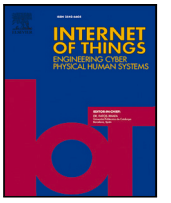
Internet of Things 34 (2025) 101739

Contents lists available at ScienceDirect

Internet of Things

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



Research article

Leveraging ontologies and Asset Administration Shells for decision-support: A case study on production planning within the injection molding domain

Patrick Sapel *, Anna Garoufali, Christian Hopmann

Institute for Plastics Processing (IKV) at RWTH Aachen University, Seffenter Weg 201, Aachen 52074, Germany

ARTICLE INFO

Dataset link: https://github.com/psapel/AAS-neo4j-Database

Keywords:
Asset Administration Shell
Graph database
Ontology
Industry 4.0
Decision-support
Injection molding

ABSTRACT

A fundamental aspect of Industry 4.0 is interoperable asset-to-asset communication, essential for creating cross-company "lab of labs". Such collaboration enables seamless data exchange across companies, streamlining manual processes like evaluating the capability of assets for specific manufacturing processes. While foundational technologies for asset interoperability exist, their integration and application in industrial contexts remain limited. Our research explores the integration of ontologies, which structure domain knowledge, and Asset Administration Shells (AAS), which represent assets in a standardized manner, to facilitate industrial interoperability. We have developed an architecture using an ontology-based graph database populated with AAS data, allowing automatic linking of AAS instances to corresponding class nodes. To demonstrate practical value, we have implemented this architecture using standardized software and tools, applying it to assess technical capabilities for a customer request in injection molding. Results confirm the potential for asset-to-asset communication in industry via graph databases, with benefits in flexible and scalable data management. However, limitations include unaddressed data safety and security concerns, as well as the need for updated database entries when AAS instances change. Additionally, challenges in scaling to integrate other domain ontologies should be tackled in future research. This work lays a foundation for advancing interoperable, cross-company data-sharing ecosystems.

1. Introduction

1.1. Motivation

In 2011, Industry 4.0 emerged as the fourth industrial revolution, aiming for more efficient production processes, especially in high-wage countries. In particular, more efficient production processes manifest, for example, in accelerated production planning and control through data-based decision-support or resilient production environments that immediately identify production alternatives in the event of a disruption, or customized production that precisely addresses the demands of individual customers [1,2]. This requires an extensive platform that acts as a data provider, where the actors on the shopfloor, the supply chain, and corresponding software systems (so-called assets) offer information about, e.g., their technical capabilities, current status, or expected trends like planned downtimes for maintenance [3]. Therefore, communication and interoperability of those assets is mandatory [4]. A central concept for addressing cross-company interoperability is the worldwide lab of labs. The lab of labs is the interconnection of all

* Corresponding author.
E-mail address: patrick.sapel@rwth-aachen.de (P. Sapel).

https://doi.org/10.1016/j.iot.2025.101739
Received 1 March 2024; Received in revised form 11 November 2024; Accepted 17 August 2025
Available online 25 September 2025

stakeholders in the supply chain or network so they can use and benefit each other's data. The vision of the lab of labs allows new business models, e.g., pay for using data from other companies, and streamlines processes, such as quickly finding a suitable partner for subcontracting [5,6]. The essential building blocks for implementing a lab of labs are interoperable assets and an underlying architecture that allows this interoperability. Nowadays, single elements for creating interoperable assets and a suitable architecture are present but need to be orchestrated to be seen as a homogeneous lab of labs.

1.2. Problem statement

Enabling asset interoperability is a challenging task. Heterogeneous systems from different vendors hinder asset interoperability as different data definitions (i.e., naming conventions, units, or formats) or communication protocols are used that require an elaborate translation, e.g., in the form of a middleware [7]. Additionally, legacy production assets that are not equipped with digital interfaces must be retrofitted to become present in the digital world [8]. In cases where data is already digitally available, it is often not directly usable for other stakeholders within the supply chain, since the data is often stored in data silos or cannot be gathered from other stakeholders due to technical aspects [9]. Another important factor is data security and privacy since manipulating production assets or involuntary shutdowns result mostly in high economic consequences [10].

Two main components predestined for implementing the lab of labs are ontologies and digital twins. Ontologies play a critical role in establishing a uniform definition of assets and their interrelationships within a given context. Defined as "a specification of a conceptualisation" [11], an ontology delineates a specific domain, fostering a standardized comprehension of its constituent elements. It encompasses comprehensive class definitions, properties, relationships, potential axioms or constraints, and instantiated individuals within a distinct domain. The ideal scenario involves formulating an ontology for a domain once and then employing it across various contexts, ensuring a consistent and standardized understanding within that domain, thus promoting interoperability. In a research setting, ontologies are well-defined and readily available for describing both the broader manufacturing domain and specific manufacturing processes. However, while ontologies exist, their practical implementation within industry practices, daily production, and decision-making processes remain underdeveloped. This discrepancy arises due to several factors: a lack of skilled professionals and the absence of software systems equipped to handle ontologies at an industrial scale. Furthermore, the lack of commitment to standards to define asset metadata and restricted ontology reuse by creators contributes to the limited interoperability among individual ontologies [12–14]. These challenges hinder the seamless integration and utilization of ontologies within industrial settings, necessitating joint efforts to bridge this gap for enhanced industrial interoperability.

A key driver for the establishment of Industry 4.0 is the concept of digital twins. Operating as a representation of physical assets in the digital realm, such as virtual 3D models or digital nameplates, digital twins possess the capability to represent entire production processes or material flows [15]. In the context of Industry 4.0, the Asset Administration Shell (AAS) stands for a tangible implementation of the digital twin concept, presently undergoing standardization (IEC 63278) and garnering increasing attention from industrial sectors. The primary focus of AAS lies in standardizing the representation of assets within the virtual world, providing a comprehensive description of asset features through properties and associated metadata [16]. Despite a well-defined structure for AAS (as documented in [17–19]), domain-specific content in the form of submodels and options for scalable, easy-to-implement AAS interoperability remain infrequent [20].

Upon evaluating the merits and limitations of both ontologies and AAS, it becomes apparent that merging these concepts holds the potential to significantly enhance interoperable production, serving as pivotal components in the realization of a lab of labs. While AAS standardizes asset characterization, ontologies provide the foundational groundwork for integrating individual AAS instances. Consequently, our research aims to address the following research question:

"How can the integration of ontologies and Asset Administration Shells facilitate interoperable shopfloors, thereby paving the way for the establishment of a worldwide lab of labs?"

1.3. Contribution

To answer the research question, we developed an architecture leveraging the required building blocks for an interoperable lab of labs, namely common data dictionaries, AAS, ontologies, graph databases, and predefined queries. Moreover, we provide a framework for its implementation that is permanently accessible on GitHub.[1]

We demonstrate that ontologies can be utilized as graph databases, which directly allocate AAS instances to the corresponding class nodes. Our architecture is a first step for enabling AAS-based, cross-company interoperability when providing the graph database centrally, e.g., in a cloud, so every participant in the supply chain or network has access to the same asset data and can query across all asset instances. Due to their standardized manner, based on common data dictionaries, the integration of all AAS instances can be processed automatically and with no programming effort. Housing the asset data in the graph database allows querying all assets independent of their location. In addition, for typical use cases, e.g., the capability check of an inquiry, predefined queries can be reused to gather efficient decision-support.

[1] https://github.com/psapel/AAS-Neo4j-Database

1.4. Organization

We structure this contribution as follows: commencing with the introduction, we present related work (Section 2) and delineate our research methodology in Section 3. This is succeeded by the presentation of our architecture for combining ontologies and AAS in Section 4, including discussing potential benefits and business applications derived from our approach. In Section 5, we manifest a framework, implement a demonstrator, and evaluate its technical capabilities within a typical industrial use case scenario. Subsequent sections encompass a critical discussion contextualizing our approach within the framework of Industry 4.0, alongside recommendations and estimations for future integration (Section 6) and a summary of our research findings, concluding with an outlook (Section 7).

2. Related work

We reviewed contributions that address AAS-to-AAS communication, with a particular emphasis on the use of ontologies, AAS, and graph databases. Table 1 provides an overview of the examined literature, which is then analyzed in detail. In this table, we highlight each study's primary technology focus, which may include ontologies, AAS, graph databases, or a combination thereof. The "subject" column specifies whether the publication centers on communication among these technologies or their management, such as through an architecture or platform. We further outline the formats, potential conversions, and applied solutions, followed by the communication protocol or architecture employed, where relevant. Finally, a brief summary is presented, covering the publication's topic and test case.

Singh et al. present a methodology for building a minimum data structure for digital twins. Therefore, they have developed a generic digital twin ontology model that can be enriched with use-case-specific properties. After the ontology is built (in Protégé), they convert the file into a relational MySQL database. Although they focus only on relational databases, they conclude that "there is a need for a way to link ontology and [graph] databases for future DT data management" [21].

Besides the production domain, *Falcone et al.* propose a graph database (Neo4j) that enables a more effective representation of digital twins in the domain of cultural heritage for architectural monuments. Their publication focuses on storing and retrieving of the data, not the communication of different digital twins. [22]

Zheng and Tian developed a framework for modeling digital twins with a particular focus on mechanical products. Based on a built ontology, they mapped this ontology to a graph database that stores the relevant data. Herein, one graph database represents one digital product twin, which is not built on standards. Since the authors center the modeling process, autonomous communication of the digital product twins is not addressed. [23]

Pham et al. convert an OWL-based ontology to a Neo4j graph database to simplify and accelerate the ontology query process [24].

To enable asset communication from the device to the Enterprise Resource Planning (ERP) level, *Ye et al.* provide a tool that converts the AASX file format[2] to Excel and vice versa. Currently, a human operator processes the conversion, and the data import process into the ERP system is not specified [25].

Another contribution of *Ye et al.* enable AAS communication between software tools. Their use case comprises the data exchange of a manufacturing order from a Manufacturing Execution System (MES) to an ERP and vice versa. The foundation is AAS for the MES and ERP based on IEC 62264 and AutomationML standards. After the MES exports the relevant data from the manufacturing order to a CSV file, an AAS interface for MES transforms the content to the MES AAS. Both the MES and ERP AAS are set up as OPC UA servers, where a subscription of the relevant properties realizes the data transfer. After receiving the data via OPC UA, an AAS interface for ERP rebuilds the CSV file and imports the information into the ERP system [26].

