Software and Systems Modeling
https://doi.org/10.1007/s10270-024-01255-0

SPECIAL SECTION PAPER

Digital twin and the asset administration shell

An Analysis of the Three Types of AASs and their Feasibility for Digital Twin Engineering

Jingxi Zhang[1] · Carsten Ellwein[1] · Malte Heithoff[2] · Judith Michael[2] · Andreas Wortmann[1]

Received: 15 March 2024 / Revised: 25 November 2024 / Accepted: 29 November 2024

Abstract
Engineering digital twins is a software and systems engineering challenge for which no systematic approach exists. The Asset Administration Shell is becoming a popular foundation for digital twins in Industry 4.0 and it comes in different types that support the engineering of different kinds and parts of digital twins. We investigate how it supports realizing common requirements for digital twins. To this end, we investigate how each of the three Asset Administration Shell types can contribute to the systematic engineering of specific components of digital twins. Therefore, we analyzed popular definitions and conceptual models of digital twins and extracted requirements that at least two of them share. We compare the resulting requirements with Asset Administration Shells of different types and conclude with open challenges in the implementation of digital twins with this technology. This supports practitioners and researchers in identifying the most suitable type of Asset Administration Shell for their specific digital twin engineering needs and identifies gaps worthy of future research toward a systematic engineering of digital twins.

Keywords Asset administration shell · Digital twin · Requirements · Manufacturing

1 Introduction

Digital twins [36, 53, 87] are becoming the technological backbone for better understanding, engineering, operating, and managing (cyber-physical) systems [26, 67]. They are investigated, created, and deployed in a variety of domains, including automated driving [23], biology [49], medicine [54], wind energy [64], smart cities [15], civil structures [61], manufacturing [14], and many more [26]. The various digital twins serve different purposes relative to the twinned actual system (AS) [36], including analysis [74], control [93], and behavior prediction [52]. Also, they are used at different times relative to the AS, *e.g.,* prior to its existence to explore its design space [57] or at its runtime to optimize its behavior [13].

A digital twin is a software system [70] that connects to an AS, automatically receives data from it, performs computations, and sends instructions back to it [53]. Such digital twins can use different kinds of models, from (1) engineering models of their twinned counterpart (*e.g.,* AutomationML [83], IEC 61499 [98] models, or simulation [41] models) to give meaning to its data (models of the actual system used in the digital twin), they can be built from (2) software models [69] or event logs [11] (models for the digital twin used during development), and they can use (3) models at runtime to support their configuration by domain experts [55] (models for the digital twin used during operation). Ultimately, digital twins employ these models

Communicated by Javier Troya and Alfonso Pierantonio.

✉ Andreas Wortmann
wortmann@isw.uni-stuttgart.de

Jingxi Zhang
jingxi.zhang@isw.uni-stuttgart.de

Carsten Ellwein
carsten.ellwein@isw.uni-stuttgart.de

Malte Heithoff
heithoff@se-rwth.de

Judith Michael
michael@se-rwth.de

[1] Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW), University of Stuttgart, Stuttgart, Germany

[2] Software Engineering, RWTH Aachen University, Aachen, Germany

Published online: 08 January 2025

together with data observed from and about the AS [77] to describe, predict, and prescribe its behavior [36].

Manufacturing is one of the most prominent domains investigating the use of digital twins [26] and software engineering has produced various technologies to facilitate their engineering [42, 78]. An important implementation technology for digital twins in manufacturing is the Asset Administration Shell (AAS) [96], which is driven by the International Digital Twin Association (IDTA).[1] The AAS is intended to be the single source of truth digital representation of any kind of asset throughout its lifecycle. To this end, the AAS is defined as a hierarchy of data models called *submodels*, each of which represents components or aspects of the asset to be described. For instance, an AAS of a car could feature submodels for the motor, the drive train, the energy efficiency, and many more depending on the purpose of the representation. Based on this idea, three types of AASs have been identified:

- *Type 1 AASs* relate models of an asset (*e.g.,* for a mobile robot, this might include submodels representing the base, sensors, actuators, installed software, task queue, operation history, etc.).
- *Type 2 AASs* connect these models to live data from the asset (for the robot example, this could entail updating the pose information of its base model regularly).
- *Type 3 AASs* can control these assets and communicate to other AASs (*e.g.,* the robot AAS can communicate with a manufacturing execution system AAS to update the task queue model to support new kinds of tasks).

The AAS is considered to be the essential technology for engineering digital twins by the IDTA, has been successfully applied to predictive maintenance [81], automotive device configuration [97], or model management [22], and is subject to ongoing development. Hence, the AAS seems to become an important foundation for systematically engineering digital twins. Hence, understanding the support of its different types for engineering digital twins is beneficial to researchers and practitioners in software and systems modeling. Therefore, we compare the requirements raised by popular conceptual models of digital twins with the different types of AASs. The contribution of this article, thus, is as follows:

1. A summary of common requirements on digital twins based on analyzing digital twin standards and white papers of associations and consortia that foster the realization of real-world digital twins in industry.
2. An analysis of the different types of asset administration shells and their support for addressing the identified requirements.
3. An outlook on future challenges to efficiently engineer digital twins based on this analysis.

In the remainder, Sect. 2 illustrates the state-of-the-art on digital twins and the AAS before Sect. 3 introduces an illustrative example of an AAS from manufacturing. Afterward, Sect. 4 analyzes common requirements on digital twins and Sect. 5 compares the capabilities of the different AAS types with these requirements. Based on this, Sect. 6 outlines challenges and Sect. 7 discusses related work. Finally, Sect. 8 concludes.

2 Background

2.1 Digital twins

The understanding of digital twins differs widely in literature (cf. 112 definitions of digital twins[2]). As of the main purposes of a definition is to decide whether something is in the set of defined things or outside of it, most of these definitions fail at supporting to make this decision precisely. Instead, the lowest common denominator seems to be that a digital twin represents something (*e.g.,* a system, a process, or a workpiece), which is of little use for detailed discussions about their functions, engineering, and operations. Moreover, often these definitions are

- Ambiguous, by deferring to another undefined term, such as a “virtual representation” [6], a “computable virtual abstraction” [90], or a “a virtual projection of the industrial facility into the cloud” [101];
- Narrow, by focusing on specific use cases, domains, or technologies, such as a “digital model of the real network environment” [33] or a ”virtual representation based on AR technology” [74]; or
- Utopian, due to all-encompassing aspirations, such as an “integrated virtual model of a real-world system containing all of its physical information” [76], a “complete digital representation” [59].

Ambiguous definitions would require a definition of their fuzzy base terms (*e.g.,* "virtual representation") to enable deciding whether something is a digital twin or not, while very narrow definitions (*e.g.,* requiring the use of AR technology) prevent understanding digital twins more generally, and utopian definitions would either logically or economically prevent building such digital twin (*e.g.,* modeling the

[1] IDTA: https://industrialdigitaltwin.org.

[2] Digital twin definitions: www.wortmann.ac/digital-twin-definitions.

behavior of the atoms of the windshield wiper fluid of a car usually is not considered required for a vehicle digital twin, yet that would be necessary for the digital twin to contain all physical information of the car).

Among the plethora of definitions, characterizations, and reference models [38] of digital twins, few have been widely accepted to be useful: either by being cited by vast numbers of researchers in the field of digital twins [53, 88], by being formalized into ISO standards [47], or by being accepted as common ground by large industrial associations about digital twins [31, 32]. Section 4 discusses these in detail.

2.2 Asset administration shell

In the context of Industry 4.0 (I4.0) every element owned by an organization having a value for the execution of the process is defined as an asset [22]. Assets can be physical, such as production resources, workpieces, and even the factory itself, but also non-physical, such as models used for describing machine behavior, software, or licenses [40].

The IDTA is developing the AAS [10, 96] to provide a technology for realizing digital twins [71]. This initiative started within Platform Industry 4.0, an initiative of the German Federal Ministry for Economic Affairs and Climate Protection and the Federal Ministry of Education and Research together with German industry, academia, associations, and unions. This technology is now also being taken up in Europe, as calls for EU Horizon start mentioning that proposals should take any relevant international standards (such as the AAS) into account.[3] Alongside its political relevance, the AAS has also achieved industrial relevance. There are currently 118 partners[4] organized in the IDTA (as of July 2024), including German (*e.g.,* SAP, Siemens, and Volkswagen), as well as international enterprises (*e.g.,* HUAWEI, Phoenix Contact, and Mitsubishi Electric).

The AAS is defined as the digital representation of the asset containing all its relevant information throughout its entire lifecycle [72] and is presented as the basis of interoperability: on the one hand, it holds information of various types and on the other hand, it functions as the interface for communication within the I4.0 network through which information can be exchanged between assets [79]. Since the AAS holds relevant information throughout the lifecycle of the asset, it must be capable of representing different sorts of information, such as properties, modeled functionalities, parameters, a summary of included components, as well as data that accrues during manufacturing or simulation and also descriptions, such as their usage instructions and technical specifications. This presupposes the ability to store or refer to heterogeneous data and models [22].

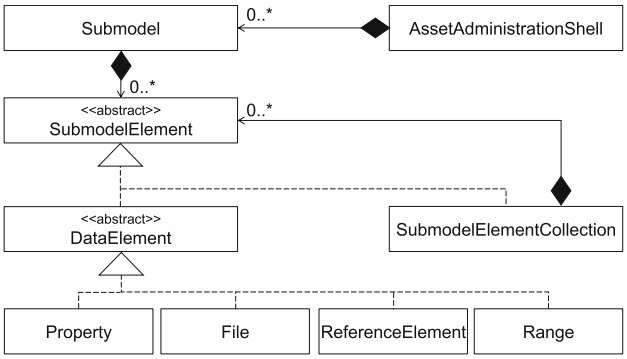

Fig. 1 Structure of the AAS according to the metamodel [35] based on [22, 79]

The structure of an AAS is specified in the form of a conceptual metamodel using UML class diagram syntax [79]. Figure 1 shows an excerpt of this metamodel focusing on the AAS submodels and their relations. Essentially, the AAS is organizing submodels hierarchically and is working on standardizing submodels and their templates. Each individual submodel of an AAS is intended to represent one content-related or functional aspect of the represented asset. Submodels can be created individually applying the previously introduced metamodel. To ensure consistency and interoperability, the IDTA provides so-called submodel templates. Submodel templates are also standardized and publicly available in their content-hub,[5] which currently features 89 submodel templates. Each parameter in a submodel template is specified with an identifier, its semantic meaning as well as a given example. The semantic meaning is indicated in line with established dictionaries, such as the ECLASS reference data standard for the unambiguous description of products and services.[6] and the IEC Common Data Dictionary [45]. The scope of the perhaps best-known IDTA submodel, *"Digital Nameplate"*, for example, is to provide information about the manufacturer and serial number of an asset. The other submodel templates cover use cases at a similar level of abstraction.

The main concept is the `Asset Administration Shell`, which represents the entire AAS of the asset. It can consist of several submodels represented by the class `Submodel`, which consists of abstract `Submodel Elements` that either are of a specified subtype, such as `File` or `Property`, or a composite of `SubmodelElements` themselves.

[3] Example call: https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/horizon-cl4-2023-twin-transition-01-04.

[4] IDTA partners: https://industrialdigitaltwin.org/en/about-idta/members-idta.

[5] ITDA AAS submodel content-hub: https://industrialdigitaltwin.org/en/content-hub/submodels.

[6] ECLASS website: https://eclass.eu/en/eclass-standard.

The abstract class `DataElement` inherits from the class `SubModelElement` from which further classes inherit. The classes `Property`, `Range`, `File`, and `ReferenceEleme` represent attributes of the asset. In the class `Property` an information pair consisting of one value and the value data type are defined. A `Range` consists of two values and the data type of the values. A `File` represents the type and location of a file, whereas `ReferenceElement` defines a logical reference either to another element of the AAS it is included in, or to an element of another AAS.

The communication capability of AASs is differentiated into passive, active, and I4.0-compliant communication capability–a distinction originally started by the Association of German Engineers (VDI) [91]. Passive AASs are also referred to as Type 1 AAS. Exchanged on a file basis, these Type 1 AAS contain static master data for an asset. Master data could for example be a serial number, as in the *"Digital Nameplate"* introduced previously, but also physical dimensions, material properties, or electrical power consumption. The Type 1 AAS, hence, consists of serialized files representing the asset and can be exchanged manually among engineers. Its data model is defined by the AAS metamodels. Those metamodels are specified in different templates which the IDTA provides to construct the AAS for different purposes.[7] These templates describe the structural composition of the submodels and the relationships between AASs. This enables domain experts to provide their knowledge in an ordered manner enabling expandability and modularization. As a passive entity, the Type 1 AAS does not have any automated data flow from or to its asset. Thus, the information in the Type 1 AAS describes asset types and instances as-designed without any real-time updates.

In addition to the serialized files of the Type 1 AAS, for Type 2 AASs an API is provided for the interaction with other components. Through this API, the Type 2 AAS can become service-oriented and reactive, *i.e.,*it may consider any information provided by the asset or any third-party software, *e.g.,* runtime information such as containing static and dynamic information about the asset. This establishes a mainly unidirectional data flow, in which live data is added to the models of the Type 1 AAS and where external methods, services, or tools can represent and relate this data.

