the real-world application example of ABB devices, ABB Ability™ information models, and the asset administration shell. We are the first to show that a transformation system simplifies the standardized provision of information on devices via digital twins, which reduces the effort and time for a company to share information with others to enable advanced IIoT use cases (the transformation approach itself is not limited to the IIoT domain). In particular, our approach enables smart use cases that connect industries including customers and manufacturers as well as manufacturers among each other. In addition, we enable artificial intelligence in industry by simplifying access to data in general.

As this paper was limited to some facets of interoperability and focused on the transformation of information models of a digital twin, in the future, we plan to broaden the scope, to achieve complete digital twin interoperability. This includes considering analytical and simulation models and to integrate more interoperability facets like transport interoperability, behavioral interoperability, and policy interoperability. Further topics for future work include concepts regarding access control (e.g., role-based or attribute-based), mapping between API definitions and validation of the system on further example applications addressing different source and target models.

CRediT authorship contribution statement

Marie Platenius-Mohr: Conceptualization, Methodology, Software, Validation, Writing - original draft, Writing - review & editing, Visualization. **Somayeh Malakuti:** Conceptualization, Methodology, Validation, Writing - original draft, Writing - review & editing, Supervision. **Sten Grüner:** Conceptualization, Validation, Resources, Writing - original draft, Writing - review & editing, Visualization, Supervision, Project administration. **Johannes Schmitt:** Methodology, Software, Validation, Writing - original draft. **Thomas Goldschmidt:** Conceptualization, Methodology, Resources, Writing - review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] B. Lheureux, A. Velosa, M. Halpern, N. Nuttall, Survey Analysis: Digital Twins Are Poised for Proliferation, Gartner Research Notes, Gartner Inc., 2019.

[2] C. Wagner, J. Grothoff, U. Epple, R. Drath, S. Malakuti, S. Grüner, M. Hoffmeister, P. Zimmermann, The role of the industry 4.0 asset administration shell and the digital twin during the life cycle of a plant, in: 22nd IEEE Int. Conf. on Emerging Technologies and Factory Automation, 2017, pp. 1–8.

[3] S. Malakuti, J. Schlake, C. Ganz, E. Harper, H. Petersen, Digital twin: an enabler for new business models, in: VDI Automation Congress, 2019, in press.

[4] Industrial Internet Consortium, Digital Twins for Industrial Applications, 2020, https://www.iiconsortium.org/pdf/IIC_Digital_Twins_Industrial_Apps_White_Paper_2020-02-18.pdf.

[5] ISO/IEC Organization, ISO/IEC 21823-1 Internet of things (IoT) – Interoperability for iot systems – Part 1: Framework, 2019.

[6] 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 2.0), 2018, https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Details-of-the-Asset-Administration-Shell-Part1.html.

[7] M. Platenius-Mohr, S. Malakuti, S. Grüner, T. Goldschmidt, Interoperable Digital Twins in IIoT Systems by Transformation of Information Models: A Case Study with Asset Administration Shell, in: Proceedings of the 9th International Conference on the Internet of Things (IoT'19), ACM, 2019, pp. 1–8.

[8] Y.T. Lee, Information modeling: From design to implementation, in: Proceedings of the Second World Manufacturing Congress, Int. Computer Science Conventions Canada/Switzerland, 1999, pp. 315–321.

[9] eCl@ss e.V., ECl@ss classification and product description, 2019, https://www.eclasscontent.com/, accessed: 2019-05-20.

[10] S. Biffl, O. Kovalenko, A. Lüder, N. Schmidt, R. Rosendahl, Semantic mapping support in AutomationML, in: Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), 2014, pp. 1–4.

[11] OPC Foundation, OPC 10001-5: Amendment 5 – dictionary reference, 2019.

[12] Int. Electrotechnical Commission, IEC 62541-1: OPC Unified Architecture – Part 1: Overview and concepts, 2016, https://webstore.iec.ch/publication/25997.

[13] K. Czarnecki, S. Helsen, Feature-based survey of model transformation approaches, IBM Syst. J. 45 (3) (2006).

[14] F. Tao, H. Zhang, A. Liu, A.Y.C. Nee, Digital Twin in Industry: State-of-the-Art, IEEE Trans. Ind. Inf. 15 (4) (2019) 2405–2415.