Bouter et al. also establish the data transfer from an ERP to the AAS. Therefore, the authors extend the AASX server (provided by Platform Industry 4.0) with an SQL connector that fetches the required data from the ERP database. An event that leads to changes in the ERP data triggers the update of the values within the AAS. However, this update process is unidirectional, so AAS cannot communicate with each other [27].

OPC UA based AAS are presented by *Hu et al.* They build AAS for a vessel and a robot arm and host the AAS as an OPC UA server, which is a standard functionality of the AASX Package Explorer, a tool widely used for AAS creation and manipulation. Thus, OPC UA clients can make use of the provided data for subsequent tasks [28].

Pribiš et al. also enable the AAS communication via OPC UA. They created a single point access for the data, referred to as the aggregation server. The aggregation server consists of a single OPC UA server and scans the network regarding the OPC UA server that originates from AAS. If one could be identified, the aggregation server fetches the AAS address space and expands its own OPC UA address space, so the aggregation server consists of all namespaces of different AAS in a central point [29].

In [30], *Neubauer et al.* developed an architecture for AAS communication within Manufacturing-X (Manufacturing-X strives for a decentralized data ecosystem, which enables digitalization across the industrial supply chain). Therefore, they use the OPC UA publish/subscribe functionality and forward data from one AAS to an MQTT broker, which can be synchronized with other AAS to enable the data flow [30].

Lu et al. set up a platform for designing a product line with the help of AAS. The AAS communication is also enabled via an OPC UA client and server principle [31].

[2] AAS can be saved in the AASX format using the AASX-Package-Explorer (AASX-PE)

Table 1
Classification of related work regarding asset-to-asset communication with the help of ontologies, AAS, and graph databases.

Author	Technology focus	Subject	Format/solution	Communication	Topic	Test case
Singh et al. [21]	Ontology	C	Ontology (OWL) to MySQL DBMS	–	Ontology transformation into a relational data model for querying purposes	Condition-Based monitoring
Falcone et al. [22]	graph DB	M	Neo4j as representation of DT	–	Continuous monitoring of cultural heritage with the help of DT	Monitoring the Cathedral of San Matteo
Zheng and Tian [23]	Ontology/graph DB	C	Ontology to graph DB	–	Ontology-based modeling method for DT of mechanical products	Modeling a DT of an helicopter
Pham et al. [24]	Ontology/graph DB	C	Ontology (OWL) to Neo4j database	–	Tackling ineffectiveness of using ontologies for data storing purposes	Comparisons in query execution time between RDBMS-based (MySQL) and graph-based (Neo4j) DB
Ye et al. [25]	AAS	C	AASX (JSON) to Excel	OPC UA	Bidirectional data exchange between enterprise and control applications	Send and receive operation strategy for a motor control system to/from ERP
Ye et al. [26]	AAS	C	MES to CSV to AAS to AAS to CSV to ERP	OPC UA	AAS-based data exchange between MES and ERP	Exchange of a manufacturing order
Bouter et al. [27]	AAS	M	AAS server enhanced SQL connector to ERP	–	Methodology for identifying the set of AAS and submodels required for a specific Industry 4.0 application scenario	Make-to-Order manufacturing
Hu et al. [28]	AAS	M	AAS as OPC UA client/server	OPC UA, MQTT, REST	Interoperable DT solution for asset-heavy industries	Robot in a shipyard/Vessel power train control system
Pribiš et al. [29]	AAS	M	AAS as OPC UA client/server	OPC UA, MQTT	Aggregating OPC UA server consists of all single AAS OPC UA server address spaces	–
Neubauer et al. [30]	AAS	M	AAS Fa^{3}st server	OPC UA, MQTT	Architecture combining AAS, Eclipse Dataspace Connector, and OPC UA in an industrial production environment	Negotiation of manufacturing capacities on a marketplace
Lu et al. [31]	AAS	M	AAS as OPC UA client/server	OPC UA, REST	Platform for AAS communication within a production network	Assets of production line, i.e., robot, AGV, CNC

(*continued on next page*)

Schweizer et al. implement active AAS that include special submodels, which enable AAS-to-AAS communication. Using these submodels, the AAS investigates the communication partners and establishes a communication channel under consideration of a suitable communication protocol (e.g., MQTT) in the first step. In the second step, the AAS retrieves the relevant property values, e.g., by subscribing to the corresponding MQTT-topic [32].

Table 1 (*continued*).

Author	Technology focus	Subject	Format/solution	Communication	Topic	Test case
Schweizer et al. [32]	AAS	M	Publish/subscribe	MQTT, HTTP	Decentralized composition of orchestrated and choreographed processes for automated commissioning via AAS	Automated commissioning of a pneumatic handling system
Kim et al. [33]	AAS	C	Native data format to XML to AAS	–	Exchange of maintenance data in process plants	Exchange of equipment condition and health status data
Treuk et al. [34]	AAS	C	CSV to AAS to JSON/FastAPI/data lake	REST (HTTP)	Architecture for data analytics from a data lake, based on AAS data	Geometric layout check of car components at an assembly line
Wein et al. [35]	AAS	C	AAS Discovery Service	AMQP, HTTP, WebSocket	Basic framework to establish a network of AAS	Path planning for tree transport in the forest
Tantik and Anderl [36]	AAS	C	AAS client/server	REST (HTTP)	Integrate AAS with W3C standard (i.e., object memory model (OMM))	Remote maintenance of a robot arm
Platenius-Mohr et al. [37]	AAS	C	Publish/subscribe	REST	Mapping DT to the AAS format	Transforming proprietary information models of a powertrain to the AAS format
Huang et al. [38]	AAS/Ontology	C	OWL to (AAS)-UML and vice versa	–	Automatic instantiation of ontologies with AAS data	Capability check of assets using MaRCO ontology
Huang et al. [39]	AAS/Ontology	C	SPARQL	–	Ontology-based capability check, based on prior data	Capability check of a robotic cell
Rimaz et al. [40]	AAS/Ontology	C	AAS (JSON) to RDF	REST	RDF representation of AAS, enabling interaction with existing RDF models such as ontologies	Intelligent waste sorting process
Kosse et al. [41]	AAS/Ontology/graph DB	C	Ontology in GraphDB/AAS to RDF	REST (HTTP)	RDF representation of AAS, populated in GraphDB for querying	Dynamic scheduling of a production line

Abbreviations: C: Communication, M: Management

An exchange of maintenance data using quasi-standardized AAS were performed by *Kim et al.* They extract data from a database in the first system into an XML file, run an Excel macro that converts the XML file to an AAS-based format, which can be inserted into the AASX-PE. Afterward, the second system can use this data by extracting the data from the AAS [33].

In the context of digitalizing industrial inspection assets through AAS, *Treuk et al.* implemented reactive AAS using the FastAPI framework to establish AAS as REST endpoints for transferring measurement data. This data is subsequently transferred and stored in a data lake. Upon the occurrence of specific events, a data analysis tool retrieves the necessary data for further processing. Their setup requires specific microservices, such as the data analysis tool, to facilitate data processing and AAS communication [34].

Wein et al. realizes the interaction of active AAS by its registration with a discovery service. This discovery service contains each AAS's ID, endpoint, and relevant attributes. When a single AAS requests a specific service (e.g., scheduling), this AAS provides the necessary boundary conditions in the form of attributes to the discovery service. The discovery service then lists all suitable interaction partners by searching the registry or matching a predefined rule to establish a connection to another AAS [35].

Tantik and Anderl provide a Component Data Model (CDM) as a specification of Object Memory Models (OMM) developed by the World Wide Web Consortium. An OMM is a structured block containing a header for identification purposes and a table of contents that briefly presents the OMM's content and payload. The authors expand the payload with specific blocks for industry 4.0 components, e.g., functional or production data, so the CDM acts as an AAS. Then, the communication is based on a client–server model using the content of the built blocks. The communication within their use case is not yet based on standardized properties. Additionally, the autonomous processing of the maintenance has to be enabled in the future [36].

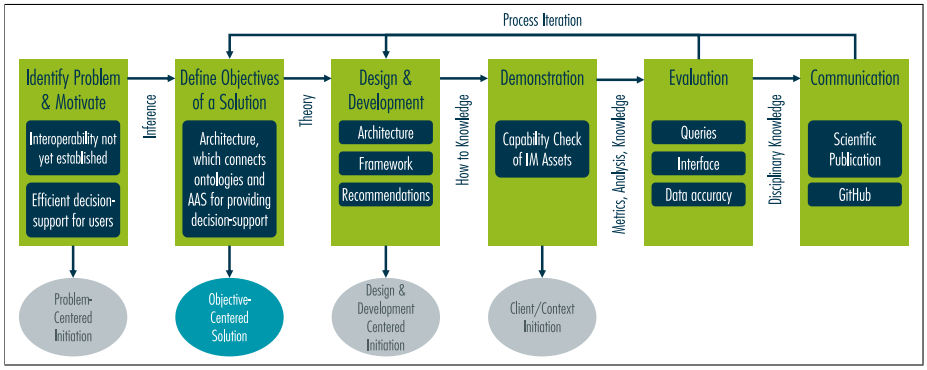


Fig. 1. DSRM process model, based on [43], enhanced with our content.

Platenius-Mohr et al. faces the transformation from proprietary digital twins to the AAS format. They realized the data access via a REST API [37].

Huang et al. focus on capability checks of a given manufacturing task by using the Manufacturing Resource Capability Ontology (MaRCO) and AAS. Therefore, they map ontology elements, i.e., classes, object and data properties, and cardinalities, to UML models representing the AAS content. Subsequently, they regenerate the AAS content back to the ontology by populating the ontology with individuals originating from the AAS classes [38]. The same authors demonstrate the mapping by means of the capability check for a production task. Therefore, the authors use SPARQL-queries to point out the resources suitable for fulfilling this production task [39].

Rimaz et al. utilize a tool that converts the content of the AAS into the Resource Description Framework (RDF), a format commonly employed by ontologies as a foundational framework for representing and sharing structured information through subject–predicate–object triples. Consequently, the transformed AAS data can be seamlessly integrated with existing RDF models and processed using additional tools from the semantic web for communication and querying purposes [40].

Kosse et al. developed a framework for AAS-based production scheduling, where each AAS represents an individual production line station. They established a general production ontology (PROD), which is instantiated in GraphDB. Furthermore, they convert the AAS instances into RDF format to also instantiate the content within GraphDB, linking the general production ontology with the AAS instances. This integration enables querying of the database to retrieve data utilized by schedulers for optimizing production schedules [41]. Both approaches necessitate transformation to RDF.