Mutual understanding seems to be that the Type 3 AAS is an extension of Type 2 AAS that enables the AASs to perform I4.0-compliant communication according to the "Industrie 4.0 Language" [92] and therefore allows vertical integration into other AAS instances, this is referred to as Type 3 AAS. Moreover, the Type 3 AASs can feature algorithms to control the represented asset (to some extent), as well as to analyze the data and manipulate its models. Type 3 AASs thus extend

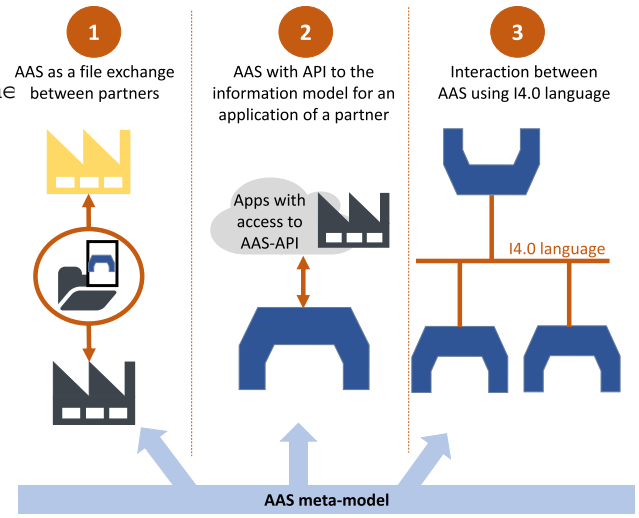

Fig. 2 The three types of AAS according to Plattform Industrie 4.0 [80]

across all RAMI4.0 [44] layers (*i.e.,* from the asset over communication to the business layer and, therefore, are able to implement business processes on their own. Unlike Type 1 AASs and Type 2 AASs, the Type 3 AAS has not yet been fully specified but is subject to ongoing research [48, 84].

The admin-shell-io package[8] by the IDTA can be considered as a reference implementation for managing administration shells. The package consists of the AASX[9] package explorer, with which administration shells can be created, edited, and visualized, but also includes the AASX server as infrastructure for deployment. If the AAS is to be integrated into a software project, the Eclipse BaSyx[10] project provides an open-source I4.0 middleware. BaSyx is available under the MIT license and offers an SDK for implementing AASs in Java, Python, and RUST.

3 Motivating example

To better illustrate how AASs are used to store information about an asset, we provide an illustrative example for an asset from the manufacturing domain and its representation in an AAS. Consider the following asset: A 3-axis milling machine (see Fig. 3), equipped with a Beckhoff TwinCAT control system from previous research [51, 95], for precision cutting, drilling, and shaping of materials. Imagine a block of aluminum positioned on the machine's work table. The operator programs the machine to create a complex part, such as a gearbox component. The spindle, which holds the

[7] IDTA AAS submodels list: https://industrialdigitaltwin.org/en/content-hub/submodels.

[8] admin-shell-io: https://github.com/admin-shell-io.

[9] Eclipse AASX website: https://github.com/eclipse-aaspe/.

[10] Eclipse BaSyx website: https://projects.eclipse.org/projects/dt.basyx.

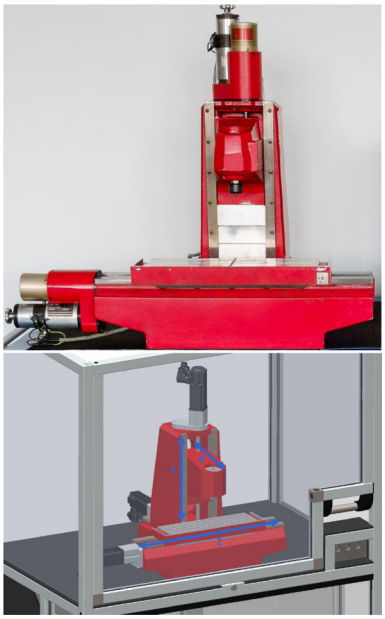

Fig. 3 The 3-axis milling machine for education and demonstration, known as OSACA (above) and its 3D model (below)

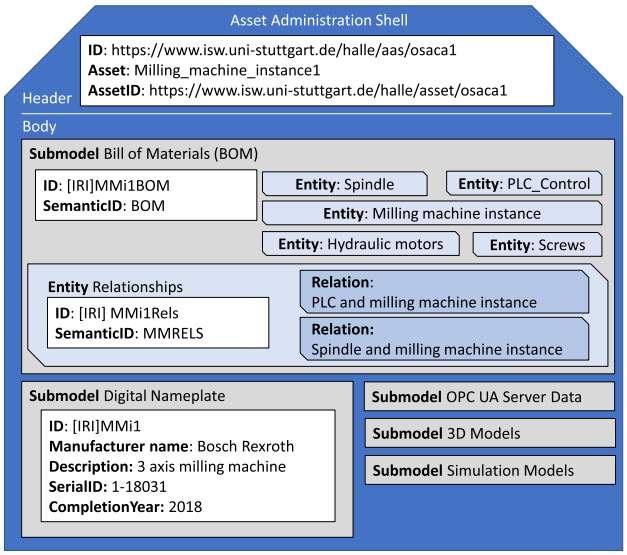

Fig. 4 Excerpt of the AAS of the 3-axis milling machine and its submodels

cutting tool, moves along the X-axis (front-to-back) and the Y-axis (up-and-down), while the table moves from left to right along the Z-axis, bringing the material into contact with the tool. As the spindle rotates the cutting tool at high speeds, it carves the aluminum to the specified dimensions and contours into a precisely engineered part. The milling machine contains hydraulic motors to enable movement along the axis, an interface to control the motors, and adapters to connect them physically to the milling machine. This setup further contains components such as fuses, switches, and a cabinet which are noted down in a bill of materials. For this machine, 3D models and simulation models that describe a milling process exist from its original engineering and the machine is capable of providing data at runtime via OPC UA specifications [58]. These data include, e.g., the position of the tool and the spindle speed.

We illustrate the AAS for this 3-axis milling machine in Fig. 4. The AAS consists of a header with an AAS identifier (ID), an asset ID, an asset description, and a body containing further information, which commonly is captured in different kinds of submodels, includes:

1. A *"Hierarchical Structures enabling Bills of Material"* (BOM) submodel template (IDTA 02011), which enables a hierarchical structure to represent the components of the milling machine. The BOM has relations to the described parts of the milling machine, *e.g.,* the relationship between the milling machine and the spindle. Described parts are entities of hydraulic motors, the spindle, the motor control, all screws, the showcase, and the milling machine as an entity.
2. A *"Digital Nameplate for Industrial Equipment"* submodel template (IDTA specification 02006), which contains an ID as well as the manufacturer's name, a description of the product, a serial number, and a year of completion of the machine.
3. An *"OPC UA Server Data Sheet"* submodel template (IDTA 02009), which contains an integration of the description of OPC UA servers. The IDTA template for this submodel is at the time of our research still in progress.
4. An *"Provision of 3D Models"* submodel template (IDTA 02026), that is able to provide 3D models of the machine. This includes both a 3D model and a simulation model.
5. An *"Provision of Simulation Models"* submodel template (IDTA 02005), that is able to provide simulation model files of the machine. The simulation itself remains in its specific exchange format and is linked in. The submodel contains further information about the type of simulation, on how to use the model and about the areas of application.

Formalizing the description of assets through well-defined submodels facilitates the automated processing of an AAS. If, for example, the *"OPC UA Server Data Sheet"* is fully specified and provided with assets, it allows reason about the self-configuration capabilities of the asset in a flexible factory. Moreover, being equipped with corresponding submodels, for instance, eases transferring assets from one entity to another: if the described machine is sold, the AAS could be made available to the future operator. This way, the new

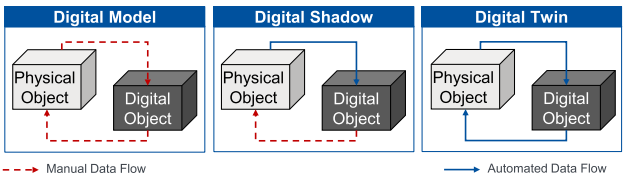

Fig. 5 Conceptualization of digital twins according to their data flows to and from the actual system [53]

operators would have access to simulation models that were created during engineering and could continue to use them, for example during a conversion. This further use is made possible in particular by the fact that (a) The uniform structure of the submodels means that all stakeholders are aware that a simulation model exists and (b) Not only the model is transferred, but also its original software and scope.

In order to provide a Type 1 AAS for the example machine the previously specified AAS is serialized including its submodels, static information such as relationships of the 3-axis milling machine are depicted. The provision of the AAS via Eclipse BaSyx, and therefore the provision via a functional interface, turns the Type 1 AAS of the illustrative example in Sect. 3 into a Type 2 AAS. The Type 2 AAS no longer solely describes the static information of the example machine, but also integrates sensor values via the OPC UA server. Static and dynamic information of the 3-axis milling machine can be accessed via the BaSyx Graphical User Interface (GUI).

4 Common requirements on digital twins

We introduce important conceptual models and frameworks describing digital twins–in general or in the context of manufacturing–from which we derive common requirements on digital twins to compare these with the AAS (*cf.* Sect. 5).

4.1 Popular academic definitions

The most prominent qualitative definition of digital twins distinguishes these from digital models and digital shadows [19] based on the automated data flows between the (cyber-)physical and digital object (*cf.* Fig. 5) [53]. Here, a digital object is considered to be:

- A `digital model`, if the data flows between both are manual, i.e., change on one side must be propagated manually to the other side. As such, the Type 1 AAS resembles the notion of digital models, as changes to the represented asset must be traced to the models of the AAS manually.
- A `digital shadow`, if the data flow from the physical object to the digital object is automated, i.e., changes to the physical object lead to changes in the digital object and in the opposite way, the data flow still is manual. This resembles to the notion of a Type 2 AAS, which, however, only requires that this unidirectional communication channel exists, but not that changes in the asset are traced automatically into the AAS.
- A `digital twin`, if both data flows are automated, i.e., if something changes in the digital object, this change is propagated to the physical asset and vice versa. As this only makes sense if there is some logic in the digital objects that can entail changes to the physical object, this conceptualization resembles Type 3 AASs to some extent.

Also, the authors hide much of the complexity of digital twins in the data flows: for instance, this definition already demands that the digital object can receive data from its actual (cyber-physical) system and send data back to it, *i.e.,* it needs to be a sufficiently complex software system that takes care of communication, synchronization, and digital representation. This also demands an interface to the actual system (*e.g.,* through OPC UA [34] or MQTT [56]), means to analyze the data, and user interfaces to control the behavior of the digital twin. Moreover, the model does not make explicit how frequently the data between the actual system and its digital twin must be exchanged (which means the digital twin might be asynchronous with the actual system for a long time). Also, how human decision-making can be incorporated, which often is necessary in manufacturing and other domains operating complex systems in reality, such as automotive or avionics, is not explained.

The 5D digital twin model extends the model of based on data flows [53] with the additional dimensions data, models, and services [88]. Here, a digital twin is a system that comprises elements of 5 dimensions: (1) Tthe AS itself, (2) Data from and about the AS, (3) Models of the AS as well as models of the digital twin, (4) Services about the AS (*e.g.,* predictive maintenance, reporting), and (5) Connections between the elements of these dimensions (cf. Figure 6).

This model assumes that all connections between all constituents are bidirectional, i.e., services can directly read data from the actual system and send commands to it as well. This allows making changes to the actual system without informing the digital twin. Moreover, both the services and the actual system can interact with data and models of the digital twin independently, which allows for introducing inconsistencies between the real-world system and its representation. Moreover, this model does not prescribe any minimally required models, data, or services.

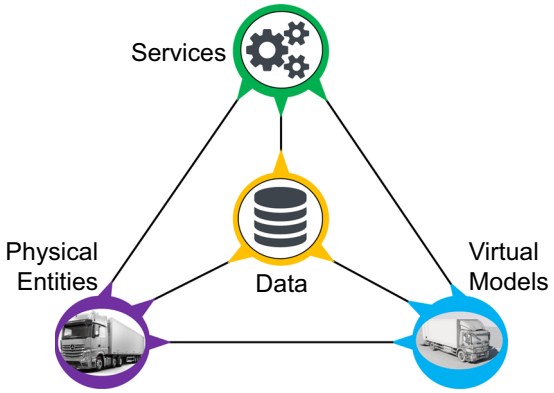

Fig. 6 Conceptualization of digital twins according to their constituents and data flows [88]

4.2 Popular Industrial Definitions

The Digital Twin Consortium (DTC) devised a list of potential capabilities of digital twins [11] according to six different categories (*cf.* Fig. 7). According to this digital twin periodic table, digital twins can provide (1) Data services, (2) Integration, (3) Intelligence, (4) User interaction, (5) Management functions, and (6) Trustworthiness capabilities. Data services (1) Contain data management processes, such as acquisition and ingestion, data interpretation with ontologies, data management with a model repository, data management methods such as pub/sub, batch processing, and data aggregation. Integration (2) Contains integration methodologies, such as platforms, APIs, and enterprise systems for both directions of abstraction more coarse, as well as more detailed systems. Intelligence (3) Describes components and services that encompass reasoning, planning, machine learning, simulation, recommendation systems, and any other capabilities being considered intelligent. User interaction (4) Contains all features the system provides for the monitoring, consisting of visualization of data, models, relationships and interaction, consisting of gamification, BPM, workflow and business intelligence methods without influencing the machine. Management functions (5) Are mostly directed to the manipulation of the system. Here for the device, the logging of the system and the data methods are advised to be used for interpretation and configuration of machines. Trustworthiness capabilities (6) Contain all features important for the trustworthiness of the system.