[15] G. Schroeder, C. Steinmetz, C. Pereira, D. Espindola, Digital Twin Data Modeling with AutomationML and a communication methodology for data exchange, IFAC-PapersOnLine 49 (2016).

[16] M. Blackstock, R. Lea, Toward a distributed data flow platform for the web of things (distributed node-red), in: 5th Int. Workshop on Web of Things, ACM, 2014, pp. 34–39.

[17] G. Xiao, J. Guo, L.D. Xu, Z. Gong, User interoperability with heterogeneous iot devices through transformation, IEEE Trans. Ind. Inf. 10 (2) (2014) 1486–1496.

[18] D. Schulz, R. Braun, J. Schmitt, Behind the façade, in: 2015 IEEE 20th Conf. on Emerging Technologies Factory Automation (ETFA), 2015.

[19] S. Malakuti, J. Schmitt, M. Platenius-Mohr, S. Grüner, R. Gitzel, P. Bihani, A four-layer architecture pattern for Digital Twins, in: Europ. Conf. on Software Architecture, 2019.

[20] X.T. Nguyen, H.T. Tran, H. Baraki, K. Geihs, Frasad: A framework for model-driven iot application development, in: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 387–392.

[21] V. Charpenay, S. Käbisch, H. Kosch, Introducing thing descriptions and interactions: An ontology for the Web of Things, in: SR+ SWIT@ ISWC, 2016, pp. 55–66.

[22] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, N. Lindström, Json-ld 1.0, 2010.

[23] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, RML: A generic language for integrated RDF Mappings of Heterogeneous Data, Ldow 1184 (2014).

[24] M. Lefrançois, A. Zimmermann, N. Bakerally, A SPARQL extension for generating RDF from heterogeneous formats, in: European Semantic Web Conference, Springer, 2017, pp. 35–50.

Marie C. Platenius-Mohr is scientist at the software Architecture group of ABB Corporate Research in Ladenburg, Germany. Before, she was a postdoctoral researcher at the Software Engineering Group of the Heinz Nixdorf Institute at Paderborn University. There, she led the software architecture team of the Collaborative Research Centre 901 "On-The-Fly Computing". She received the Ph.D. in computer science from Paderborn University in 2016. Her main research interests include Internet of Things, software architecture, and model-driven & component-based software engineering.

Somayeh Malakut is a senior scientist in ABB Corporate Research Center in Germany, there as a certified software architect and requirement engineer she focuses on software system architectural aspects of Industrial IoT systems. In 2018–2019, She was the chair of Digital Twin Interoperability Task Group in Industrial Internet Consortium (IIC), as well as the co-chair of the joint task group between IIC and Platform Industrie 4.0 on the topic of Digital Twin Industrie 4.0 Component. She has received her Ph.D. in 2011 from the software engineering chair in the University of Twente, Netherlands, and has worked as a postdoctoral researcher and software engineer on different topics.

Sten Grüner is a senior scientist at software architecture research group of ABB Corporate Research Center Germany. He received a Dr.-Ing. (automaton engineering) and a Dipl.-Inform degrees (computer science) from RWTH Aachen University in 2017 and 2011, respectively. His research interests include software architecture of IoT systems in domains of discrete manufacturing and process industries, information modeling, and software product line engineering for industrial applications.

Johannes Schmit is senior scientist at the Networks and Devices team at ABB Corporate Research in Ladenburg, Germany. Johannes holds a Ph.D. in electrical engineering from the TU Darmstadt, Germany. His research interests include information-centric concepts and communication for automaton systems, and OPC UA on embedded or mobile devices up to cloud based systems.

Thomas Goldschmidt studied computer science at the Hochschule Furtwangen. He received a Ph.D. from Karlsruhe Institute of Technology in 2010 while working as a researcher at the FZI Research Center of Information Technologies in Karlsruhe. Thomas joined ABB in 2010 and worked as a principal scientist at ABB Corporate Research in the area of model-driven development, domain-specific languages as tell as cloud architectures. In 2017 he moved to ABBs Digital organization as a platform architect and not works on defining the architecture of ABBs internal cloud platform.