Related work regarding AAS-to-AAS communication and utilizing ontologies with AAS for enabling interoperability has shown that interconnecting AAS via a graph database originating from an ontology is not considered yet. Hence, our contribution is a first step for bridging the gap between AAS and ontologies within a communication context.

3. Methodology

In addressing our research question, we adhere to the Design Science Research Methodology (DSRM) formulated by *Hevner et al.* [42] and expanded upon by *Peffers et al.* [43]. This methodology aligns seamlessly with our objective of constructing an information system utilizing AAS acting as a decision-support system. Fig. 1 illustrates our extended DSRM process model tailored to address our research query. Subsequently, we provide details regarding each individual step.

Identify Problem & Motivate: The transition to an Industry 4.0 environment requires a fully interoperable production setup, yet the existing building blocks are not orchestrated in a way that supports seamless practical implementation. Although these elements exist, such as ontologies to define class structures within a domain and AAS for representing instance data, there is no cohesive system to leverage their combined potential for interoperability. Such a setup could offer numerous benefits, particularly through AAS-based decision-support that relieves users from manual data and information gathering while enabling real-time data sharing across entities.

Define Objectives of a Solution: To solve the identified problems, our research seeks to address this gap by exploring how the integration of ontologies and AAS can enable an interoperable “lab of labs”. Therefore, our objective is to build an architecture that effectively integrates these key building blocks, enabling interoperability and decision-support for production tasks. To accomplish this, the architecture must connect ontologies and AAS in a way that aligns with current standards, leveraging user-friendly tools that enhance accessibility for practitioners. Specifically, our solution considers:

- Building an architecture that connects and orchestrates all essential components.
- Adhering to existing standards to promote compatibility and streamline adoption across various production environments.
- Prioritizing usability by focusing on tools that are easy to use, with minimal programming requirements for operators.
- Utilizing existing interfaces and syntax in AAS rather than expanding the current syntax, ensuring that only standard AAS elements are employed to maximize compatibility and minimize complexity.

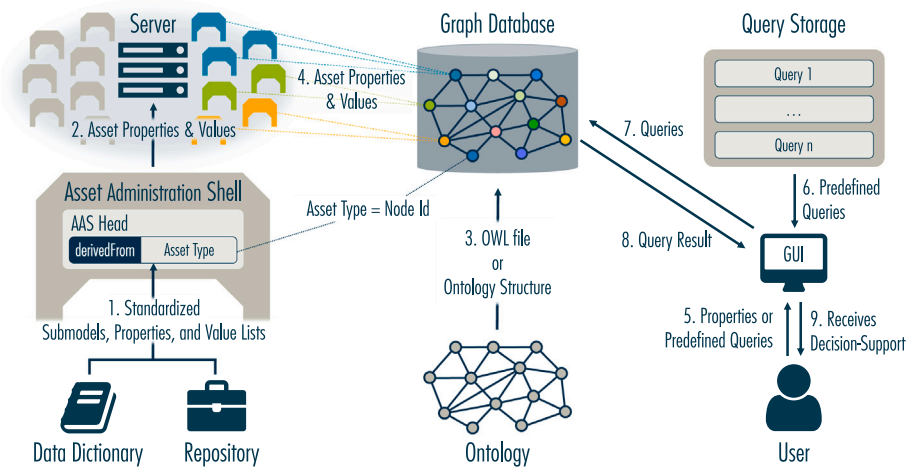


Fig. 2. Architecture leveraging ontologies and AAS to enable cross-company decision-support, based on standardized data definition.

Design & Development: In order to transfer the objectives into practice, the Design Science Research Methodology (DSRM) emphasizes the creation of artifacts, defined as "constructs, models, methods, or instantiations" [43]. In our research, the artifacts include the architecture that connects ontologies and AAS, as well as practical recommendations for implementation. Key design requirements for this architecture are that it must be independent of specific production processes and communication technologies, ensuring broad applicability across different domains. Thus, our created artifacts are:

- An architecture that seamlessly links ontologies and AAS.
- A framework using established, open-source tools for AAS handling.
- Recommendations for implementing this architecture.

This comprehensive approach facilitates a modular setup, empowering companies to integrate their systems without requiring custom-built, proprietary solutions.

Demonstration: To validate the architecture's applicability, we created a "lab of labs" demonstrator. This demonstrator allows us to test our approach through a practical use case: production planning within the Injection Molding (IM) domain. Specifically, we assess the capability of three different asset types to determine if they are able to produce a customer inquiry with given characteristics.

Evaluation: Testing insights gathered throughout the evaluation phase have driven successive refinements to the demonstrator's design. Adjustments included restructuring database queries for efficiency and refining interfaces, such as the selection of the AAS' connection property, to improve performance and ease of use. Additionally, this phase highlights key challenges in industrial adoption, such as ensuring data accuracy and maintaining standardized connections between assets and class nodes within the graph database.

Communication: Our research culminates in a set of best practices and a publicly accessible GitHub repository containing the demonstrator code. These artifacts support future implementations, allowing other researchers and practitioners to build upon our work.

4. Architecture

In this section, we present how AAS and ontologies can realize an interoperable environment using the example of a decision-support system. First, we provide a holistic overview of this setup, the used components, and their interactions from a high-level perspective. Subsequently, we will discuss each component in detail. Fig. 2 shows the entire setup.

In this architecture, we utilize AAS to represent each asset's master data (1). Typical master data have no or low update frequencies and a long lifetime, such as characteristics, identifiers, or classifications of objects [44]. The AAS provides a standardized syntax that ensures a consistent structure for representing assets, including submodels, properties, and value lists derived from data dictionaries or repositories. This standardization is essential for interoperability, as defining the asset type with a globally unique identifier guarantees a universal reference point for each asset instance.

To make all AAS data accessible, a server hosts the AAS instances, thus bringing each asset's data into the virtual realm where it is centrally visible and accessible (2). However, while ontologies are useful for defining the structure and relationships of asset data, they are not inherently suited for data storage. To address this, the ontology structure is either transferred or reconstructed within a graph database using an OWL file or by rebuilding it with a database definition language (3). In this graph database, nodes represent class categories, such as an injection molding machine, a mold, or an inquiry. Importantly, the name of each class node corresponds to the Asset Type (or the corresponding global unique identifier), forming a docking point to align AAS data with the

Preferred name	max. clamping force
IRDI	0173-1#02-AAJ507#003
Definition	maximal unit of the clamping force
Short name of unit	kN
Quantity	force
Type of Property	Non-dependent
Valency type	Multivalent
Rounding rule	
Definition class	ECLASS (0173-1#01-RAA001#001)
Property data type	Real (measure)
Allow negative values	false
Date of creation	11.02.2011
Version date	07.10.2023
Creator	System

Fig. 3. Property max. clamping force, available from ECLASS.

correct class nodes within the graph database. Data integration occurs seamlessly through an interface, such as a REST API, which scans the AAS server, automatically fetching and assigning AAS data to the relevant class nodes in the graph database (4). This automated process ensures the database is populated with asset instances linked to the correct class nodes, reducing manual setup and ensuring accurate data linkage.

For decision-support, users have the option of specifying which properties to be queried or they can use predefined queries directly (5). These predefined queries, stored in an object store, streamline the process by handling typical production use cases, such as verifying an asset's capability for fulfilling customer requests (6). When users initiate a query (7), the system processes it and returns a result (8), providing the necessary insights. Through this architecture, users receive decision-support grounded in the AAS-based database, enabling informed and data-driven decision-making for production and asset management (9). In the following subsections, the single building blocks will be presented in detail.

4.1. Common data dictionaries

In the context of Industry 4.0 and the need to achieve interoperable production, a standardized domain definition is essential. Common Data Dictionaries (CDD) serve as the foundation, administering and maintaining relevant properties from a central instance and are present in a FAIR (Findable, Accessible, Interoperable, Reusable) and standardized manner. This ensures a shared understanding of the domain, independent of vendors [45,46]. These dictionaries are publicly accessible and facilitate access and reuse. Elements within these CDD are typically officially registered, possessing internationally unique identifiers and corresponding metadata. Consequently, locating elements based on their unique identifier is straightforward, fostering interoperability through standardized shared metadata. Notable examples of such dictionaries include ECALSS[3] and the IEC CDD,[4] which is aligned with IEC 61360-4 [47]. Fig. 3 illustrates the ECLASS specification of the property `max. clamping force`.

In our established architecture, we assume that a Common Data Dictionary contains, at a minimum, a property termed `AssetType` and its corresponding specifications (e.g., `InjectionMoldingMachine` or `DrillingMachine`). For our architecture, these elements are essential as they play a central role in referencing AAS instances to the appropriate class nodes within the graph database.

4.2. Asset administration shells

The Asset Administration Shell (AAS) represents a specific implementation of a digital twin, primarily designed to standardize asset representation within the virtual realm. This framework ensures a standardized syntax and semantics for describing assets. Structurally, the AAS comprises two parts: the AAS head, responsible for identifying the AAS and the associated asset, and the AAS body, comprising all relevant asset properties along with their specific values. Moreover, the AAS enhances properties with semantics and unique identifiers. Here, the data specification according to IEC 61360-1 [48] is highly recommended in an Industry 4.0 context, e.g., by guideline VDI 2193 (Language for I4.0 components - Structure of messages) [49]. IEC 61360-1 (Standard data

[3] https://eclass.eu/en
[4] https://cdd.iec.ch/cdd/iec61360/iec61360.nsf/Welcome?OpenPage

element types with associated classification scheme) specifies a reference dictionary for metadata, clustered in semantic, identifying, administrative, and value attributes. Depending on its lifecycle stage, the AAS can represent either an asset type or an instance. The AAS type includes properties valid for all elements within a class, while AAS instances specify concrete values exclusive to a specific asset. Serving as a central repository, the AAS consolidates relevant asset data and information sourced from various systems like ERP, maintaining synchronization between the AAS and these software systems. It also allows direct storage of data and information, such as an asset's nameplate [36,50].