Based on these observations, the DTC proposes an architectural framework of building blocks for creating digital twins for different applications as depicted in Fig. 8 [32]. Its main components are a virtual representation on top of an IT/OT platform and the service interfaces of this representation. The virtual representation consists of data ("stored representations"), models, and functions for their integration. Taken this to the illustrative example from Sect. 3: the models of the milling machine can consist of multiple different models. These models could be designed by providing SysML in addition to 3D models in a 3D environment for simulation purposes. It provides interfaces (1) To synchronize with the real world, (2) With external data sources, and (3) To deploy services leveraging the virtual representation. The IT/OT platform includes orchestration middleware, networking infrastructure, APIs that provide access to services, and integration representation/functions for data storage and data transformation. This middleware could be manufacturing-specific management systems or even systems composed of data management services such as a database, a network interface such as OPCUA, or local platform-specific APIs from cloud platforms such as Azure.

Another main part of the architectural framework are services for privacy, physical and cyber-security, safety, resilience, and reliability. For example, the milling machine could provide safety measurements for its condition. In milling processes, it is often not safe for a worker to open the cabin in which a mill is located. Other metrics such as resilience and reliability can be measured based on KPIs. Finally, security and privacy are often addressed at a higher level of abstraction. The network within a factory must be secure from outside attacks to prevent malicious use of machines within the factory.

The Digital Twin Framework for Manufacturing as depicted in Fig. 9 (ISO 23247) [47] defines a conceptual digital twin framework for manufacturing. The framework consists of three layers of components that provide functionality on top of observable manufacturing elements (OMEs), which are items providing observable properties (*e.g.,* staff, a manufacturing plant, a robot) defined through existing standards from the manufacturing domain. The device communication entity's bottom layer comprises functional elements (FEs) to collect and process data from OMEs, as well as to actuate and control the OMEs. An example of such FEs could be a system that uses cameras and distance sensors for monitoring. Such a system could accurately determine the position and condition of a workpiece under the mill. The digital twin entity middle layer uses device communication to represent, manage, operate, simulate, and maintain the devices observed and controlled through the OMEs. For example, the distance and camera inputs are collected, processed, and forwarded to the digital twin entity. This entity contains models and analysis methods based on simulation models of the mill to predict a failure or malfunction. It could also provide an interface for updating the models, such as the path the mill needs to move along or the drilling speed and settings of the environment.

Through the user interface entity top layer, users and additional services can leverage the digital twin. For example, the

[11] DTC capabilities table: https://www.digitaltwinconsortium.org/initiatives/capabilities-periodic-table/.

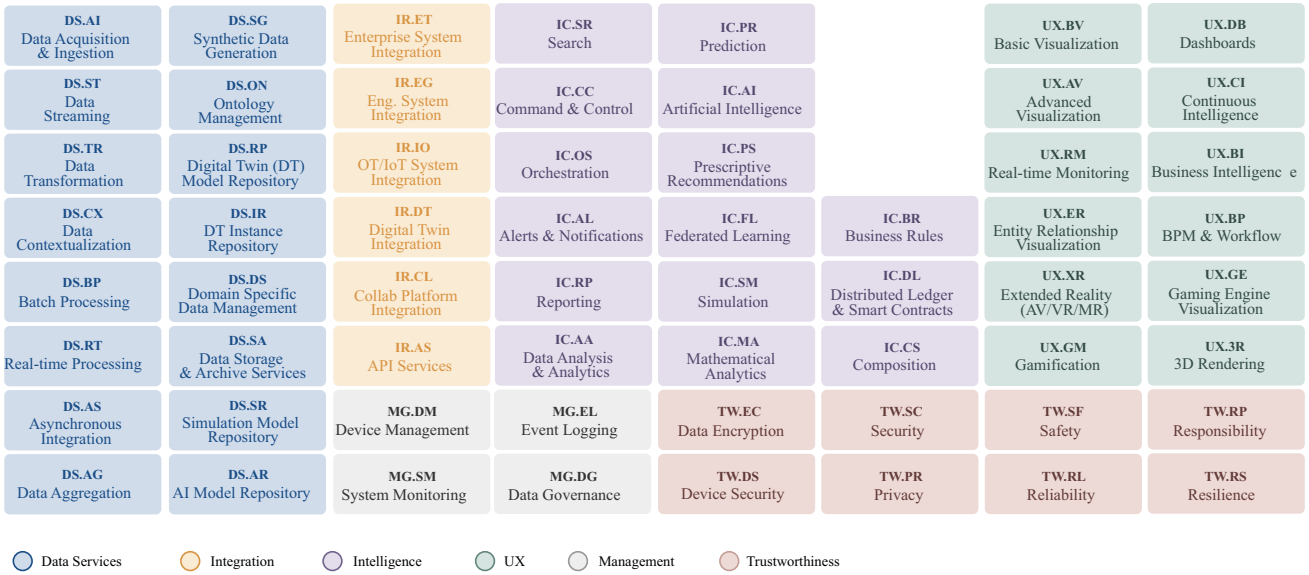

Fig. 7 Potential capabilities of digital twins according to the Digital Twin Consortium[9]

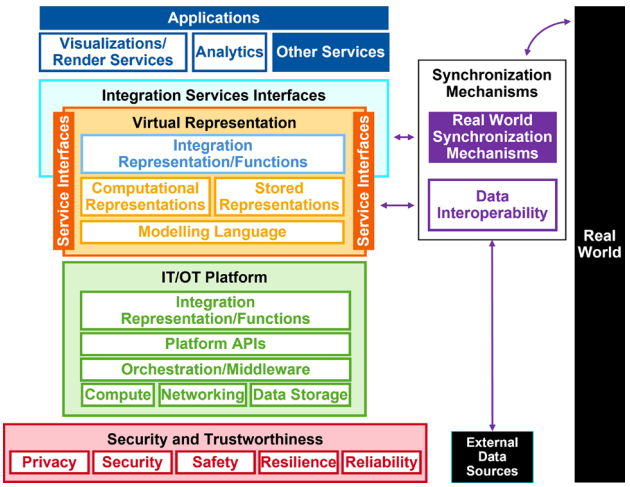

Fig. 8 The DTC platform stack architectural framework for digital twins [32]

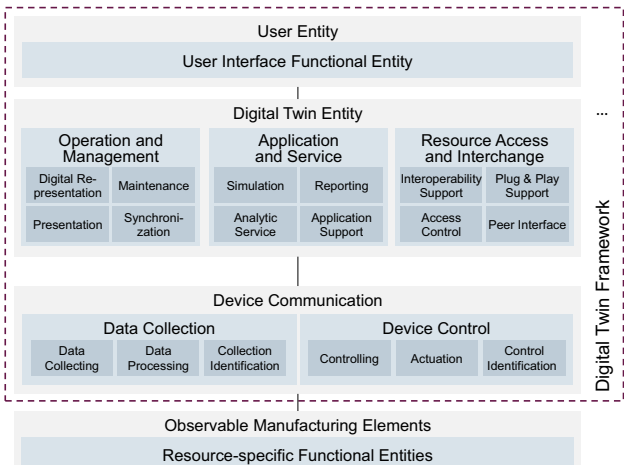

Fig. 9 Conceptual model of digital twins for manufacturing in ISO 23247 [47]

interpreted data and model of the machine could be visualized at run time for maintenance without stopping the machine during milling. In this way, downtime for maintenance can be reduced and the behavior of the mill can be optimized under the supervision of a machine expert.

4.3 Common requirements on digital twins

We have analyzed the conceptual frameworks for digital twins introduced above and identified capabilities that are required by (1) At least one of the frameworks, (2) At least two of the frameworks, and (3) All of the frameworks. Table 1 summarizes our findings, where each row represents an identified capability. To this, the first column assigns an identifier to the capability, the second column briefly describes the capability, the third column describes which framework it was derived from, and the last column details its context.

From this analysis, it also follows that the commonalities of the *all* investigated models, *i.e.,* the essence of a digital twin based on its potential capabilities, consist of (1) Retrieving data from its counterpart (**R01**), (2) Sending data to its counterpart (**R02**), and (3) Digitally representing its counterpart (**R04**). Additional commonalities between some of the investigated models arise from considering requirements that at least two of them have in common: This, for instance, includes having a user interface (**R03**) and, according to [47, 53], the synchronization of properties between AS and the digital twin (**R05**) and its manifestation (cf. [70]).

Table 1 Common Requirements for Digital Twins

Req.	The digital twin…	Sources	Context
R01 (Asset Receiving)	Can receive data from its twinned counterpart	All	This capability can have the form of automated data flows from the twinned system to the digital twin [53, 88], dedicated data collection components [47], or data ingestion functionalities [31]
R02 (Asset Sending)	Can send data to its twinned counterpart	All	This capability also is foundational to all investigated digital twin models
R03 (GUI)	Has a user interface	[31, 47]	The form of the UI is generally underspecified [47] but could range from basic visualizations to virtual reality [31]
R04 (Representing)	Can represent its counterpart digitally	All	Either through data or models. This does not entail requiring a user interface (see R03)
R05 (Synchronizing)	Can synchronize (selected) properties with its counterpart	[47, 53]	This is vital for the definition data-flow-based definition of digital twins [53] and made explicit by requiring a synchronization component according to [47]
R06 (Reporting)	Can report information to selected recipients aside from the AAS, *e.g.,* by sending a message to the asset's operator	[31, 47]	Using unspecified reporting capabilities [31] or a reporting component [47]
R07 (Twin Communication)	Can communicate with other digital twins	[31, 47]	Either through unspecified integration means [31] or a dedicated peer interface [47]
R08 (System Interaction)	Can interact with third-party systems *e.g.,* a manufacturing execution system or an ERP system	[31, 47]	This can have the form of dedicated interoperability support components [47] or of interfaces to external data source [31]
R09 (Added Value Services)	Provides services to act on data and models	[31, 47, 88]	Much of the added value functionality of a digital twin is very specific to the AS or the processes on the AS, i.e., it can hardly be generalized. Instead, [47] and [31] propose that digital twins yield services that realize this added value functionality specifically tailored to their use cases
R10 (Reasoning)	Can reason about data from/about the twinned counterpart as well as about data obtained from other systems (*cf.* R08, R09)	[31, 47]	To enable various kinds of such reasoning, the different frameworks propose specific analytics services [31, 47]

Other commonalities (see [31, 47]) include different ways of communication, including the need for reporting information to selected recipients (**R06**), communication with other digital twins (**R07**), and interaction possibilities with third-party systems (**R08**). Even though **R07** and **R08** are both requirements related to providing communication functionality, we made this differentiation on purpose: It makes a difference if we communicate with an application where we know the structure of the software system and might influence it (white-boy system), i.e., a digital twin created with a similar technology or using AAS, or an application, where we can only rely on the provided APIs of a third-party system (black-boy system), i.e., Manufacturing Execution Systems, or Enterprise-Resource-Planning Systems. More requirements are related to the functionalities that digital twins provide: this includes the need for different kinds of services [31, 47, 88] (**R09**) to act on data and models (some of them including detailed examples), as well as to explicitly describe the need for reasoning and analytics on data [31, 47] (**R10**). Clearly, **R09** could be broken down into more specific categories of services, e.g., monitoring, optimization, prediction, visualization (cf. the purposes of DTs in the systematic mapping study on software engineering for DTs [26] or the DT capabilities in Fig. 7), however, what services a specific digital twin needs is strongly dependent on the purpose one wants to create a digital twin for. Considering this requirement from a software architecture perspective, all services require data and/or models as input, process this information to gain new knowledge, and produce an output. Therefore, we summarize them into one higher level requirement.

5 Engineering digital twins with the AAS

This section explores possible methodologies for deploying each type of AAS and contrasts these approaches with find-

```
01 <environment xmlns="https://admin-shell.io/aas/3/0">
02   <assetAdministrationShells>
03     <assetAdministrationShell>
04       <id>https://www.isw.uni-stuttgart.de/halle/aas/osaca1</id>
05       <assetInformation>
06         <assetKind>milling_machine_instance1</assetKind>
07         <globalAssetId>https://www.isw.uni-stuttgart.de/
08                                     halle/asset/osaca1</globalAssetId>
09       </assetInformation>
10     </assetAdministrationShell>
11   </assetAdministrationShells>
12 </environment>
```

Fig. 10 Example XML instance of the milling machine

ings from earlier digital twin research. Our objective is to assess how each AAS type relates to the requirements outlined in Sect. 4. We aim to pinpoint the present status of AAS development and discover methodologies for constructing digital twins with the AAS. This will not only shed light on the practical applications of AAS in the engineering of digital twins but also contribute to the broader discourse on enhancing digital twin technology through the integration of the AAS. The differences between the three types of AAS from [79] are depicted in Fig. 2.