Because of the already specified syntax and semantics, we utilize AAS in our architecture. More specifically, we take advantage of the standardized AAS head as every AAS head contains the property `derivedFrom`, ensuring its definite presence. In our configuration, we employ the `derivedFrom` property to establish the assignment of the AAS instance to the relevant class node in the graph database. This property serves to classify the asset instance by referring it to an AAS type, ensuring its availability in the AASX-PE. Furthermore, initiatives like the Industrial Digital Twin Association (IDTA)[5] focus on developing domain-specific property definitions with enhanced semantics and fostering its standardization. Integrating new properties into a Common Data Dictionary promotes the establishment of a standardized and FAIR vocabulary, fostering a consistent characterization of all asset instances. This uniformity across AAS from diverse vendors ensures compatibility. In the context of this architecture, as the AAS consistently provides current values in the form of master data from various systems, it stands for the single source of truth. Consequently, the graph database, retrieving values from the AAS, must also reflect the latest status, and therefore is always up-to-date. Moreover, the simplicity of deploying AAS to a server and its JSON-based content storage facilitates straightforward data retrieval, for instance, via REST. In summary, AAS serves as a suitable framework for delivering vendor-independent and semantically enriched asset instances within a networked environment.

4.3. Ontologies

Ontologies serve the purpose of establishing a shared understanding and formal representation of knowledge within a specific domain. Their structured and formalized nature makes them well-suited for modeling complex environments, including the production domain. An ontology contains classes, their relationships, instances, and axioms. Axioms, constituting self-evident truths within the modeled domain, form the foundation for implicit knowledge acquisition through inference and reasoning, which is another benefit of ontologies besides a shared domain comprehension [51,52]. For instance, if "plastics material" is categorized as a "raw material", and "polypropylene" is categorized as a "plastics material", the ontology inherently infers that "polypropylene" must also qualify as a "raw material". Ontologies are typically stored in the OWL format. Similar to AAS, each ontology element is identified with an Internationalized Resource Identifier (IRI) and can be associated with annotation properties representing metadata definitions. However, unlike AAS, these annotation properties are not intrinsically based on IEC 61360-1 unless explicitly defined by the user. Additionally, ontology elements are generally not integrated or sourced from Common Data Dictionaries, reducing the need to include these elements within such dictionaries compared to AAS submodels and properties.

Despite their suitability for domain representation and knowledge inference, ontologies are not optimal for storing extensive datasets [53]. Consequently, employing pure ontologies as databases in our context and industrial use cases seems to be impractical. Instead, we leverage ontologies as a conceptual framework for structuring a graph database. In this context, graph databases are ideal due to their structural similarity to ontologies and their proven potential in navigating complex industrial environments.

4.4. Graph database

In the context of Industry 4.0, databases play a pivotal role in managing the increasing volume of data. Addressing the need to store unstructured data like images or documents, Not-only-SQL (NoSQL) databases have emerged, among which are graph databases. Graph databases store objects and instances as interconnected nodes. Their advantages include a human-intuitive database design, efficient querying of complex structures, and the absence of constraints on node property schemas. Therefore, graph databases are particularly recommended for handling numerous n:n relations [54,55]. Leveraging these advantages and the akin structure of ontologies, we employ a graph database in our architecture. We focus especially on connections between individual classes and their connection via object properties. Data and annotation properties delineating class features and metadata, as well as instances, originate from the AAS and therefore are not incorporated from the ontology. To populate the database with an asset instance, a dedicated node is created for the AAS hosted on the server, enriched with its properties and corresponding values. This node is then linked to the relevant class node. Establishing the connection between AAS instances and the correct database class node involves specifying each class node with the property `AssetType` and the corresponding asset type as its value. The asset types are derived from standardized definitions within a Common Data Dictionary. A comparison between the AAS' `derivedFrom` value and the class node's `AssetType` establishes a connection automatically if both values are identical. Using the (`isA`) relationship, where for example an InjectionMoldingMachine `isA` TechnicalAsset, instance nodes are assigned to their respective classes. While class definitions must be configured within the graph database, this requirement does not extend to class attributes. This is because the AAS includes all attributes in a standardized format, which are then transferred to the graph database along with the class definitions. This approach ensures that the semantics consistently originate from the AAS, following a standardized structure, leading to a seamless integration of new AAS instances. A benefit of this method is its ability to incorporate AAS with company-specific, non-standardized attributes into the graph database, thereby creating a flexible database without the necessity to modify the underlying database structure. This capability aligns with the utilization of different data dictionaries concomitant to DIN EN IEC 62832 (Digital Factory framework) [56]. Consequently, users can query the database using the properties and values sourced from the AAS.

[5] https://industrialdigitaltwin.org/en/

4.5. Query storage and user interface

The database and its contents serve multifaceted purposes, particularly offering users target-oriented decision-support. To facilitate user interaction, a GUI is required, allowing users to select individual queries. We provide a GUI that allows the user to perform technical capability checks for incoming inquiries. Recognizing that users in need of decision-support for production-related tasks may not be proficient in writing database queries, we have predefined typical queries and query sets (for the injection molding domain). Users can easily select relevant queries or query sets via the GUI to execute them.

Therefore, multiple queries are predefined and can be used to assemble more complex queries. We recommend storing these queries in an object database, as it is very scalable and facilitates uncomplicated retrieval of their content [57]. If the need arises for new queries, programmers can effortlessly integrate them into the database. The standardized description ensures that formulating new queries demands minimal effort as the notation of classes and properties to be queried are known. This approach results in a scalable and adaptable query storage system.

This architectural framework is adaptable for deployment within a singular shopfloor or across an entire supply chain or network. In the former scenario, the graph database, AAS, and query storage are hosted locally, whereas in the latter, these elements reside on public clouds, thereby granting different companies access to the shared database.

Envisioning the implementation of the lab of labs opens opportunities for novel business applications and services that providers and companies can offer their customers. For instance, service providers may offer ontology modeling for various manufacturing domains and enterprise layers, aligning with standards like IEC 62264 [58], with the subsequent transfer to a graph database as a service. Additionally, database expansion and the underlying software's maintenance could be managed by a service provider. Concerning AAS, the allocation of AAS for technical assets yields manifold benefits for customers. Vendors can equip their asset instances with AAS, making them accessible to customers. Leveraging instantiated AAS, customers can efficiently and paperless create corresponding master data, eliminating manual data input. This process requires that higher-level software systems embrace standardized characteristics. Employing AAS results in accelerated master data management and augmented data quality, reducing input errors and ensuring uniform data records.

In summary, we propose an architecture that employs a graph database built upon a formal ontology and leverages the content within AAS for executing queries. Our proposal introduces the advantage of utilizing a graph database capable of handling complex queries, making it well-suited for industrial applications. This architectural framework represents a novel contribution to AAS interoperability and communication and serves as a basis for implementing the lab of labs.

5. Real world example and implementation

The impulse for implementing a graph database integrated with AAS stems from the domain of production planning within the scope of injection molding. Injection molding constitutes a discontinuous manufacturing process where raw plastic granules undergo melting due to friction and heat, subsequently being injected into a mold, which is the negative counterpart of the intended part. After its solidification, the final part is ejected from the mold, either directly into a transportation container or, in the case of sensitive components like lenses, assisted by a handling device. Injection molding spans various industrial sectors, such as automotive and semiconductor industries, proving highly cost-effective in mass production. Once the process is correctly set up, both the injection molding process itself and its accompanying logistics (e.g., material supply and goods movement) can be autonomously executed [59]. This section delineates the implementation of the architecture introduced in Section 4, demonstrating how planners can derive benefits from our implementation in the production planning phase within the domain of injection molding, seamlessly facilitated by AAS.

Before starting the production, the production planning process demands the identification of a technically capable composition of assets. In this particular use case, a company (referred to as the "customer") seeks to subcontract the manufacturing of lenses for indoor lighting. Accordingly, the customer submits an inquiry including the technical specifications of the intended product. The potential manufacturer is required to determine whether the necessary technical requisites align with their machinery capabilities. Aside from specific technical prerequisites concerning the injection molding machine, the mold's functionality is restricted to oil cooling only. Additionally, to ensure the high surface quality of the lenses, their release is exclusively permitted via vacuum handling, while free-falling is not allowed. Therefore, the technical requirements originated from the inquiry have to be compared with the corresponding technical properties of the assets. For this purpose, we derived seven properties to be considered in the capability check. Table 2 illustrates the pairwise comparison and the involved properties.

The technical capability check for IMM comprises five characteristics. Besides technical characteristics, i.e., `MaxClampingForce` and `MaxPlasticizingCapacity`, the mounting dimensions and required space of the injection mold are checked. Additionally, the `RequiredHandlingType` and allowed `Coolant` for the TCU have to be checked.

Fig. 4 introduces the AAS we developed to compare the technical prerequisites outlined in the customer's inquiry with the technical capabilities of the producers' assets, i.e., four injection molding machines, two temperature control units, and two handling devices. The AAS contain the above specified properties and their values.

In addition, Fig. 4 illustrates the anticipated outcome of the database request, where capable property values are highlighted in green and incapable property values in red. It can be seen that Injection Molding Machines 1 and 2 meet the technical requirements for manufacturing the product, complemented by Temperature Control Unit 2 and Handling Device 1.

In Fig. 5, we delineate the tools and software employed in our implementation, presented in the form of a comprehensive framework.

Table 2
Comparison of requirements and capabilities between customer and manufacturer.

Property inquiry	Comp.	Asset	Property asset
MinRequiredClampingForce	≤	IMM	MaxClampingForce
ShotVolume	≤	IMM	MaxPlasticizingCapacity
GreaterMoldDimension	≤	IMM	GreaterClearDistanceBetweenColumns
SmallerMoldDimension	≤	IMM	SmallerClearDistanceBetweenColumns
RequiredOpeningStroke	≤	IMM	MaxOpeningStroke
RequiredHandlingType	=	HandDev	HandlingType
Coolant	=	TCU	Coolant

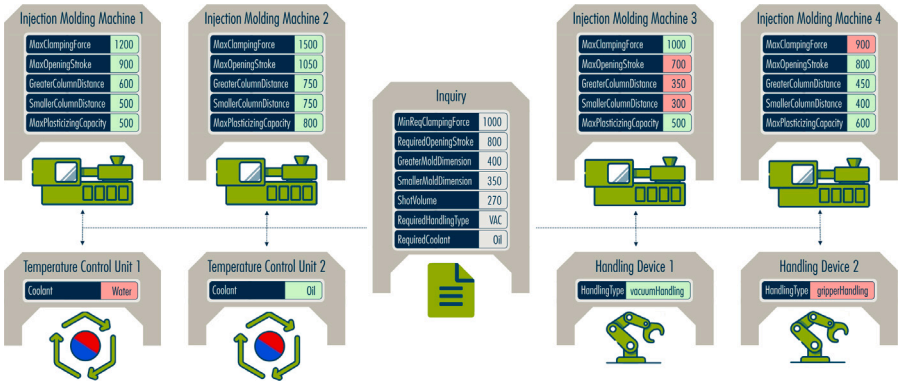


Fig. 4. AAS, their properties, and the result of the pairwise comparison with the technical requirements from the inquiry.

Fig. 5. The framework we used for our implementation.

In a previous study, we developed an extensive ontology focused on Production Planning and Control (PPC) within the Injection Molding (IM) domain [60]. Our methodology involved using Protégé (version 5.5.0) to construct an OWL file encapsulating the ontology, now accessible on GitHub.[6] Fig. 6 depicts the UML class diagram, providing a visual representation of the ontology's classes and their interconnections.

The left section shows the PPC classes and remains independent of the underlying manufacturing process. In contrast, the right section was purposefully tailored for the IM domain. Here, we introduced the class `TechnicalAsset` to link assets from any manufacturing process with the PPC domain. This ontology forms the structural foundation for our graph database.

[6] https://github.com/psapel/PPCinIM

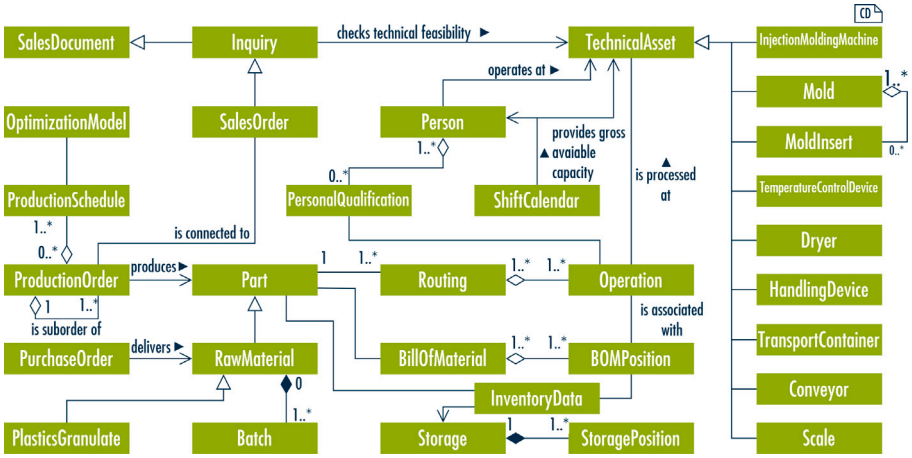


Fig. 6. UML class diagram of the PPCinIM ontology [60].

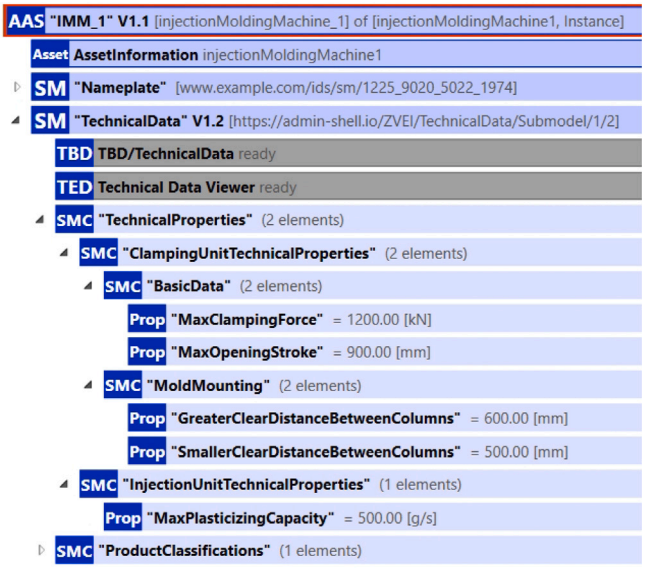


Fig. 7. The AAS of an injection molding machine within the AASX-PE.

To generate AAS files representing instances, we employed the AASX-PE V3 2023-09-12.alpha, showcased in Fig. 7, exemplary highlighting the AAS of injection molding machines.

AASX-PE, a graph-based freeware tool, facilitates intuitive AAS creation conforming to IEC 63278 standards and offers data specifications compliant with IEC 61360-1. This approach contributes significantly to achieving a standardized data definition. Moreover, AASX-PE allows the integration of registered properties, such as ECLASS, a well-established Common Data Dictionary, promoting the reuse of predefined properties. Another earlier work involved building four AAS submodel specifications specifically for the injection molding domain, describing the injection molding machine, injection mold, temperature control unit, and hot runner device. These submodels are available from IDTA and ECLASS. Consequently, we reused certain properties for our demonstrator AAS. For hosting the AAS, we utilized the AASX Server V3 2023-09-13.alpha, seamlessly compatible with AASX-PE. The structured AAS hosted on this server are accessible via a web browser, exemplified by the instance of the injection molding machine in Fig. 8 (IMM_1).

In our real-world use case demonstration, we employed Neo4j (version 5.4) as our chosen graph database. Utilizing CYPHER as the graph query language, we reconstructed the ontological classes and their interconnections within Neo4j, excluding data properties and annotations. To facilitate the population of class nodes with their instances and properties directly from the hosted AAS, we leveraged the APOC library (Awesome Procedures On Cypher), an extension for Neo4j, written primarily in Java. This library "consists of many (about 450) procedures and functions to help with many different tasks in areas like data integration, graph algorithms or data conversion" [61]. Among these functions, one enables the retrieval of JSON data from servers, enabling

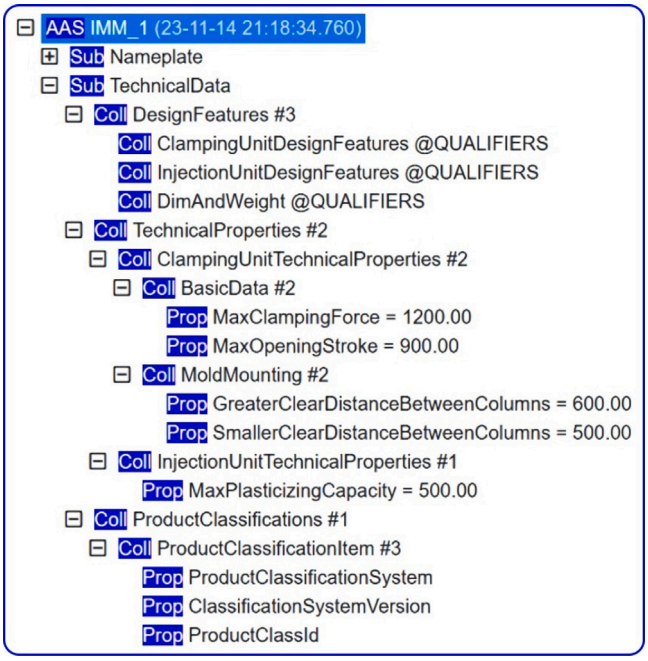


Fig. 8. The AAS of IMM_1 hosted and displayed on the AASX server.

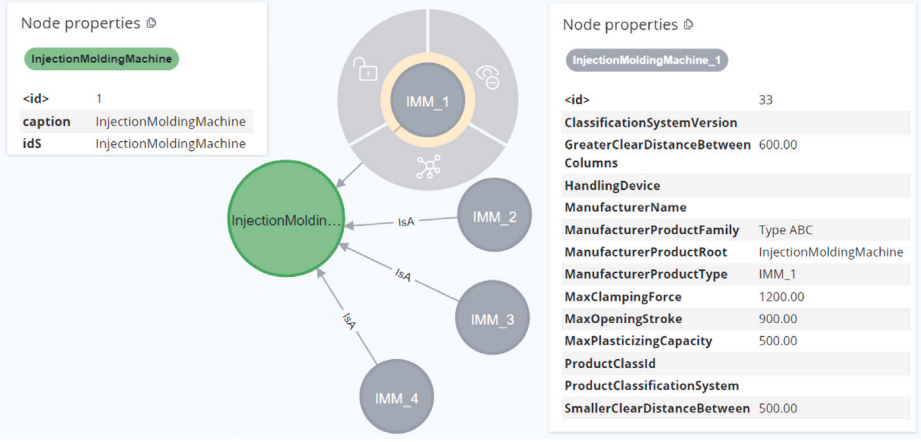


Fig. 9. The IMM class node and one exemplary instance (IMM_1) within Neo4j.

the extraction of AAS content into Neo4j, creating new instance nodes, populating them with associated properties and values, and establishing connections to the correct class nodes. The `derivedFrom` property (from AAS) and the `AssetType` property (Neo4j class node) were utilized as the bridge between these elements (for details, see Section 4). Fig. 9 illustrates the injection molding machine's class node and its properties on the left side, while the instance IMM_1, accompanied by its corresponding properties and values, is depicted on the right side.

Upon establishing and populating the Neo4j database with data from the hosted AAS, we gained the capability to execute business functions by querying the database using CYPHER. In our use case, our goal was to assess the technical capability of manufacturing a given job. To facilitate this, we developed a web-based GUI using FLASK, a Python-based web framework, enabling users to use predefined query sets. Fig. 10 shows the GUI.

The welcome page of our demonstrator provides three capability checks, one for each asset (i.e., temperature control unit (TCU), handling device, and injection molding machine). The capability assessment involves pairwise comparisons conducted via CYPHER queries (cf. Table 2). Each query is translated into a separate Python file and stored in the MinIO as an object database.

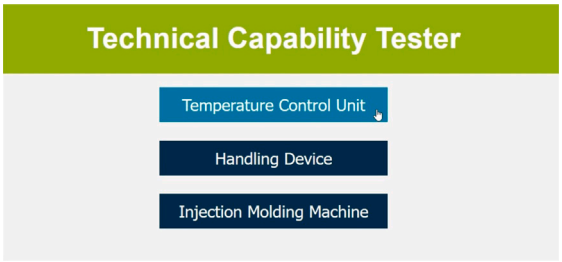


Fig. 10. The GUI, which shows the welcome page of our demonstrator.

A central application manages the access to the different queries and the Neo4j database. Therefore, we use the Neo4j Python Driver 5.15.[7] This API enables the communication between the database server and clients, hence allowing a direct connection to the Neo4j database and fetching its data. Once the user selects the necessary queries via the GUI, the application automatically connects to the Neo4j database and runs the desired queries the user requests. The query fetches the relevant inquiry properties and corresponding asset properties and executes the pairwise comparison. Finally, the result is forwarded to the user and visible on the GUI. Fig. 11 demonstrates the outcome of the capability check.

As expected in Fig. 4, IMM_1 and IMM_2 are capable to manufacture the lense, supported by HandlingDevice_1 and TemperatureControlUnit_2. The demonstrator is available on GitHub.[8]

6. Discussion

The translation of our conceptual framework into industrial practice is accompanied by several challenges.