5.1 Type 1 asset administration shell

The need for Type 1 AAS is grounded in its file transfer as defined by IDTA [80]. As such, Type 1 AASs exist as serialized files, such as XML or JSON formats *cf.* Fig. 10). Here we see the set of AASs beginning (ll. 2 ff) and the definition of the milling machine (ll. 3-9), which contains the attributes from the illustrative example of Sect. 3 (ll. 4-7).

These serialized shells encapsulate static information and can be shared among partners of different companies. This file distribution follows a template,[12] which outlines the structural relations between the contained submodels.

Considering the illustrative example from Sect. 3, the CAD model, the simulation models, and models of the milling machine are represented as submodels in the AAS as depicted in Fig. 4. Submodels can be identified by using BaSyx [50] to iterate over the relations established between the elements of the AAS and the linked submodels. However, BaSyx does not support parsing these submodels. For instance, a CAD model can be identified through the bill of materials, in which the relations and the instances are referenced in our example in Fig. 4. Further, the bill of materials also links the submodels of the spindle, the milling machine, the frame, and the simulation models. Another aspect is the connection to further AAS. These are also denoted in submodels which contain the relation as an ID, a description, and a value that contains the concrete address of the server. Comparing this with the digital twin requirements of the previous section, we notice that an explicit service component is missing and that the physical entities are depicted as an asset without concrete reference to the connection between the AAS and its asset depicted in blue in Fig. 11. In general, the Type 1 AAS can be used as a knowledge base, in which a dedicated submodel may contain relations between submodels. Such relations can also be noted in descriptive files. When implementing a connection between a digital twin and Type 1 AAS, the necessary interface includes an implementation, *e.g.,* BaSyx, to identify the submodels. Reasoning on top of the submodels becomes a feature that the digital twin provides. Now we look at Table 2 and compare Type 1 AAS with the digital twin requirements from Table 1. First and foremost, the Type 1 AAS are serialized files without an active part. This leads to **R01**, **R02**, and **R03** being answered with a clear no. The Type 1 AAS cannot receive data from its twinned counterpart without an interface (**R01**) *e.g.,* BaSyx. This also means that it cannot send data back to its twinned counterpart (**R02**). There is also no explicit user interface provided off-the-shelf (**R03**). The Type 1 AAS represents its counterpart digitally (**R04**), *e.g.,* the serialized files, the relations, and documentation describe static properties of the asset. It cannot synchronize properties with its counterpart, as the data have to be managed manually (**R05**). This also means that it cannot actively report information (**R06**), not actively interact with other AAS (**R08**), provides no services to act on data and models (**R09**), and cannot reason about data (**R10**). The Type 1 AAS can be integrated with other AAS by adding information on the relation with further shells in submodels. However an active behavior as communicating with other AAS is not possible (**R07**). In conclusion, we observe that Type 1 AAS can be used as a knowledge base, while the digital twin has to provide means to analyze and reason on the knowledge.

5.2 Type 2 asset administration shell

Type 2 AAS addresses a prevalent challenge in the current Industry 4.0 landscape: inconsistency in communication protocols between tools provided by various providers [37]. Thus, in contrast to pure file sharing, as with the Type 1 AAS, Type 2 AASs [9] are realized as an executable software system *e.g.,* on a server. To read and write AAS files (and submodels), APIs are provided. Such Type 2 AASs can be realized with *e.g.,* Eclipse BaSyx or AASX as mentioned in Sect. 2.

In our example, this implies utilizing the submodels defined for a Type 1 AAS as the foundation for the Type 2 AAS's repository as illustrated in Fig. 11. The repository can for example be equipped with a REST API [37], enabling access to a database. This database could facilitate the retrieval of information by providing shells on top of assets. In a centralized database, it is crucial to manage access in an industrial setting to safeguard the confidentiality and

[12] https://industrialdigitaltwin.org/en/content-hub/submodels

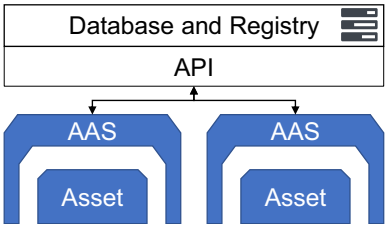

Fig. 11 Instance for a Type 2 AAS with connection between a centralized database and AAS [37]

consistency of data and models across multiple submodels and AASs. Thus leading to increased importance of data and model consistency methods. However, another possible way to implement this could be an interface that provides access to Type 1 AAS. Typically, operations performed using these shells include a name (the name is used as an unique identifier for accessing the operation), a semantic reference (a link to the metamodel of the operation - in general a link to a submodel), an explanation (general information of the operation, *e.g.,* a textual description of the functionality of the operation), input and output parameters. In the illustrative example, the application of a change to the PT, specifically the spindle of the milling machine, will affect the submodels of the spindle as well as all submodels which it is referenced in. To further illustrate, consider the type of the spindle, which changes from a high precision spindle to a low precision, high power spindle. Given the fact that the AAS is unable to detect changes on its own, the change of a spindle of unknown origin and type implies the need for a manual configuration of the submodel. Regarding the requirements on digital twins, similar issues are observed as in the Type 1 AAS, with the exception of improved connectivity due to its reactive nature.

Comparing the Type 2 AAS, as defined by IDTA in the specifications [46], with the requirements for DTs from Table 1, we observe the following (*cf.* Table 3). There is a discrepancy between the specified properties a Type 2 AAS has and the actual implemented properties. In the table, the fully filled circles represent the specified properties, the partly filled circles represent properties which are not specified but suggested by context, and the empty circle denote undefined requirements. Section 5.4 investigates the implementations and their properties in greater detail. The core of the Type 2 AAS according to the specifications provides an API with create, read, update and delete operations for managing submodels, so it can passively receive data from the asset (**R01**), but it cannot actively send data back to the asset. This means for **R01**, that it is fulfilled. It is possible to propagate values to submodels to influence the system's behavior, but actively sending data is not possible (**R02**). In general, the specification enables a user interface (**R03**), but it is not specified nor suggested by IDTA. As the Type 1 AAS is part of Type 2 AAS, the actual system is represented digitally (**R04**).

The requirements (**R05**–**R10**) depend on the functionality of the actual implementation of the specified API to realize the reactive nature of the Type 2 AAS. We briefly discuss these requirements and provide further insight in Sect. 5.4. A Type 2 AAS, by its very nature, is unable to synchronize data

Table 2 Type 1 AAS comparison with digital twin requirements

Req.	Eval.	Explanation
R01 (Asset Receiving)	○	The serialized files require an interface to write data on them. The Type 1 AAS is used like a file system
R02 (Asset Sending)	○	The Type 1 AAS collections of related serialized files without any behavior
R03 (GUI)	○	There is no off-the-shelf GUI for Type 1 AASs
R04 (Representing)	●	The information on the asset can be stored inside the serialized files with their relations
R05 (Synchronizing)	○	Synchronization between AAS and asset is not included for Type 1 AASs
R06 (Reporting)	○	There are templates supporting specifying recipients *e.g.,* other AAS aside from itself, in the Type 1 AAS, but no data exchange mechanism is implemented
R07 (Twin Communication)	○	The recipient specification mechanism of **R06** (Reporting)could be used as the base for any twin to twin communication. However, there is not off-the-shelf support for such
R08 (System Interaction)	○	There is no active behavior in Type 1 AAS
R09 (Added Value Services)	○	The interface for interacting with the AAS is implemented in BaSyx, but the Type 1 AAS does not provide services by default
R10 (Reasoning)	○	Since neither **R08** nor **R09** is fulfilled the AAS Type 1 cannot actively reason about data from the twinned counterpart

○ The requirement is not required by IDTA
◑ The requirement is suggested by IDTA
● The requirement is defined by IDTA

Table 3 Type 2 AAS comparison with digital twin requirements

Req.	Eval.	Explanation
R01 (Asset Receiving)	●	The Type 2 AAS is capable of passively receiving data from the Asset or multiple AAS through an API
R02 (Asset Sending)	○	The Type 2 AAS cannot actively send data to the twinned counterpart
R03 (GUI)	○	The Type 2 AAS requires an implementation for the API. In relation to the implementation the API may of may not be a full user interface, which is not specified by the IDTA
R04 (Representing)	●	The Type 2 AAS can represent its counterpart digitally through the submodels and their relations
R05 (Synchronizing)	◑	The Type 2 AAS as specified by the IDTA comes with a possibility of defining a timer for the update of data within the AAS. The active synchronization relies on implementation
R06 (Reporting)	◑	The Type 2 AAS is also not capable of actively reporting information to selected recipients, but through references between submodels it may propagate data to further Type 2 AAS
R07 (Twin Communication)	○	The Type 2 AAS can adjust relations between properties in submodels and refer to further Type 2 AAS submodels. Thus it cannot perform data exchanges with other DTs
R08 (System Interaction)	○	The Type 2 AAS can indirectly interact with other systems by defining references to further submodels of other AAS. Overall the Type 2 AAS cannot interact with other systems
R09 (Added Value Services)	◑	The Type 2 AAS provides an API for manipulating submodels. Whether these API are sufficient enough to be a service is dependent on the actual implementation
R10 (Reasoning)	◑	The Type 2 AAS cannot actively reason about data from the twinned counterpart, but *e.g.,* there can be references defined inside submodels calculating metrics for evaluating the throughput or performance. Further the IDTA defines an option with regex queries to reason on data. This again depends on the actual implementation

○ The requirement is not required by IDTA
◑ The requirement is suggested by IDTA
● The requirement is defined by IDTA

between AASs. However, the IDTA specification suggests a solution in which a timer can be defined for data update intervals (**R05**). While the specification does not explicitly address the communication and interaction between digital twins, an integration with another Type 2 AAS is possible and suggested, by referencing the submodels and APIs of the AAS (**R06**). A reference could enable a communication between DTs and the interaction with other systems (**R07** and **R08**) by using the references a way of data access from a shared repository, as shown in Fig. 11. Since these features have to be active communication, where data is sent back and forth between DTs, the **R07** and **R08** are not fulfilled. In regard to the provided services, when deploying the API as a REST-service it is possible to provide functionalities. As specification for Type 2 AAS ends with the requirement of an API. A service to access the API's functionalities is suggested by IDTA, but also (**R09**)dependent on the actual implementation. Finally, the requirement for active reasoning is suggested in the specification, where references between submodels alongside regular expressions could be used for drawing conclusions on data (**R10**).

5.3 Type 3 asset administration shell

As the Type 3 AAS is not yet fully specified and still under ongoing research, we base our analysis on the research concepts. The Type 3 Asset Administration Shell extends the Type 2 AAS with additional features. It has an active behavior in addition to the ability to communicate and negotiate on its own [48, 80, 84]. This means it contains data-transforming functions, can obtain and transform/abstract data autonomously (*e.g.,* for analyzing purposes), and is also able to act upon the AS on its own. For this bidirectional communication with the asset, the Type 3 AAS uses a well-defined I4.0-Language [8].

Others discuss the reactive components of a Type 3 AAS and describe a possible architecture of such a system [43]. This architecture comprises a passive and an active part of the AAS. The passive part is already discussed for the Type 1 and Type 2 AAS. The active part contains algorithms managed by a component manager and scheduled by an interaction manager. All these can interact with the environment via a messenger component that communicates using the I4.0 lan-

Table 4 Type 3 AAS comparison with digital twin requirements

Req.	Eval.	Explanation
R01 (Asset Receiving)	●	The Type 3 AAS has an active communication with its asset
R02 (Asset Sending)	●	Via the I4.0 language, the Type 3 AAS can send data to its asset
R03 (GUI)	◑	With its active behavior, the Type 3 AAS could implement and host a graphical user interface
R04 (Representing)	●	The Type3 AAS can represent its counterpart digitally through the submodels and their relations
R05 (Synchronizing)	◑	Via the I4.0 language, the Type 3 AAS can send synchronizing commands to its asset
R06 (Reporting)	◑	The Type 3 AAS comprises active components that could implement sending reporting information via the I4.0 language
R07 (Twin Communication)	●	The Type 3 AAS comprises active components and defined interfaces, so that it can communicate with other DTs for specific submodel purposes
R08 (System Interaction)	●	The Type 3 AAS comprises active components that could implement communicating with other systems for specific submodel purposes
R09 (Added Value Services)	◑	Within its active behavior, the Type 3 AAS is supposed to (autonomously) compute data, *e.g.,* for analysis purposes. But the extent is not defined by the IDTA
R10 (Reasoning)	◑	Within its active behavior, the Type 3 AAS is supposed to (autonomously) compute data, *e.g.,* for analysis purposes. But the extent is not defined by the IDTA

○ The requirement is not required by IDTA
◑ The requirement is suggested by IDTA
● The requirement is defined by IDTA

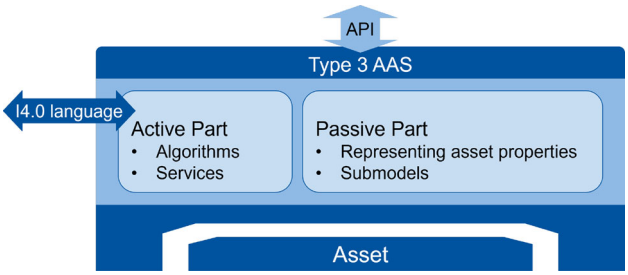

Fig. 12 Active and passive part of the Type 3 AAS. Inspired from [43]

guage. In this section, we call the single algorithms together with the interaction and component manager *services*. This concept is then again refined by Mediavilla et al. in [60]. They describe a conceptual architecture for engineering change management with a Type 3 AAS. Here, the AAS contains an API and again active services.