Throughout the design and implementation phases of our architecture, we identified essential prerequisites that necessitate attention for successful industrial applications. In this section, we critically evaluate our architecture, detailing its current status and highlighting specific challenges and risks. Subsequently, we propose solution approaches, recommendations, and assessments aimed at addressing these issues.

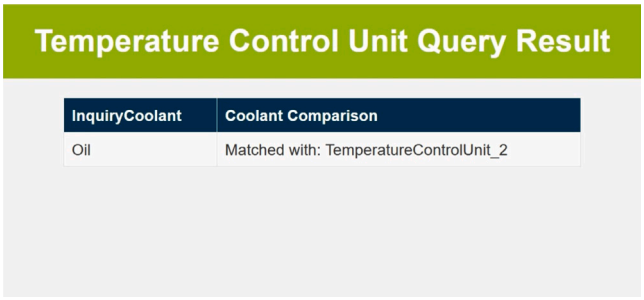
Rare AAS applications: The usage of AAS for asset representation in the virtual domain currently remains limited, which makes its wide-spread application challenging. Since AAS act as a pivotal aspect for interoperability in Industry 4.0, we envision an increasing interest in AAS, particularly through their integration as an international standard in IEC 63278 [19], initiatives such as IDTA, and the demonstration of potential applications and advantages by pioneers in the field. Due to the growing number of AAS submodels, a foundation exists for initiating the use of AAS. We recommend the implementation for small-scale AAS use cases to recognize their benefits and subsequently scale up their range of applications. However, if the adoption of AAS does not persist in industrial practices, the basis for the existence of our developed architecture would be void, given that the data input for the graph database relies on AAS.

Standardized AAS for all assets: A widespread application of AAS is only useful when adhering to standardized submodels and properties. Challenges arising from varying interpretations of individual properties among competitors hinder standardization. Initiatives like IDTA attempt to unite competitors to establish a shared standard. Our experience within the domain of plastics processing indicates a willingness among cross-company workgroups to form a unified domain understanding. Therefore, we recommend the establishment of cross-company working groups to foster a shared domain understanding, ideally transitioning into an official standard (e.g., IEC) or specifications such as IDTA, enabling other companies to build upon these outcomes. Before beginning with creating new AAS, we strongly recommend reusing predefined elements from Common Data Dictionaries. Since the connection of the AAS with the graph database, as well as the queries in our architecture, relies on standardized property designations, adherence to these designations is mandatory for the operation of the architecture.

Vendor-specific property definition: A challenge in establishing a standardized domain understanding could arise from achieving consensus on property designations. If no agreement can be found, we propose utilizing the meta property `synonymous_name`, specified in IEC 61360–1 but currently absent in AAS specifications. This property permits different names for the same property, weakening issues arising from varying nomenclatures. Here, integrated middleware acting as a translator are suitable for aligning different names with the same meaning [62,63].

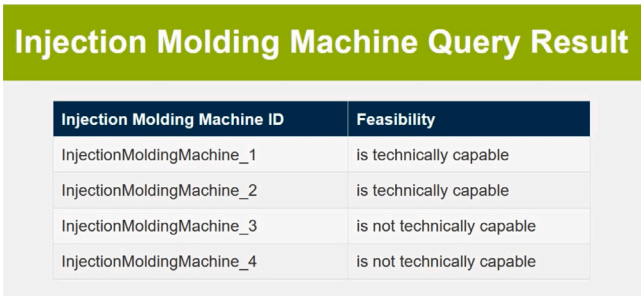
AAS as Single Source of Truth (SSOT): We posit the AAS as a single source of truth (SSOT), ensuring that the graph database consistently maintains actual values. Considering that AAS is designed to administer an asset's characteristics by comprising its master data, which indicates no or low update frequency and long-term validity, designating AAS as SSOT for this purpose is pragmatic and valid. Thus, we do not recommend using AAS and our architecture in its current form for high-frequency data. Consequently, our architecture is restricted in its capacity to manage master data, lacking any interface designed to facilitate real-time updates.

[7] https://Neo4j.com/docs/api/python-driver/current/
[8] https://github.com/psapel/AAS-Neo4j-Database



(a) Result of the TCU capability check

(b) Result of the handling device capability check



(c) Result of the injection molding machine capability check

Fig. 11. Result of the queries that serve a decision-support for the production planner by displaying capable assets.

Unique graph database structure for domains: Establishing a singular graph database structure per domain holds significant importance for a user-friendly lab of labs. Multiple disparate database structures can lead to compatibility issues, especially in cross-company collaborations. The challenge for this demand is the agreement of a durable domain structure derived from ontologies. An examination of the existing literature has uncovered several ontologies describing the manufacturing domain, partially overlapping in the definition of the domain elements [64]. We suggest developing a comprehensive domain structure based on prior work, transitioning it into an international standard, and centrally managing it, with crucial industry stakeholders supporting its adoption. As a guiding principle, the ontology we integrated into our architecture can serve as an initial foundation since it builds on preexisting standards.

Absence of default reasoning in graph databases: While ontologies excel in inference and reasoning, transferring ontology structures into graph databases results in a loss of these capabilities. Prioritizing efficient database querying over semantic reasoning aligns with our industrial-focused approach. Nevertheless, in instances where semantic reasoning is necessary, ontology files can still be utilized for such purposes, with new insights subsequently integrated into the graph database.

Definition of queries: Our approach of storing each query in a separate file leads to a potentially vast number of files in complex environments. Establishing object stores supports managing these files efficiently, with low memory requirements for individual query files. Consequently, we propose the clustering of queries based on their intended application, e.g., for technical capability checks or shop floor scheduling. Similar to the central management of the platform and database, we believe that creating, publishing, and maintaining these queries have the potential to generate new business opportunities. Subsequently, meaningful clustering of queries can be performed by those vendors.

Platform updating and maintenance: Due to a dynamically production environment, the existing platform and corresponding graph database may require updates with regard to its structure, such as integrating new assets. In addition, maintenance regarding the underlying software is mandatory, e.g., for implementing security updates. Therefore, skilled labor is necessary. While managing in-house platforms is comparatively simple, updating a cross-company platform may be challenging due to different stakeholders. In addition, the responsibilities, and thus also liability for possible errors, must be defined. We think that a cross-company collaboration platform requires central management by third-party vendors, which can emerge as a new business case in the era of Industry 4.0 as discussed in Section 4. Here, the liability and responsibility for database updating and maintenance rests with the provider, not the user.

Data safety and security: Our focus on asset interoperability omits considerations regarding data safety and security, crucial aspects in a connected world. While our contribution does not consider these aspects, prior work addresses security concerns within lab of labs setups [6] and highlight security aspects for the AAS, e.g., [65,66]. In our view, the unencrypted representation of assets' master data within internal applications raises minimal concerns, considering that this information typically aligns with data readily available on data sheets. In contrast, keeping the assets' master data private when sharing technical capabilities across the supply network may be advantageous. An important reason behind this approach is the utilization of machinery knowledge to assess its effectiveness, enabling estimations of sensitive data like machine hour rates [67].

In summary, the current implementation of our architecture represents a research state and has not yet been universally implemented in the industry due to the existing challenges. However, our demonstrator underlines the potential inherent in our architecture, assuming addressing the aforementioned issues. Thus, our demonstrator can serve as a foundation and blueprint for enabling asset-to-asset communication using AAS. We anticipate a shift in the mindset of companies toward employing semantic technologies for asset interoperability, thereby enabling accelerated decision-making and interoperable asset communication as envisioned in Industry 4.0.

7. Conclusion and outlook

The advent of Industry 4.0 holds the promise of increasing efficiency in production processes through autonomous operations and fast, data-driven decision-making, thereby optimizing production. This necessitates seamless communication among production assets for interoperability. Within the sphere of Industry 4.0 research, the establishment of asset-to-asset communication and interoperability within a lab of labs poses a complex challenge.

Our study reveals that the foundational elements required to enable asset-to-asset communication and interoperability already exist. Central to asset interoperability are ontologies, instrumental in formalizing and structuring domains, and Asset Administration Shells, which represent assets in the virtual world, enhancing their attributes with semantics. Bridging the gap between ontologies and AAS holds the potential for realizing interoperability.

However, while these concepts are extensively defined, our research identifies deficiencies in their integration and collaboration, hindering interoperability. Moreover, their practical implementation within industrial settings remains lacking.

To address these challenges, we have developed an architecture that facilitates the querying of AAS using a graph database structured by an ontology. This architecture includes defined, persistent interfaces that enable modular and scalable integration of AAS data into the graph database, allowing seamless connectivity and data updates. We have also implemented a query object store containing predefined, standardized queries that can be reused across different applications, streamlining data access for decision-making. Additionally, a user interface serves as the entry point for decision-makers, providing a neutral platform that is independent of both underlying communication technologies and specific production technologies, ensuring broad applicability and flexibility. Crucially, our approach aligns with FAIR data principles, ensuring that data is findable through unique identifiers, accessible via a Common Data Dictionary, standardized for consistent interpretation, and universally applicable and reusable across diverse contexts. This combination of structured interfaces, reusable queries, and a user-friendly access point supports an interoperable architecture that can adapt to various industrial settings, advancing efficient data sharing and decision-support capabilities.

Our framework demonstrates that modularity and scalability are attainable through standardized property definitions, necessitating adherence to asset description standards in the virtual world. The framework is built on established tools, including Neo4j for the graph database and MinIO as an object store while leveraging open-source programs for AAS creation and implementation. Implementing this framework showcases its potential for industrial applicability, rendering AAS data accessible and usable in virtual environments. Once the framework is set up, users are no longer required to manually add AAS instances to the graph database; instead, AAS integration can be performed automatically, ensuring that relevant data is readily available for queries. Furthermore, we evaluated the framework in a production planning use case, where its decision-support capabilities facilitated a technical capability check by comparing injection molding assets directly with a customer's inquiry requirements. This evaluation confirms the framework's utility in operational decision-making contexts, particularly in complex, data-driven environments.

Moving forward, our further research involves interconnecting different domains (e.g., injection molding and tooling) for seamless data exchange. This necessitates comprehensive standardization of domain-specific properties to enable AAS utilization and ensure interoperability. Additionally, in the context of cross-company scenarios, ensuring secure data streams is essential, requiring further exploration of security aspects within our architectural design. Another aspect is the consideration of automatic updates in case AAS data changes.

Abbreviations

AAS Asset Administration Shell

AASX-PE AASX-Package-Explorer

API Application Programming Interface

APOC Awesome Procedures On Cypher

CDD Common Data Dictionary

CDM Component Data Model

CSV Comma-Separated Values

DBMS Database Management System

RDBMS Relational Database Management System

DSRM Design Science Research Methodology

DT Digital Twin

ERP Enterprise Resource Planning

GUI Graphical User Interface

HandDev Handling Device

HTTP Hypertext Transfer Protocol

I4.0 Industry 4.0

IDTA Industrial Digital Twin Association

IEC International Electrotechnical Commission

IM Injection Molding

IMM Injection Molding Machine

IRI Internationalized Resource Identifier

JSON JavaScript Object Notation

MES Manufaction Execution System

MQTT Message Queuing Telemetry Transport

NoSQL Not only Structured Query Language

OMM Object Memory Model

OWL Web Ontology Language

PPC Production Planning and Control

RDF Resource Description Framework

REST Representational State Transfer

SPARQL SPARQL Protocol and RDF Query Language

SQL Structured Query Language

SSOT Single Source of Truth

TCU Temperature Control Unit

UML Unified Modeling Language

XML eXtensible Markup Language

CRediT authorship contribution statement

Patrick Sapel: Methodology, Writing – original draft, Validation, Writing – review & editing, Investigation, Conceptualization, Visualization, Supervision, Data curation. **Anna Garoufali:** Software, Validation, Data curation. **Christian Hopmann:** Resources, Writing – review & editing.

Declaration of competing interest

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

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), Germany under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

Data availability

The dataset generated during the current study is available in the GitHub repository: https://github.com/psapel/AAS-neo4j-Database.

References

[1] H. Kagermann, W. Wahlster, J. Helbig, Securing the Future of German Manufacturing Industry: Recommendations for Implementing the Strategic Initiative Industrie 4.0. Final Report of the Industrie 4.0 Working Group, Acatech— National Academy of Science and Engineering, 2013, URL: https://en.acatech.de/wp-content/uploads/sites/6/2018/03/Final_report_Industrie_4.0_accessible.pdf.

[2] C. Brecher, D. Özdemir, M. Brockmann, Introduction to integrative production technology, Prod. Eng. 11 (2) (2017) 93–95, http://dx.doi.org/10.1007/s11740-017-0730-y.

[3] A. Nayyar, A. Kumar (Eds.), A Roadmap to Industry 4. 0: Smart Production, Sharp Business and Sustainable Development, in: Advances in Science, Technology and Innovation Ser, Springer, Cham, 2020, URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5986767.

[4] B.Ç. Uslu, S.Ü.O. Fırat, A comprehensive study on internet of things based on key artificial intelligence technologies and industry 4.0, in: Research Anthology on Cross-Industry Challenges of Industry 4.0, IGI Global, Hershey, Pennsylvania (701 E. Chocolate Avenue, Hershey, Pennsylvania, 17033, USA), 2021, pp. 171–191, http://dx.doi.org/10.4018/978-1-7998-8548-1.ch010.

[5] M.R. Liebenberg, Autonomous Agents for the World Wide Lab: Artificial Intelligence in the Manufacturing Industry (Ph.D. thesis), RWTH Aachen - Faculty of Mathematics, Computer Science and Natural Sciences, Aachen, 2022.

[6] J. Pennekamp, M. Dahlmanns, L. Gleim, S. Decker, K. Wehrle, 2019 IEEE Global Conference on Internet of Things (GCIoT), IEEE, Piscataway, NJ, 2019, http://dx.doi.org/10.1109/GCIoT47977.2019, URL: https://ieeexplore.ieee.org/servlet/opac?punumber=9043328.

[7] M. Sadeghi, A. Carenini, O. Corcho, M. Rossi, R. Santoro, A. Vogelsang, Interoperability of heterogeneous systems of systems: Review of challenges, emerging requirements and options, in: J. Hong, M. Lanperne, J.W. Park, T. Cerny, H. Shahriar (Eds.), Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, ACM, New York, NY, USA, 2023, pp. 741–750, http://dx.doi.org/10.1145/3555776.3577692.

[8] T. Lins, R.A.R. Oliveira, Cyber-physical production systems retrofitting in context of industry 4.0, Comput. Ind. Eng. 139 (2020) 106193, http://dx.doi.org/10.1016/j.cie.2019.106193.

[9] S. Kamm, N. Jazdi, M. Weyrich, Knowledge Discovery in Heterogeneous and Unstructured Data of Industry 4.0 Systems: Challenges and Approaches, Procedia CIRP 104 (2021) 975–980, http://dx.doi.org/10.1016/j.procir.2021.11.164.

[10] C. Salkin, M. Oner, A. Ustundag, E. Cevikcan, A Conceptual Framework for Industry 4.0, in: A. Ustundag, E. Cevikcan (Eds.), Industry 4.0: Managing the Digital Transformation, in: Springer Series in Advanced Manufacturing, Springer International Publishing, Cham, 2018, pp. 3–23, http://dx.doi.org/10.1007/978-3-319-57870-5_1.

[11] T.R. Gruber, A translation approach to portable ontology specifications, Knowl. Acquis. 5 (1993) 199–220, URL: http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.

[12] F. Psarommatis, P.A. Dreyfus, D. Kiritsis, Chapter 9 - The role of big data analytics in the context of modeling design and operation of manufacturing systems, in: D. Mourtzis (Ed.), Design and Operation of Production Networks for Mass Personalization in the Era of Cloud Technology, Elsevier, 2022, pp. 243–275, http://dx.doi.org/10.1016/B978-0-12-823657-4.00012-9.

[13] M. Vigo, S. Bail, C. Jay, R. Stevens, Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design, Int. J. Hum.-Comput. Stud. 72 (12) (2014) 835–845, http://dx.doi.org/10.1016/j.ijhcs.2014.07.005.

[14] F. Ameri, D. Sormaz, F. Psarommatis, D. Kiritsis, Industrial ontologies for interoperability in agile and resilient manufacturing, Int. J. Prod. Res. 60 (2) (2022) 420–441, http://dx.doi.org/10.1080/00207543.2021.1987553.

[15] C. Semeraro, M. Lezoche, H. Panetto, M. Dassisti, Digital twin paradigm: A systematic literature review, Comput. Ind. 130 (2021) 103469, http://dx.doi.org/10.1016/j.compind.2021.103469.

[16] S. Beden, Q. Cao, A. Beckmann, Semantic asset administration shells in industry 4.0: A survey, in: 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems, ICPS, IEEE, 2021, pp. 31–38, http://dx.doi.org/10.1109/ICPS49255.2021.9468266.

[17] Federal Ministry for Economic Affairs and Climate Action, Details of the Asset Administration Shell: Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC02), Berlin, 2022, URL: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html.

[18] Federal Ministry for Economic Affairs and Climate Action, Details of the Asset Administration Shell: Part 2 - Interoperability at Runtime – Exchanging Information via Application Programming Interfaces (Version 1.0RC02), Berlin, 2021, URL: https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part_2_V1.html.

[19] DIN EN IEC 63278-1:2022-07, Asset Administration Shell for Industrial Applications – Part 1: Asset Administration Shell Structure, Standard, Beuth Verlag GmbH, Berlin, 2022.

[20] B. Evans, S. Braun, J. Ulmer, J. Wollert, AAS implementations - Current problems and solutions, in: 2022 20th International Conference on Mechatronics - Mechatronika, ME, IEEE, 2022, pp. 1–6, http://dx.doi.org/10.1109/ME54704.2022.9982933.

[21] S. Singh, E. Shehab, N. Higgins, K. Fowler, D. Reynolds, J.A. Erkoyuncu, P. Gadd, Data management for developing digital twin ontology model, Proc. Inst. Mech. Eng. Part B: J. Eng. Manuf. 235 (14) (2021) 2323–2337, http://dx.doi.org/10.1177/0954405420978117.

[22] M. Falcone, A. Origlia, M. Campi, S. Di Martino, From architectural survey to continuous monitoring: Graph-based data management for cultural heritage conservation with digital twins, Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci. XLIII-B4-2021 (2021) 47–53, http://dx.doi.org/10.5194/isprs-archives-XLIII-B4-2021-47-2021.

[23] M. Zheng, L. Tian, A hierarchical integrated modeling method for the digital twin of mechanical products, Machines 10 (1) (2022) 2, http://dx.doi.org/10.3390/machines10010002.

[24] P. Pham, T. Nguyen, P. Do, Computing domain ontology knowledge representation and reasoning on graph database, in: V. Bhateja, B. Le Nguyen, N.G. Nguyen, S.C. Satapathy, D.-N. Le (Eds.), Information Systems Design and Intelligent Applications, in: Advances in Intelligent Systems and Computing, vol. 672, Springer Singapore, Singapore, 2018, pp. 765–775, http://dx.doi.org/10.1007/978-981-10-7512-4_75.

[25] X. Ye, W.S. Song, S.H. Hong, Y.C. Kim, N.H. Yoo, Toward data interoperability of enterprise and control applications via the industry 4.0 asset administration shell, IEEE Access 10 (2022) 35795–35803, http://dx.doi.org/10.1109/ACCESS.2022.3163738.

[26] X. Ye, M. Yu, W.S. Song, S.H. Hong, An asset administration shell method for data exchange between manufacturing software applications, IEEE Access 9 (2021) 144171–144178, http://dx.doi.org/10.1109/ACCESS.2021.3122175.

[27] C. Bouter, M. Pourjafarian, L. Simar, R. Wilterdink, Towards a comprehensive methodology for modelling submodels in the industry 4.0 asset administration shell, in: 2021 IEEE 23rd Conference on Business Informatics, CBI, IEEE, 2021, pp. 10–19, http://dx.doi.org/10.1109/CBI52690.2021.10050.

[28] Z. Hu, A. Haghshenas, A. Hasan, S. Sørenes, A. Karlsen, S. Alaliyat, Interoperable digital twin solutions for asset-heavy industry, in: E. Karaarslan, O. Aydin, U. Cali, M. Challenger (Eds.), Digital Twin Driven Intelligent Systems and Emerging Metaverse, Springer Nature Singapore, Singapore, 2023, pp. 195–208, http://dx.doi.org/10.1007/978-981-99-0252-1_9.

[29] R. Pribis, L. Beno, P. Drahos, An industrial communication platform for industry 4.0 - case study, in: J. Cigánek (Ed.), 2020 Cybernetics & Informatics (K & I), IEEE, Piscataway, NJ, 2020, pp. 1–9, http://dx.doi.org/10.1109/KI48306.2020.9039873.