Ultimately, a Type 3 AAS contains a passive part, already present in a Type 1 AAS and a Type 2 AAS as well as an active part (cf. Table 4), illustrated in Fig 12. This active part contains reactive and proactive services that implement the autonomous behavior required by the IDTA.

With the Type 3 AAS extending the capabilities of the Type 2 AAS with additional I4.0-compliant communication and active behavior, we take **R01** (Asset Receiving) and **R04** (Representing) as been fulfilled by each Type 3 AAS. Therefore, we concentrate on the requirements which were for the Type 2 AAS not fulfilled and for digital twins required, namely requirements **R02**, **R03**, **R07**, and **R08** and the already implemented digital twin requirements **R05**, **R06**, **R09**, **R10**. We refer to the reference architecture from [43], to discuss which digital twin requirements must be fulfilled by the Type 3 AAS and which are possible or likely to be implemented. With the ability to autonomously send data to its associated asset, an implementation of the Type 3 AAS fulfills **R02** (Asset Sending). This data comes in the format, the asset is capable of processing. In most cases, this might be control commands to an actual system. The AAS uses the I4.0 language to communicate with the asset. **R03** (GUI) describes that a twin needs to have a user interface. According to [89], a user interface is an "interface that enables information to be passed between a human user and hardware or software components of a computer system". The Type 3 AAS offers an HTTP interface which can act as a remote interface when logging in via a remote console. To this point, the AAS does not require to have a graphical user interface by itself but can produce data that can be interpreted as graphical elements. Although, referring to the definition given in [89], the HTTP interface together with a remote machine, interpreting the communication, might act as an interface, the Type 3 AAS does not come with a GUI out-of-the-box. Therefore, we state that R03 is possible but not defined by the IDTA. **R05** (Synchronizing) asks for the capability to synchronize (selected) properties with its counterpart. Those properties

therefore must be present in the asset and in the AAS. Synchronization goes bidirectional. With its ability to receive data from the asset, the AAS is already capable of mirroring properties and property changes in the AAS, either by being notified or by pulling the required information. The Type 3 AAS is capable of bidirectional communication with its asset. Therefore the AAS can send control commands to the asset which alter the asset's properties. A synchronization service could detect all changes in the AAS properties to be synchronized and send according to control commands to its asset. However, the synchronization must be triggered, either by some AAS functionality or by a user. Manual user synchronization can be triggered via the user interface and autonomous synchronization of asset properties can be done by a monitoring service. With this, the Type 3 AAS does not fulfill **R05** out-of-the-box, but the implementation of it is possible and very much likely to utilize its abilities. **R06** (Reporting) describes the ability to report information. We already described how a service could autonomously synchronize data with its associated asset. In the same manner, the Type 3 AAS can synchronize reporting information with another party, either bidirectional or only from the AAS to the reporting recipients. So, the Type 3 AAS again is not out-of-the box capable of reporting information, but has all components to implement such a behavior. A Type 3 AAS is capable of an I4.0 language and at least some HTTP interfaces. Assuming the I4.0 language is processable by other DTs, a Type 3 AAS can communicate with other digital twins, fulfilling **R07** (Twin Communication), the ability to communicate with other digital twins. This communication is processed in the AAS active services for specific submodel purposes. Other 3rd party systems can communicate with the AAS via the HTTP interface or the I4.0 language, respectively, fulfilling **R08** System Interaction, the ability to interact with other systems. **R09** (Added Value Services) requires services to act on data and models, while **R10** (Reasoning) also requires services to reason about data from other systems. The architecture for a Type 3 AAS defines an active behavior and requires autonomous services. These act on asset data and potentially models that are not limited to asset data and models, but can also act and reason on data from other systems. The IDTA does not specify in which manner the active behavior interacts with the asset data or models, but it is very likely that the Type 3 AAS does act on asset data and reason about it, as the main use case in industrial environments.

In summary, our understanding is that a Type 3 AAS is capable of implementing all of the previously derived requirements for a digital twin. With its active autonomous behavior, it overcomes the shortcomings of the Type 2 AAS regarding our derived digital twin requirements.

5.4 Common implementations of AAS

Now we look at some implementations of AAS from industry and from research to provide a broad overview of possible functionalities in-between the specified AAS types in Table 5. A file transfer protocol, as *i.e.,* supported by FileZilla server ,[16] may be employed for the manipulation of Type 1 AAS. In its most basic form, the Type 1 AAS is a file stored on a local drive (**R04**). This approach is compliant with the Type 1 AAS as defined in the specification. However, as the file is unable to manage data in its submodels, it is not capable of receiving data, thus not fulfilling the requirement of being a Type 2 AAS.

The IDTA provides sample projects[17] and a server[18] hosting them. These projects fulfill the requirement for the asset to be represented digitally (**R04**) and are in accordance with the Type 1 AAS. However, as the file server is also unable to manage data in its submodels, the files are not capable of receiving data on their own, thus not fulfilling the requirement of being a Type 2 AAS. Only in combination with the server the minimal requirement of being a Type 2 AAS would be fulfilled. In conclusion, both the file transfer protocol and the sample projects, provide no further features and thus do not fulfill any other requirements.

The BaSyx framework provides an API to interact with the Type 1 AAS through a component called registry. The BaSyx framework further provides a database interface to persist the state of the Type 1 AAS. The BaSyx framework is conform to a Type 2 AAS. It is able to receive data from the asset (**R01**) and represent the asset digitally (**R04**). Functionalities such as sending data to the twinned counterpart (**R02**), active reporting to a selected recipient (**R06**), communication and interaction with other DTs (**R07** and **R08**), the provision of BaSyx as a service (**R09**) and the reasoning on data (**R10**) are open for future extensions. At this time, two features have been incorporated into BaSyx, which extend it from a Type 2 AAS. With the AAS Web UI provided by the BaSyx framework a user interface is added (**R03**). A *DataBridge* component enables a timed synchronization between the asset and the Type 1 AAS **R05**. Any further functionality can be built on top of this basis. This means that requirements for a Type 3 AAS are partly fulfilled, while mandatory ones such as **R07** and **R08** are not fulfilled. In research on the connection between Type 2 AAS and a database [37] the connection between the AAS and an API to automatically store data was researched (**R05**). With the involvement of multiple Type 1 AAS, connected to a single database backend a connection to selected recipients is shown (**R06**). Since the backend is independent from the

[16] Filezilla: https://filezilla-project.org/.

[17] AAS samples: https://www.admin-shell-io.com/samples/.

[18] Eclipse AAS package explorer: https://github.com/eclipse-aaspe.

Table 5 Implementations with regard to AAS specification conformness and requirement fulfillment

Reference	Type 1 AAS	Type 2 AAS	Type 3 AAS	Fulfills Requirements	Explanation
File Transfer Protocol (FTP) supported by FileZilla server	●	○	○	**R04**	A file transfer protocol can be treated as a simplified Type 1 AAS. Any action has to be done manually, thus meaning the AAS is a passive system
IDTA samples [15] without AASX server	●	○	○	**R04**	The submodel templates define the internal structure of a submodel. This also includes references across submodels. Thus a connection to other instances is also established
BaSyx	●	●	◑	**R01**, **R03**, **R04**, **R05**	The BaSyx framework is a prominent API for managing Type 1 AAS. Thus fulfilling the condition of being a Type 2 AAS
Research implementation by Evans et al. in 2022[37]	●	●	◑	**R01**, **R04**, **R05**, **R06**, **R08**	This publication focuses on the connection between AAS and a backend to persistently store data of the system. Thus automated data propagation and synchronization was implemented on top of the Type 2 AAS specifications
Eclipse Papyrus for Manufacturing	●	●	○	**R01**, **R04**	Eclipse Papyrus4Manufacturing is a modeling tool for Industry 4.0 applications, which provides an integration with BaSyx. Depending on the implementation and version of BaSyx, this tooling also fulfills more or less requirements

○ No requirements of the AAS type is fulfilled
◑ Some requirements are fulfilled
● The minimal requirements are fulfilled

Type 2 AAS an interaction between Type 2 AAS and a self-developed system is shown (**R08**). With the representation of the asset and the ability to receive data from the asset **R01** and **R04** are fulfilled. Since the focus does not cover the communication between DTs, any value services or reasoning on data the requirements **R07, R09** and **R10** are not fulfilled. Overall this research is a Type 2 AAS as per definition and adds features to it. In other research [7, 10, 22] the focus is shifted toward the modeling of submodels inside of the AAS with regard to the templates provided by the IDTA. While the modeling is also an important aspect, it remains unclear how a DT is supposed to interact with other DTs and systems. Consequently, it is not possible to find an answer whether our requirements are fulfilled. Eclipse Papyrus4Manufacturing[19] is a graphical modeling tool for industry 4.0 and provides integration with BaSyx. Thus also fulfilling the requirement of being a Type 2 AAS. As this tooling is also reliant on the implementation and specification of the user, fulfillment of the requirements **R05**-**R10** remain open.

[19] Eclipse Papyrus4Manufacturing: https://eclipse.dev/papyrus/components/manufacturing/.

6 Challenges and opportunities

We discuss challenges and opportunities for creating Digital Twins using AAS and model-driven development methods. Find more ideas on applicable methods within the Model-Based Software Engineering Body of Knowledge [20] and the Systems Engineering Body of Knowledge [85].

6.1 Efficient engineering of digital twins through their reuse

Digital twins are complex software artifacts that combine data, models, and services to provide better insights and added value on top of the twinned system. Consequently, their engineering is challenging and costly. With one of the main reasons for the success of software being its reusability (*e.g.,* classes and libraries in-the-small, apps and containers in-the-large), digital twins should benefit from efficient reuse mechanisms as well. Methods for the composition of language-independent software through superimposition [4] or the composition of event-specific context-dependent behavior through language constructs [5] are commonly

practiced for software. For digital twins, this is not the case. Although recent studies claim to compose digital twins, they commonly limit themselves to a virtual representation [3, 28].

Wherever a digital twin of a larger system (*e.g.,* a factory) logically consists of digital twins of smaller systems (*e.g.,* production machinery), reusing the smaller digital twin as part of the larger one should be as easy as reusing existing classes in another software [65]. Moreover, digital twins of specific systems should be transferable to similar systems easily, *e.g.,* allowing for systematic reuse (parts of) the digital twin of a sports car of one brand with the sports car of another brand. Through MDE methods a possibility opens of leveraging existing models of a digital twin for a transfer across the boundaries [17]. These methods focus on heterogeneous models, bidirectional synchronization, and the development throughout the system lifecycle. While also opening up new challenges such as the question for a modeling language, an architectural framework, inconsistency, model and data management. Currently, neither form of reuse is supported systematically, which is why creating digital twins still demands tremendous handcrafting of software artifacts, which hampers the adoption of digital twins.

Reuse for AAS is a built-in property through the use of AAS submodels. These define the underlying data structure and thus promise modularity and reusability. Therefore, for Type 1 and Type 2, different AASs of similar assets or even similar domains are built similarly. However, since Type 1 and Type 2 AASs have little or no software engineering, their development consists mainly of setting the right values. Type 3 AAS engineering consists of more sophisticated software engineering that defines their active behavior. This active behavior could be implemented, for example, by a service that works on data consistent with the data models of the submodels. This promises reuse of such services for different Type 3 AASs, since the input data looks the same. Composability of AASs from other smaller AASs may be possible due to well-defined interfaces, but the number of different submodels[20] and possible combinations of them, as well as their quality, make it difficult to predict the actual engineering effort.

6.2 Low-code configuration for AAS

Digital twins will be largely operated and configured by experts in the respective application domains. These rarely have received formal software engineering training, which limits the expressiveness of technologies that should be applied to configuring digital twins. Using low-code development platforms or low-code development approaches [30] for digital twin engineering can enable domain experts to create, configure, and operate tailored digital twins for their specific domain of interest [25] using their domain expertise, concepts, and terminology.

Similar approaches could be applied to ease the use of AAS to develop digital twins. This includes, *e.g.,* (1) Low-code interfaces to create, access, and reason over submodels, (2) Libraries for submodels supporting easy reuse, (3) Support for automated deployment of digital twins based on AAS technologies, and (4) Low-code modeling languages that enable orchestrating the services used by the digital twins. Moreover, if specific application domains select specific sets of AAS submodels and services that might be relevant for them, one could place a low-code configuration layer on top that enables the tailoring for specific use cases on a higher level of abstraction.

As these parts of AAS are technology dependent, the low-code development platforms have to be tailored toward specific technologies currently developed for AAS, *e.g.,*, the BaSyx framework, and other frameworks to be developed in the future. Establishing standards and standardized interfaces will help making low-code development platforms more technology-independent.

In addition, the concrete application domain may also have different requirements for a low-code platform coming from possible users, *e.g.,* which level of abstraction has to be provided for the intended users when describing configuration information. This covers the whole spectrum from (1) the type of representation, *e.g.,* text-based approaches, graphical interfaces, or wizards, (2) different levels of knowledge, *e.g.,* domain experts covering a broad spectrum of information about an asset to niche knowledge for a particular part of the asset, and (3) the depth of technology understanding, *e.g.,* without software engineering skills up to software engineering experts. Thus, low-code development platforms have to be tailored for their intended AAS user groups.

Such low-code development platforms could either be developed as open-source, *e.g.,* BESSER [2], or in-house if companies create digital twins for themselves regularly or sell them to customers. However, their core development will be easier if more AAS aspects are standardized, especially services to be used within Type 3 AASs and their communication interfaces within a digital twin implementation.

6.3 Derivation of AAS Digital Twins from engineering models

In manufacturing, a lot of information that would be interesting for digital twin creation with AAS, is already created during, *e.g.,* the engineering process of production machines, factory planning, and process planning (cf. “Representing Systems with Models” in the Systems Engineering Body of Knowledge [85]). With the SysML v2 release, it is also to be expected that more systems engineering models will be

[20] https://industrialdigitaltwin.org/en/content-hub/submodels.

available in an interchangeable format within the next few years. Reusing this information would be helpful for engineering digital twins [21]. This requires extraction of the information from these engineering models into information needed in submodels. This transformation could be improved by reusing the information already provided and standardized in submodels, as they could parametrize this transformation.

Moreover, up to now, it is unclear how information during runtime of an asset could be brought back to the development process of the asset [24] using AAS technologies. This requires adding a methodology and tools on top which enable the analysis of information relevant to improving the development. As the realization of Type 3 AAS is planned, such mechanisms could be integrated in the active part as algorithms. However, reuse of these functionalities is not considered if they are to be realized newly for every AAS.

There exists a large MDE body of knowledge on models@runtime [12, 18] which is specifically of interest for AAS Type 2 and Type 3. Models@runtime approaches can support the bidirectional synchronization between digital twins and their counterparts [16]. While different types of runtime models [12], *e.g.,* structure, behavior, quality, goal, requirements models, could be of interest for digital twins developed with AAS technologies, concrete examples, and their implementations are missing.

6.4 Communication between AASs

When it comes to connections between different digital twins, open challenges still remain [66], *e.g.,* how digital twins on different levels of details can be integrated or can exchange information on different levels of granularity. It still requires manual effort to map and integrate information between different levels. This problem arises in particular with AAS, as they claim to represent an AS over its entire lifecycle, which consequently results in integration challenges when ownership of an AS is transferred from one party to another [39].

Semantic challenges, such as inconsistencies, could be solved by using standardized submodels [37]. However, this might not be possible in all cases, as this requires forcing a certain view of the world on each application domain. Thus, mechanisms for translating or linking information from different submodels might be a more sensible way to go.

The technical connection between the AASs for Type 2 AAS is, depending on the Type 2 interpretation, based on established technologies, primarily REST via HTTP(S) and Open Platform Communication Unified Architecture (OPC UA) or is implemented in proprietary interfaces OPC UA is used in particular to drive convergence between Information Technology (IT) and Operational Technology (OT) systems and to ensure interoperability in manufacturing [71]. Standardized, Industry 4.0-compliant communication is part of Type 3 AAS [80], a specification is pending.

6.5 Non-functional requirements and the AAS

One limitation of our analysis of requirements is that it is purely based on functional requirements. This is due to two reasons: The existing approaches rarely mention non-functional requirements (*e.g.,* [32] mentions privacy, security, safety, resilience, and reliability), and such requirements are strongly connected to the application domain, *i.e.,* safety-critical domains require different aspects in their digital twins than other domains. This also includes the question of how to handle inconsistency or similarity. When creating digital twins for complex cyber-physical systems, we have to cope with the inconsistent behavior of these assets. There exist approaches for the explicit representation and treatment of uncertainty [29], that require uncertainty-aware controlling components in the digital twin. Other approaches enable us to measure the similarity between an asset and its digital twin [68].

MDE tackles some of these non-functional requirements [17], e.g., uncertainty modeling integrated in the digital twin development to explicitly cover it in the design phase, or the ability to integrate evolved asset information in the form of models. With the various submodels [21] regarding different non-functional requirements like *Functional Safety*, *Security Engineering*, *Reliability*, *Maintenance*, or ecological *Carbon Footprint*, the AAS covers the means to support storing information needed for fulfilling some of the non-functional requirements stated. However, how this data is used and played back to the actual system in an, *e.g.,* Type 3 AAS, remains to be defined and is up to the developer for now.

6.6 Evolvement of the actual system and the AAS

Engineering models, models@runtime and the asset we are creating a digital twin for could evolve. Thus, approaches for digital twin evolution need to be developed [27, 63] and adapted for digital twins we are creating with AASs. Currently, new submodel templates are developed which are a conservative extension of existing AAS submodels. Thus, for Type 1 AAS rather values in submodels are changing based on the changing reality. For Type 2 and Type 3, it could also affect the connection to the actual system and its control interfaces. If the Type 3 AAS autonomously computes data, *e.g.,* for analysis purposes, depending on the requirements for this analysis the need for data might also change over time.

There exists a variety of MDE approaches to support system evolvement, *e.g.,* for (meta-)model evolution, software migration, software reuse, and to check correctness and consistency during and after model evolution. However, approaches for the co-evolution of AAS across multiple

[21] https://industrialdigitaltwin.org/en/content-hub/submodels.

meta-levels including its submodels, models, and data are to be researched. Here, collaborating with researchers from data and process modeling [62] would be helpful to integrate MDE methods with methods from data and process engineering. In addition, the evolution of AAS services with models, and their data is a challenging open research aspect.

7 Related work

A systematic literature review analyzing the characteristics of current implementations of AAS includes papers published in English on Scopus between 2017 and 2021 identifies specific implementations of the AAS in 29 of the 45 analyzed publications [1]. The authors compare the usage of the terms AAS and digital twin in these 29 papers and found that 19 of the papers did not contain any statements on this. Out of the remaining 10 papers, 5 consider AAS and digital twin being synonymous terms, 3 publications consider the AAS as a concrete implementation of a digital, and 2 publications consider the AAS as the information model of a digital twin. A more detailed resolution, *i.e.,* based on criteria, is not provided. Others investigate how data-driven approaches, *e.g.,* machine and deep learning models, can be used for predictive maintenance in the industry, especially the automotive domain, and how they can be integrated into a digital twin represented using the AAS [81]. The study focuses on a medium-duty hydrogen truck, where six submodels are defined to support the predictive maintenance pipeline and four services use them, namely feature extraction, training, prediction, and maintenance.

Other research compares the expressiveness of a digital twin description framework with the expressiveness of the AAS [73]. The description framework describes digital twins through their usages, enablers, and models, where models are the information that enablers use to support certain usages. The description framework distinguishes 12 characteristics, including support for representing the system-under-study, the data to be transmitted, timing information, fidelity of the digital twin, lifecycle information, and more. For each characteristic, the authors investigate how Type 1 AASs enable their respective representation out-of-the-box. From this, the authors conclude that the AAS supports four characteristics fully, four partially, four implicitly, and two not at all (*e.g.,* "fidelity considerations" are unsupported). However, as the AAS generally is a container that can be populated with different submodels (cf. Sect. 3), the expressiveness of an AAS depends on the submodels used. For instance, if a submodel capturing fidelity information becomes available, the corresponding characteristics of the description framework might become expressible with an AAS. Consequently, that study must be understood as a snapshot of the expressiveness of AASs relative to available submodels at the time of the analysis. Another study compares the ease of implementation of Type 2 AASs using BaSyx [50] and Eclipse AASX. Based on their qualitative, anecdotal, observations, they conclude that the technologies are sufficiently accessible and mature enough to facilitate the implementation of Type 2 AASs [99]. Others compare the communication in the AASX and Eclipse BaSyx server implementation and show challenges related to the pulling of information [37]. The analysis proposes the realization of Type 2 AASs with an event-based server to enable seamless orchestration and deployment.

A systematic literature review [1] considers and analyzes the research on AASs in relation to manufacturing systems. The result is an evaluation of the emerging practical uses of AAS implementations within production systems. This review noticed the overlapping definitions between AASs and digital twins with a relationship that is not clearly defined. Further gaps emerge for AASs for simulation models and AAS for bidirectional data exchange between AAS and its asset. Other researchers employ Type 1 AASs as a knowledge base of implementations of digital twins [100], which is an obvious use of AASs for digital twins.

Further research [75] discusses the virtual representation of assets in an AASs using smart factory technology as an example. Their contribution consists of the distinction of four requirements for an asset for virtual representations. These requirements are

1. Provision of efficient information for creating a DT automatically with a library containing the configuration.
2. Vertical integration, which represents the characteristic that the DT can be operated and integrated based on one virtual representation from the asset layer at the bottom to the enterprise layer at the top.
3. Horizontal coordination represents the coordination between the DT and engineering applications *i.e.,* services.
4. The DT should derive performance indicators repeatedly with simulation, which is the core technical functionality of the DT.

The study includes applications to industrial vehicle production lines and a smart factory for producing samples and small-sized components using additive manufacturing. Here, the digital twin is treated in the form of a virtual factory technology with simulation as its core technical functionality. This limits their research into creating a definition of an asset without going into detail on the surrounding system and how the data is transferred between AAS and machine or simulation.

Others propose a high-level structured framework of eight steps for creating AASs in production environments [82] and evaluate this on the production of distributed high-rate electrolyzer. These steps include deriving requirements,

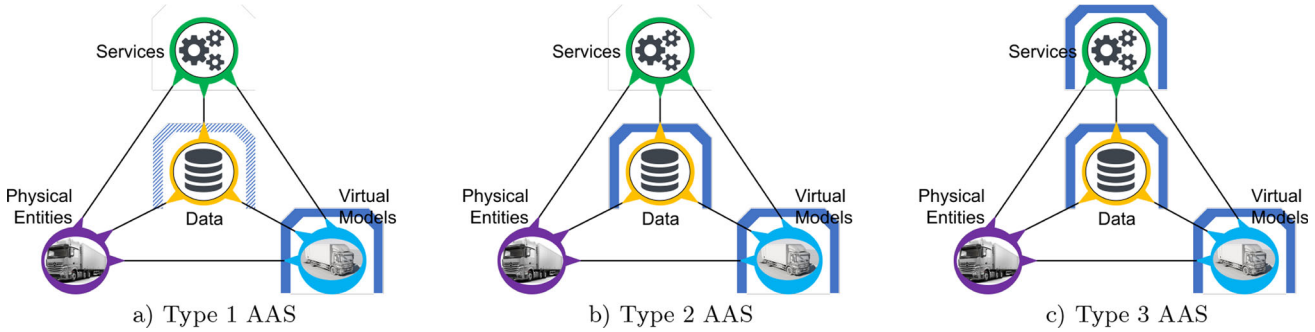

Fig. 13 The different types of asset administration shells cover different parts of digital twins (the DT representation is based on the 5D model by [88])

selecting assets, and creating the AAS with properties chosen according to existing standards. The goal is to guide the development of digital twins by constructing an AAS. In this work, the digital twin (technologies) are used to structure the digital representation of a distributed manufacturing systems, *i.e.,* they do not require any active behavior. In particular, the developed then AAS follows a Type 1 classification, emphasizing basic digital representations without active behavior.

Further research investigates using the concepts of AAS on the example of the lifecycle of a plant [94]. The authors do not provide a concrete implementation, but some high-level suggestions for device manufacturers, system integrators, plant owners, and Industry 4.0 architects, *e.g.,* which parts they have to realize themselves and which parts are provided by the AAS. The authors also discuss the conceptual overlap between digital twins and AAS. They follow a definition of digital twin mainly based on the one given by NASA as a simulation of a physical system enriched with sensor data [86]. They find the overlap of digital twins and AASs in similar concepts around semantic specifications of the physical system. Thus, the authors state that a fully developed digital twin in the future can be used synonymously for an AAS. The described AAS follows a Type 1 classification without active behavior.

8 Conclusion

Both, the Asset Administration Shell and digital twins are concepts aiming to foster the digital transformation. Where the digital twin is understood wildly differently in general, the most prominent definitions and novel standards seem to suggest them being complex software systems that monitor a twinned system, reason about this, and send back commands. As such, implementations of digital twins could be AASs.