[30] M. Neubauer, L. Steinle, C. Reiff, S. Ajdinović, L. Klingel, A. Lechler, A. Verl, Architecture for manufacturing-x: Bringing asset administration shell, eclipse dataspace connector and OPC UA together, Manuf. Lett. 37 (2023) 1–6, http://dx.doi.org/10.1016/j.mfglet.2023.05.002.

[31] Q. Lu, X. Li, G. Li, M. Li, J. Zhou, J. Liu, X. Shen, F. Zhu, A general asset administration shell platform for production line design, in: 2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence, DTPI, IEEE, 2021, pp. 192–195, http://dx.doi.org/10.1109/DTPI52967.2021.9540170.

[32] H. Schweizer, N. Braunisch, R. Alt, K. Schmitz, M. Wollschlaeger, Prozesskomposition in verteilten automatisierungssystemen, At - Autom. 69 (3) (2021) 242–255, http://dx.doi.org/10.1515/auto-2020-0118.

[33] B. Kim, S. Kim, H. Teijgeler, J. Lee, J.Y. Lee, D. Lim, H.-W. Suh, D. Mun, Use of asset administration shell coupled with ISO 15926 to facilitate the exchange of equipment condition and health status data of a process plant, Processes 10 (10) (2022) 2155, http://dx.doi.org/10.3390/pr10102155.

[34] I.M. Treuk, A.O. Júnior, R.P. Lopes, G. Mota, J. Joaquim Mira, P. Leitao, Digitalization of industrial inspection assets through the asset administration shell, in: 2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems, ICPS, IEEE, 2024, pp. 1–6, http://dx.doi.org/10.1109/ICPS59941.2024.10640027.

[35] S. Wein, C. Fimmers, S. Storms, C. Brecher, M. Gebhard, M. Schluse, J. Rubmann, Embedding active asset administration shells in the Internet of Things using the smart systems service infrastructure, in: 2020 IEEE 18th International Conference on Industrial Informatics, INDIN, IEEE, 2020, pp. 23–28, http://dx.doi.org/10.1109/INDIN45582.2020.9442249.

[36] E. Tantik, R. Anderl, Integrated data model and structure for the asset administration shell in industrie 4.0, Procedia CIRP 60 (2017) 86–91, http://dx.doi.org/10.1016/j.procir.2017.01.048.

[37] M. Platenius-Mohr, S. Malakuti, S. Grüner, J. Schmitt, T. Goldschmidt, File- and API-based interoperability of digital twins by model transformation: An IIoT case study using asset administration shell, Future Gener. Comput. Syst. 113 (2020) 94–105, http://dx.doi.org/10.1016/j.future.2020.07.004.

[38] Y. Huang, S. Douib, L. Palacios Medinacelli, J. Malenfant, Enabling semantic interoperability of asset administration shells through an ontology-based modeling method, in: ACM / IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS'22), Association for Computing Machinery, Montréal, Canada, 2022, pp. 497–502, http://dx.doi.org/10.1145/3550356.3561606, URL: https://hal.science/hal-03968479.

[39] Y. Huang, S. Dhouib, L.P. Medinacelli, J. Malenfant, Semantic interoperability of digital twins: Ontology-based capability checking in AAS modeling framework, in: 2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems, ICPS, 2023, pp. 1–8, http://dx.doi.org/10.1109/ICPS58381.2023.10128003.

[40] M.H. Rimaz, C. Plociennik, M. Ruskowski, Semantic asset administration shell for circular economy, in: E. Blomqvist, R. García-Castro, D. Hernández, P. Hitzler, M. Lindecrantz, M. Poveda-Villalón (Eds.), Knowledge Graphs for Sustainability 2024, Greece, 2024, pp. 1–14.

[41] S. Kosse, V. Betker, P. Hagedorn, M. König, T. Schmidt, A semantic digital twin for the dynamic scheduling of industry 4.0-based production of precast concrete elements, Adv. Eng. Informatics 62 (2024) 102677, http://dx.doi.org/10.1016/j.aei.2024.102677.

[42] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, MIS Q. 28 (1) (2004) 75–105.

[43] K. Peffers, T. Tuunanen, M.A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, J. Manage. Inf. Syst. 24 (3) (2007) 45–77, http://dx.doi.org/10.2753/MIS0742-1222240302.

[44] R. Scheuch, Master Data Management: Strategie, Organisation, Architektur, first ed., dpunkt.verlag, Heidelberg, 2012, URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=7240466.

[45] H. van Vlijmen, A. Mons, A. Waalkens, W. Franke, A. Baak, G. Ruiter, C. Kirkpatrick, L.O.B. Da Silva Santos, B. Meerman, R. Jellema, D. Arts, M. Kersloot, S. Knijnenburg, S. Lusher, R. Verbeeck, J.-M. Neefs, The need of industry to go FAIR, Data Intell. 2 (1–2) (2020) 276–284, http://dx.doi.org/10.1162/dint_a_00050.

[46] A. Basu, D. Warzel, A. Eftekhari, J.S. Kirby, J. Freymann, J. Knable, A. Sharma, P. Jacobs, Call for data standardization: Lessons learned and recommendations in an imaging study, JCO Clin. Cancer Inf. 3 (2019) 1–11, http://dx.doi.org/10.1200/CCI.19.00056.

[47] DIN EN 61360-4:2005-11, Standard Data Element Types with Associated Classification Scheme for Electric Components - Part 4: IEC Reference Collection of Standard Data Element Types and Component Classes, Standard, Beuth Verlag GmbH, Berlin, 2005.

[48] DIN EN 61360-1:2018-07, DIN EN 61360-1:2018-07: Standard Data Element Types with Associated Classification Scheme - Part 1: Definitions - Principles and Methods, Standard, Beuth Verlag GmbH, Berlin, 2018.

[49] VDI 2193 Part 1:2020-04, Language for I4.0 Components - Structure of messages, Guideline, Beuth Verlag GmbH, Berlin, 2020.

[50] C. Wagner, J. Grothoff, U. Epple, R. Drath, S. Malakuti, S. Gruner, M. Hoffmeister, P. Zimermann, The role of the industry 4.0 asset administration shell and the digital twin during the life cycle of a plant, in: 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation, IEEE, Piscataway, NJ, 2017, pp. 1–8, http://dx.doi.org/10.1109/ETFA.2017.8247583.

[51] A. Marchetti, F. Ronzano, M. Tesconi, S. Minutoli, Formalizing Knowledge by Ontologies: OWL and KIF, Istituto di Informatica e Telematica, Pisa, 2008, p. 15, URL: https://core.ac.uk/reader/37831990.

[52] H.S. Pinto, J.P. Martins, Ontologies: How can they be built? Knowl. Inf. Syst. 6 (4) (2004) 441–464, http://dx.doi.org/10.1007/s10115-003-0138-1.

[53] M. Uschold, Ontology and database schema: What's the difference? Appl. Ontol. 10 (3–4) (2015) 243–258, http://dx.doi.org/10.3233/AO-150158.

[54] M. Kleppmann, Designing Data-Intensive Applications, first ed., Upfront Books and Safari, Boston, MA, 2021, URL: https://learning.oreilly.com/library/view/-/1449373321/?ar.
[55] I. Robinson, J. Webber, E. Eifrem, Graph Databases: New Opportunities for Connected Data, second ed., O'Reilly, Beijing and Boston and Farnham and Sebastopol and Tokyo, 2015, URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=3564533.
[56] DIN EN IEC 62832-1:2022-05, Industrial-Process Measurement, Control and Automation - Digital Factory Framework - Part 1: General Principles, Standard, Beuth Verlag GmbH, Berlin, 2012.
[57] J. Reis, Fundamentals of Data Engineering: Plan and Build Robust Data Systems, first ed., O'Reilly Media Incorporated, Sebastopol, 2022.
[58] DIN EN 62264-1:2014-07, Enterprise-Control System Integration - Part 1: Models and Terminology, Standard, Beuth Verlag GmbH, Berlin, 2014.
[59] D.V. Rosato, D.V. Rosato, M.G. Rosato (Eds.), Injection Molding Handbook, third ed., Springer US, Boston, MA and s.l., 2000, http://dx.doi.org/10.1007/978-1-4615-4597-2.
[60] P. Sapel, C. Hopmann, Towards an ontology-based dictionary for production planning and control in the domain of injection molding as a basis for standardized asset administration shells, J. Ind. Inf. Integr. 35 (2023) 100488, http://dx.doi.org/10.1016/j.jii.2023.100488.
[61] neo4j Labs, APOC Documentation 4.3 - Introduction, 2023, URL: https://neo4j.com/labs/apoc/4.3/introduction/.
[62] T.R. de Almeida, J.M.N. David, R.S.P. Maciel, Supporting semantic interoperability in a middleware infrastructure for the development of collaborative services, in: Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design, CSCWD, 2011, pp. 422–428, http://dx.doi.org/10.1109/CSCWD.2011.5960108.
[63] G.M. Ortego, A Semantic Middleware to Enhance Current Multimedia Retrieval Systems with Content-Based Functionalities (Ph.D. thesis), University of the Basque Country, Donostia-San Sebastián, 2011.
[64] P. Sapel, L. Molinas Comet, I. Dimitriadis, C. Hopmann, S. Decker, A review and classification of manufacturing ontologies, J. Intell. Manuf. (2024) http://dx.doi.org/10.1007/s10845-024-02425-z.
[65] A.M. Hosseini, T. Sauter, W. Kastner, Towards adding safety and security properties to the industry 4.0 asset administration shell, in: 2021 17th IEEE International Conference on Factory Communication Systems, WFCS, IEEE, 2021, pp. 41–44, http://dx.doi.org/10.1109/WFCS46889.2021.9483606.
[66] A.M. Hosseini, T. Sauter, W. Kastner, A safety and security reference architecture for asset administration shell design, in: 2022 IEEE 18th International Conference on Factory Communication Systems, WFCS, IEEE, 2022, pp. 1–8, http://dx.doi.org/10.1109/WFCS53837.2022.9779188.
[67] J. Pennekamp, P. Sapel, I.B. Fink, S. Wagner, S. Reuter, C. Hopmann, K. Wehrle, M. Henze, Revisiting the Privacy Needs of Real-World Applicable Company Benchmarking, RWTH Aachen University, 2020, http://dx.doi.org/10.18154/RWTH-2021-01321.