We examined the question of whether and to which extent the AAS meets the requirements of digital twins based on popular definitions, standards, and models. We found no simple answer to this question as it is much more dependent on the assumed AAS type, as illustrated in Fig. 13: There is only a slight congruence between the Type 1 AAS and digital twins, which lies in understanding the AAS as the knowledge base of the digital twin, covering virtual models and data about, but not from the actual system. Type 2 AASs extend that to resemble an infrastructure for digital shadows over static models of the actual system. Type 2 AASs, cover not only virtual models, but also data from and about the actual system. Literature and reference implementations on the Type 3 AAS indicate that a Type 3 AAS, which shall be able to send commands back to a connected system, could indeed be nearly complete implementation technology for digital twins in the sense of the requirements identified in this paper. Type 3 AASs extend the Type 2 AASs to include services about the actual system. However, with the specification of Type 3 AAS still ongoing, future work has to revisit this assumption.

Acknowledgements Partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—Model-Based DevOps—505496753. Website: https://mbdo.github.io/ Partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–EXC 2023 Internet of Production–390621612. Website: https://www.iop.rwth-aachen.de Partly funded by the joint project SDM4FZI, supported by the Bundesministerium für Wirtschaft und Klimaschutz (BMWK, Federal Ministry for Economic Affairs and Climate Action) as part of the "Future Investments in the Automotive Industry" funding program. Website: https://www.sdm4fzi.de/.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Abdel-Aty, T.A., Negri, E., Galparoli, S.: Asset administration shell in manufacturing: applications and relationship with digital twin. IFAC-PapersOnLine **55**(10), 2533–2538 (2022)
2. Alfonso, I., Conrardy, A., Sulejmani, A., Nirumand, A., Ul Haq, F., Gomez-Vazquez, M., Sottet, J.S., Cabot, J.: Building besser: an open-source low-code platform. In: Enterprise, Business-Process and Information Systems Modeling, pp. 203–212. Springer Nature Switzerland (2024)
3. Andryushkevich, S.K., Kovalyov, S.P., Nefedov, E.: Composition and application of power system digital twins based on ontological modeling. In: 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), vol. 1, pp. 1536–1542. IEEE (2019)
4. Apel, S., Kastner, C., Lengauer, C.: Featurehouse: language-independent, automated software composition. In: 2009 IEEE 31st International Conference on Software Engineering, pp. 221–231. IEEE (2009)
5. Appeltauer, M., Hirschfeld, R., Masuhara, H., Haupt, M., Kawauchi, K.: Event-specific software composition in context-oriented programming. In: Software Composition: 9th International Conference, SC 2010, Malaga, Spain, July 1-2, 2010. Proceedings 9, pp. 50–65. Springer (2010)
6. Ardanza, A., Moreno, A., Segura, Á., de la Cruz, M., Aguinaga, D.: Sustainable and flexible industrial human machine interfaces to support adaptable applications in the Industry 4.0 paradigm. Int. J. Prod. Res. **57**, 4045–4059 (2019)
7. Arm, J., Benesl, T., Marcon, P., Bradac, Z., Schröder, T., Belyaev, A., Werner, T., Braun, V., Kamensky, P., Zezulka, F., et al.: Automated design and integration of asset administration shells in components of industry 4.0. Sensors **21**(6), 2004 (2021)
8. Bader, S., Barnstedt, E., Bedenbender, H., Berres, B., Billmann, M., Ristin, M.: Details of the asset administration shell-part 1: the exchange of information between partners in the value chain of industrie 4.0 (2022)
9. Bader, S., Berres, B., Boss, B., Gatterburg, A., Hoffmeister, M., Kogan, Y., Köpke, A., Lieske, M., Miny, T., Neidig, J., Orzelski, A., Pollmeier, S., Sauer, M., Schel, D., Schröder, T., Thron, M., Usländer, T., Vialkowitsch, J., Vollmar, F., Ziesche, C.: Details of the asset administration shell. part 2 -interoperability at runtime-exchanging information via application programming interfaces (version 1.0rc01) (2020)
10. Bader, S.R., Maleshkova, M.: The semantic asset administration shell. In: Semantic Systems. The Power of AI and Knowledge Graphs: 15th International Conference, SEMANTiCS 2019, pp. 159–174. Springer (2019)
11. Bano, D., Michael, J., Rumpe, B., Varga, S., Weske, M.: Process-aware digital twin cockpit synthesis from event logs. J. Comput. Lang. **70**, 101121 (2022)
12. Bencomo, N., Götz, S., Song, H.: Models@run.time: a guided tour of the state of the art and research challenges. Softw. Syst. Model. **18**(5), 3049–3082 (2019)
13. Biesinger, F., Meike, D., Kraß, B., Weyrich, M.: A case study for a digital twin of body-in-white production systems general concept for automated updating of planning projects in the digital factory. In: 23rd International Conference on Emerging Technologies and Factory Automation (ETFA) (2018)
14. Bolender, T., Bürvenich, G., Dalibor, M., Rumpe, B., Wortmann, A.: Self-adaptive manufacturing with digital twins. In: 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). SEAMS '21, pp. 1–4. Association for Computing Machinery, New York, NY, USA (2021)
15. Bonetti, F., Bucchiarone, A., Michael, J., Cicchetti, A., Marconi, A., Rumpe, B.: Digital twins of socio-technical ecosystems to drive societal change. In: International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). ACM/IEEE (2024)
16. Bordeleau, F., Combemale, B., Eramo, R., van den Brand, M., Wimmer, M.: Towards model-driven digital twin engineering: current opportunities and future challenges. In: Systems Modelling and Management, pp. 43–54. Springer International Publishing, Cham (2020)
17. Bordeleau, F., Combemale, B., Eramo, R., van den Brand, M., Wimmer, M.: Towards model-driven digital twin engineering: Current opportunities and future challenges. In: Babur, Ö., Denil, J., Vogel-Heuser, B. (eds.) Systems Modelling and Management, pp. 43–54. Springer International Publishing, Cham (2020)
18. Brambilla, M., Cabot, J., Wimmer, M.: Model-driven software engineering in practice: second edition. Synth. Lectures Softw. Eng. **3**(1), 1–207 (2017)
19. Brecher, C., Dalibor, M., Rumpe, B., Schilling, K., Wortmann, A.: An ecosystem for digital shadows in manufacturing. Proc. CIRP **104**, 833–838 (2021)
20. Burgueño, L., Ciccozzi, F., Famelis, M., Kappel, G., Lambers, L., Mosser, S., Paige, R.F., Pierantonio, A., Rensink, A., Salay, R., Taentzer, G., Vallecillo, A., Wimmer, M.: Contents for a model-based software engineering body of knowledge. Softw. Syst. Model. **18**(6), 3193–3205 (2019)
21. Caesar, B., Jansen, N., Weigand, M., Ramonat, M., Gundlach, C.S., Fay, A., Rumpe, B.: Extracting functional machine knowledge from STEP files for digital twins. In: 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE (2022)
22. Cavalieri, S., Salafia, M.G.: Asset administration shell for PLC representation based on IEC 61131–3. IEEE Access **8**, 142,606-142,621 (2020)
23. Chen, X., Kang, E., Shiraishi, S., Preciado, V.M., Jiang, Z.: Digital behavioral twins for safe connected cars. In: Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (2018)
24. Combemale, B., Jansen, N., Jézéquel, J.M., Michael, J., Perez, Q., Rademacher, F., Rumpe, B., Vojtisek, D., Wortmann, A., Zhang, J.: Model-based DevOps: foundations and challenges. In: Di Ruscio, D., Lambers, L. (eds.) International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 429–433. IEEE, ACM/IEEE (2023)
25. Dalibor, M., Heithoff, M., Michael, J., Netz, L., Pfeiffer, J., Rumpe, B., Varga, S., Wortmann, A.: Generating customized low-code development platforms for digital twins. J. Comput. Lang. **70**, 101117 (2022)
26. Dalibor, M., Jansen, N., Rumpe, B., Schmalzing, D., Wachtmeister, L., Wimmer, M., Wortmann, A.: A cross-domain systematic mapping study on software engineering for Digital Twins. J. Syst. Softw. **193**, 111361 (2022)
27. David, I., Bork, D.: Towards a taxonomy of digital twin evolution for technical sustainability. In: ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion. IEEE (2023)
28. De Giacomo, G., Favorito, M., Leotta, F., Mecella, M., Silo, L.: Digital twin composition in smart manufacturing via Markov decision processes. Comput. Ind. **149**, 103,916 (2023)
29. Deantoni, J., Muñoz, P., Gomes, C., Verbrugge, C., Mittal, R., Heinrich, R., Bellis, S., Vallecillo, A.: Quantifying and combining uncertainty for improving the behavior of digital twin systems (2024)

30. Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., Wimmer, M.: Low-code development and model-driven engineering: two sides of the same coin? Softw. Syst. Model. **21**(2), 437–446 (2022)
31. Digital Twin consortium: capabilities periodic table (2022). https://www.digitaltwinconsortium.org/initiatives/capabilities-periodic-table/. Last accessed: 2024-03-12
32. Digital Twin consortium: platform stack architectural framework: an introductory guide (2023). https://www.digitaltwinconsortium.org/wp-content/uploads/sites/3/2023/07/Platform-Stack-Architectural-Framework-2023-07-11.pdf. Last accessed: 2024-03-01
33. Dong, R., She, C., Hardjawana, W., Li, Y., Vucetic, B.: Deep learning for hybrid 5G services in mobile edge computing systems: learn from a Digital Twin. IEEE Trans. Wireless Commun. **18**, 4692–4707 (2019)
34. Drath, R., Mosch, C., Hoppe, S., Faath, A., Barnstedt, E., Fiebiger, B., Schlögl, W.: Diskussionspapier–Interoperabilität mit der Verwaltungsschale, OPC UA und AutomationML Zielbild und Handlungsempfehlungen für industrielle Interoperabilität. Dokumentversion 5.3 (2023). https://opcfoundation.org/wp-content/uploads/2023/04/Diskussionspapier-Zielbild-und-Handlungsempfehlungen-fur-industrielle-Interoperabilitat-5.3-protected.pdf. Last accessed: 2024-03-12
35. Ellwein, C., Neumann, R., Verl, A.: Software-defined manufacturing: data representation. Proc. CIRP **118**, 360–365 (2023)
36. Eramo, R., Bordeleau, F., Combemale, B., van Den Brand, M., Wimmer, M., Wortmann, A.: Conceptualizing digital twins. IEEE Software (2021)
37. Evans, B., Braun, S., Ulmer, J., Wollert, J.: AAS implementations–current problems and solutions. In: 2022 20th International Conference on Mechatronics - Mechatronika (ME) (2022)
38. Ferko, E., Bucaioni, A., Behnam, M.: Architecting digital twins. IEEE Access **10**, 50335–50350 (2022)
39. Frick, F., Ellwein, C., Lechler, A., Neubauer, M., Verl, A.: Software-defined manufacturing: Reference architecture. In: 2024 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), pp. 1289–1295 (2024)
40. Frysak, J., Kaar, C., Stary, C.: Benefits and pitfalls applying RAMI4.0. In: 2018 IEEE Industrial Cyber-Physical Systems (ICPS), pp. 32–37 (2018)
41. Fur, S., Ajdinović, S., Lechler, A., Verl, A.: Towards an implementation of simulation based digital twins in cyber-physical production systems environments. In: 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–4 (2023)
42. Fur, S., Heithoff, M., Michael, J., Netz, L., Pfeiffer, J., Rumpe, B., Wortmann, A.: Sustainable digital twin engineering for the internet of production. In: Digital Twin Driven Intelligent Systems and Emerging Metaverse, pp. 101–121. Springer (2023).
43. Grüner, S., Hoernicke, M., Stark, K., Schoch, N., Eskandani, N., Pretlove, J.: Towards asset administration shell-based continuous engineering in process industries. At-Automatisierungstechnik **71**(8), 689–708 (2023)
44. Hankel, M., Rexroth, B.: The reference architectural model industrie 4.0 (rami 4.0). Zvei **2**(2), 4–9 (2015)
45. IEC Central Secretary: Common data dictionary. Standard IEC 61360-4, International Electrotechnical Commission (IEC), Geneva, CH (2005). https://cdd.iec.ch/cdd/iec61360/iec61360.nsf/TreeFrameset
46. Industrial Digital Twin Association: Details of the asset administration shell - part 2 (2021). https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part2_V1.html. Last accessed: 2024-08-09
47. ISO/DIS: ISO/DIS 23247 automation systems and integration-digital twin framework for manufacturing. Tech. rep., International Standardization Organization (ISO) (2020)
48. Jacoby, M., Volz, F., Weißenbacher, C., Müller, J.: FA 3 ST Service–An open source implementation of the reactive asset administration shell. In: 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–8. IEEE (2022)
49. Joordens, M., Jamshidi, M.: On the development of robot fish swarms in virtual reality with digital twins. In: 13th Annual Conference on System of Systems Engineering (SoSE) (2018)
50. Kannoth, S., Hermann, J., Damm, M., Rübel, P., Rusin, D., Jacobi, M., Mittelsdorf, B., Kuhn, T., Antonino, P.O.: Enabling SMeS to industry 4.0 using the BaSyx middleware: a case study. In: Software Architecture: 15th European Conference, ECSA 2021, Virtual Event, Sweden, September 13-17, 2021, Proceedings, pp. 277–294. Springer (2021)
51. Klingel, L., Heine, A., Acher, S., Dausend, N., Verl, A.: Simulation-based predictive real-time collision avoidance for automated production systems. In: 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), pp. 1–6 (2023)
52. Knapp, G., Mukherjee, T., Zuback, J., Wei, H., Palmer, T., De, A., DebRoy, T.: Building blocks for a digital twin of additive manufacturing. Acta Mater. **135**, 390–399 (2017)
53. Kritzinger, W., Karner, M., Traar, G., Henjes, J., Sihn, W.: Digital Twin in manufacturing: a categorical literature review and classification. Ifac-PapersOnline **51**(11), 1016–1022 (2018)
54. Lauzeral, N., Borzacchiello, D., Kugler, M., George, D., Rémond, Y., Hostettler, A., Chinesta, F.: A model order reduction approach to create patient-specific mechanical models of human liver in computational medicine applications. Comput. Methods Programs Biomed. **170**, 95–106 (2019)
55. Lehner, D., Pfeiffer, J., Tinsel, E.F., Strljic, M.M., Sint, S., Vierhauser, M., Wortmann, A., Wimmer, M.: Digital twin platforms: requirements, capabilities, and future prospects. IEEE Softw. **39**(2), 53–61 (2022)
56. Lou, P., Liu, S., Hu, J., Li, R., Xiao, Z., Yan, J.: Intelligent machine tool based on edge-cloud collaboration. IEEE Access **8**, 139,953-139,965 (2020)
57. Lutters, E.: Pilot production environments driven by digital twins. S. Afr. J. Ind. Eng. **29**, 40–53 (2018)
58. Majumder, M., Wiesmayr, B., Zoitl, A.: Extending the OPC UA companion specification for an IEC 61499-based control system. In: 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–4 (2023)
59. Mandolla, C., Petruzzelli, A.M., Percoco, G., Urbinati, A.: Building a digital twin for additive manufacturing through the exploitation of blockchain: a case analysis of the aircraft industry. Comput. Ind. **109**, 134–152 (2019)
60. Mediavilla, M.A., Lagnese, M., Pomp, A., Meisen, T.: Asset administration shell-based engineering change management process: challenges and ways forward. Proc. CIRP **120**, 1010–1015 (2023)
61. Michael, J., Blankenbach, J., Derksen, J., Finklenburg, B., Fuentes, R., Gries, T., Hendiani, S., Herlé, S., Hesseler, S., Kimm, M., Kirchhof, J.C., Rumpe, B., Schüttrumpf, H., Walther, G.: Integrating models of civil structures in digital twins: state-of-the-Art and challenges. J. Infrastruct. Intell. Resil. **3**(3), 100100 (2024)
62. Michael, J., Bork, D., Wimmer, M., Mayr, H.: Quo Vadis Modeling? findings of a community survey, an Ad-hoc bibliometric analysis, and expert interviews on data, process, and software modeling. J. Softw. Syst. Model. (SoSyM) **23**(1), 7–28 (2024)
63. Michael, J., David, I., Bork, D.: Digital Twin evolution for sustainable smart ecosystems. In: International Conference on

Model Driven Engineering Languages and Systems Companion (MODELS-C). ACM/IEEE (2024)
64. Michael, J., Nachmann, I., Netz, L., Rumpe, B., Stüber, S.: Generating Digital Twin cockpits for parameter management in the engineering of wind turbines. In: Modellierung 2022, LNI, pp. 33–48. GI (2022)
65. Michael, J., Pfeiffer, J., Rumpe, B., Wortmann, A.: Integration challenges for digital twin systems-of-systems. In: 2022 IEEE/ACM 10th International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems (SESoS), pp. 9–12. IEEE (2022).
66. Michael, J., Pfeiffer, J., Rumpe, B., Wortmann, A.: Integration challenges for Digital Twin systems-of-systems. In: 10th IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems, pp. 9–12. ACM (2022)
67. Michael, J., Schwammberger, M., Wortmann, A.: Explaining cyberphysical system behavior with Digital Twins. IEEE Softw. **41**(1), 55–63 (2024)
68. Muñoz, P., Wimmer, M., Troya, J., Vallecillo, A.: Using trace alignments for measuring the similarity between a physical and its digital twin. In: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS '22, pp. 503–510. Association for Computing Machinery, New York, NY, USA (2022)
69. Muñoz, P., Troya, J., Vallecillo, A.: Using UML and OCL models to realize high-level digital twins. In: 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 212–220. IEEE (2021)
70. Muñoz Ariza, P., Troya-Castilla, J., Vallecillo-Moreno, A.J., et al.: A conceptual architecture for building digital twins. In: STAF Workshops (2023)
71. Neubauer, M., Steinle, L., Reiff, C., Ajdinovic, S., Klingel, L., Lechler, A., Verl, A.: Architecture for manufacturing-x: bringing asset administration shell, eclipse dataspace connector and OPC UA together. Manuf. Letter. **37**, 1–6 (2023)
72. für Normung, D.D.I.: Reference architecture model industrie 4.0 (rami4. 0) (2016)
73. Oakes, B.J., Parsai, A., Meyers, B., David, I., Mierlo, S.V., Demeyer, S., Denil, J., Meulenaere, P.D., Vangheluwe, H.: A digital twin description framework and its mapping to asset administration shell. In: International Conference on Model-Driven Engineering and Software Development, pp. 1–24. Springer (2021)
74. Pargmann, H., Euhausen, D., Faber, R.: Intelligent big data processing for wind farm monitoring and analysis based on cloud-technologies and digital twins: a quantitative approach. In: 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA) (2018)
75. Park, K., Yang, J., Noh, S.D.: Vredi: Virtual representation for a digital twin application in a work-center-level asset administration shell. Journal of Intelligent Manufacturing (2021)
76. Park, K.T., Nam, Y.W., Lee, H.S., Im, S.J., Noh, S.D., Son, J.Y., Kim, H.: Design and implementation of a digital twin application for a connected micro smart factory. Int. J. Comput. Integr. Manuf. **32**, 596–614 (2019)
77. Pérez-Porras, D., Muñoz, P., Troya, J., Vallecillo, A.: Key-value vs graph-based data lakes for realizing digital twin systems (poster). In: STAF Workshops (2022)
78. Pfeiffer, J., Lehner, D., Wortmann, A., Wimmer, M.: Modeling capabilities of Digital Twin platforms–Old wine in new bottles? J. Object Technol. **21**(3), 1–14 (2022)
79. Plattform Industrie 4.0: Details of the Asset Administration Shell (2019). https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V2.html. Last accessed: 2024-02-28
80. Plattform Industrie 4.0: Asset administration shell reading guide (2022). https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/AAS-ReadingGuide_202201.pdf?__blob=publicationFile&v=1. Last accessed: 2024-03-12
81. Rahal, J.R., Schwarz, A., Sahelices, B., Weis, R., Antón, S.D.: The asset administration shell as enabler for predictive maintenance: a review. J. Intell. Manuf. **8**, 1–15 (2023)
82. Risling, M., Himmelstoss, H., Brandstetter, A., Shi, D., Bauernhansl, T.: Bridging the gap: a framework for structuring the asset administration shell in digital twin implementation for industry 4.0. ESSN: 2701-6277 pp. 760–770 (2023)
83. Schroeder, G.N., Steinmetz, C., Pereira, C.E., Espindola, D.B.: Digital twin data modeling with automationml and a communication methodology for data exchange. IFAC-PapersOnLine **49**(30), 12–17 (2016)
84. Schäfer, S., Schöttke, D., Kämpfe, T., Denkov, V., Zielstorff, A.: Component test – test strategies with asset administration shells. In: 2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE), pp. 1–7 (2023)
85. SEBoK Editorial Board: Guide to the systems engineering body of knowledge (sebok) (2024). https://sebokwiki.org/w/images/sebokwiki-farm!w/d/db/Guide_to_the_Systems_Engineering_Body_of_Knowledge_v2.10.pdf
86. Shafto, M., Conroy, M., Doyle, R., Glaessgen, E., Kemp, C., LeMoigne, J., Wang, L.: Draft modeling, simulation, information technology & processing roadmap. Technol. Area **11**, 1–32 (2010)
87. Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., Sui, F.: Digital twin-driven product design, manufacturing and service with big data. Int. J. Adv. Manuf. Technol. **94**, 3563–3576 (2018)
88. Tao, F., Liu, W., Zhang, M., Hu, Tl., Qi, Q., Zhang, H., Sui, F., Wang, T., Xu, H., Huang, Z., et al.: Five-dimension digital twin model and its ten applications. Comput. Integr. Manuf. Syst. **25**(1), 1–18 (2019)
89. The Institute of Electrical and Electronics Engineers: IEEE Standard Glossary of Software Engineering Terminology. IEEE Std **610**(12–1990), pp. 1–84 (1990)
90. Ullah, A.S.: Modeling and simulation of complex manufacturing phenomena using sensor signals from the perspective of Industry 4.0. Adv. Eng. Inform. **39**, 1–13 (2019)
91. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA): VDI-Statusreport Industrie 4.0 - Gegenstände, Entitäten, Komponenten (2014)
92. Verein Deutscher Ingeneure, Verband der Elektotechnik Elektronik Informationstechnik: Sprache für I4.0-Komponenten (2020)
93. Verner, I., Cuperman, D., Fang, A., Reitman, M., Romm, T., Balikin, G.: Robot online learning through Digital Twin experiments: a weightlifting project. In: Online Engineering & Internet of Things: Proceedings of the 14th International Conference on Remote Engineering and Virtual Instrumentation REV 2017, held 15-17 March 2017, Columbia University, New York, USA, pp. 307–314. Springer (2018)
94. Wagner, C., Grothoff, J., Epple, U., Drath, R., Malakuti, S., Grüner, S., Hoffmeister, M., Zimermann, P.: The role of the industry 4.0 asset administration shell and the digital twin during the life cycle of a plant. In: 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–8 (2017)
95. Walker, M., Klingel, L., Oechsle, S., Neubauer, M., Lechler, A., Verl, A.: Safeguarded continuous deployment of control containers through real-time simulation. In: 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–8 (2023)
96. Wei, K., Sun, J., Liu, R.: A review of asset administration shell. In: 2019 IEEE International Conference on Industrial Engineer-

ing and Engineering Management (IEEM), pp. 1460–1465. IEEE (2019)
97. Wenger, M., Zoitl, A., Müller, T.: Connecting PLCs with their asset administration shell for automatic device configuration. In: 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), pp. 74–79. IEEE (2018)
98. Wiesmayr, B., Zoitl, A., Prenzel, L., Steinhorst, S.: Supporting a model-driven development process for distributed control software. In: 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–8 (2022)
99. Yallic, F., Albayrak, Ö., Ünal, P.: Asset administration shell generation and usage for digital twins: a case study for non-destructive testing. In: IN4PL, pp. 299–306 (2022)
100. Ye, X., Xu, W., Liu, J., Zhong, Y., Liu, Q., Zhou, Z., Song, W.S., Hong, S.H.: Implementing digital twin and asset administration shell models for a simulated sorting production system. IFAC-PapersOnLine **56**(2), 11880–11887 (2023)
101. Yusupbekov, N., Abdurasulov, F., Adilov, F., Ivanyan, A.: Application of cloud technologies for optimization of complex processes of industrial enterprises. In: International Conference on Theory and Applications of Fuzzy Systems and Soft Computing (2018)

Jingxi Zhang is a Research assistant currently prusuing Ph.D. at Institut for Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart in the field of digital twins. He has received B.Sc. and M.Sc. in Software Engineering at the University of Stuttgart in 2021 and 2023 respectively. His research interests are systems engineering, digital twins and artificial intelligence.

Carsten Ellwein born 1989, studied Software Engineering and Management at Heilbronn University of Applied Sciences. He joined the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) at the University of Stuttgart in 2016 and became the head of the Software and Engineering Methods group in 2018.

Malte Heithoff completed his Master's degree in Computer Science at RWTH Aachen University and has been pursuing his Ph.D. at the Chair of Software Engineering under the supervision of Prof. Dr. Bernhard Rumpe since 2021. His research focuses on the model-driven engineering of information systems and digital twins, particularly in the production domain. For more information, please visit https://www.se-rwth.de/staff/Malte.Heithoff/.

Judith Michael is PostDoc and team leader at the Software Engineering Chair of RWTH Aachen University, Germany, and the spea ker of the modeling community (QFAM) within the German Informatics Society (GI). Her research focuses on model-driven software engineering, the engineering of digital twins, and software language engineering. Her Ph.D. thesis at Alpen-Adria-Universität Klagenfurt was about cognitive modeling for assistive systems and her habilitation at RWTH Aachen University was about model-driven engineering of digital twins with informative and assistive services. For more information, please visit http://judith-michael.at

Andreas Wortmann is a full professor at the Institute for Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart. His main research interests are mod el-driven software and systems engineering, software language engineering, and digital twins. On these topics, he co-authored over 150 peer-reviewed international publications. Moreover, he serves on the editorial boards of SoSyM and JoT in the program committees of various international conferences. Find out more at http://www.wortmann.ac.