Computers in Industry 148 (2023) 103910

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.sciencedirect.com/journal/computers-in-industry

Modelling with AAS and RDF in Industry 4.0[☆]

Sjoerd Rongen[a,*], Nikoletta Nikolova[a], Mark van der Pas[b]

[a] TNO, Anna van Buerenplein 1, The Hague 2595DA, the Netherlands
[b] Semaku B.V., Torenallee 20, Eindhoven 5617BC, Noord-Brabant, the Netherlands

ARTICLE INFO

Keywords:
Industry 4.0
Modelling
Knowledge representation
RDF
AAS
Semantic interoperability

ABSTRACT

Industry 4.0 has proposed the Asset Administration Shell (AAS) model for digital twins. This model should help to solve interoperability issues, a topic that is also addressed by the Semantic Web and its Resource Description Framework (RDF). AAS and RDF-based models have their own strengths. AAS models are easier to integrate with operational technologies in a production environment, whereas RDF-based models offer more semantic expressiveness and advanced querying. In the Horizon MAS4AI project we found that both modelling paradigms can complement each other to develop agentbased digital twins for modular production environments. In this work we propose two different approaches to bridge both modelling paradigms. First we define a set of mapping rules to generate an AAS model from a given RDF-based model, supporting model development. Secondly, we propose to use RDF-based models to generate a digital shadow of AASs to improve semantic discoverability. Preliminary results demonstrate that heterogeneity of metamodels does not exclude achieving semantic interoperability, as well as that greater functionality can be obtained compared to using both models in isolation. The solutions will be further developed in collaboration with pilot lines in the MAS4AI project.

1. Introduction

The physical and digital world are getting increasingly intertwined. This may go for society as a whole, and surely applies to the manufacturing industry with the advent of digital twinning, Cyber-physical systems, and connected and digital factories all contributing to the Industry 4.0 movement being presented in various projects and initiatives. However, besides the increasing prevalence of IT systems and digital solutions to control the manufacturing environment, we are also seeing an increasing number of standards, models, and metamodels being used to describe concepts in and related to the manufacturing environment.

In our experience, defining a single semantic model to guarantee uniformity in definitions and modelling is an utopia and instead a world with data heterogeneity is unavoidable. As such, metamodels fulfil an important role as abstract language to express all these different domain and application specific data models. Designing, developing, deploying and testing multiple metamodels, which can support each other, in an interoperable solution will be vital to keep up momentum in the digitization of industry and achieving the promises of Industry 4.0. Without connected and reusable information models different initiatives all have to redo already available work and risk development into isolated silos that cannot easily be connected to other initiatives, which would greatly reduce interoperability of industry as a whole.

One approach to tackling the challenge of semantic interoperability is the meta language of RDF (the standard model for Linked Data). This has been gaining traction in recent years in a variety of domains. Examples are the adoption of the European commission data portal, which can be accessed through SPARQL (P. O. of European Union, 2017) and Building Information Management, through initiatives such as IFC (Technical Committee, 2018); the ISO 21597–1 standard (Technical Committee, 2020), using an RDF-based model (Nederveen et al., 2010). However, the adoption of RDF in the manufacturing industry has been slower (Schröder et al., 2021). This is not due to lack of models within industry (see Table 1 from (Beden et al., 2021) for examples), but due to the use cases in which those models are needed. We observed this in the MAS4AI (MAS4AI Consortium, 2020) project, part of the European HORIZON 2020 program as well. As recognized by Hildebrand et al (Hildebrand et al., 2019)., in industry there are either already established standards in place, such as OPC companion specifications, or in other cases proprietary solutions are used, such as those provided by large manufacturing equipment providers. This makes it complicated to

☆ Funded by MAS4AI Horizon 2020 research and innovation program under grant agreement No. 957204.
* Corresponding author.
E-mail addresses: sjoerd.rongen@tno.nl (S. Rongen), nikoletta.nikolova@tno.nl (N. Nikolova), mark.van.der.pas@semaku.com (M. van der Pas).

https://doi.org/10.1016/j.compind.2023.103910
Received 7 October 2022; Received in revised form 24 February 2023; Accepted 27 March 2023
Available online 4 April 2023

Table 1
Overview of rules (SWRL human readable syntax (Horrocks et al., 2004)) implemented in the tool to convert an RDF-based model into a AAS template.

#	Rule antecedent	AAS element [Class]
1	rdf:type(?x, sh:NodeShape) ∧ mas4ai:hasInterface(?x, ?i)	Asset Administration Shell [aas:AssetAdministrationShell]
2	rdf:type(?x, sh:PropertyShape) ∧ (not sh:class(?x, ?c)) ∧ (not sh:datatype(?x, rdf:langString))	Property [aas:Property]
3	rdf:type(?x, sh:PropertyShape) ∧ sh:datatype(?x, rdf:langString)	Multi Language Property [aas:MultiLanguageProperty]
4	rdf:type(?x, sh:PropertyShape) ∧ sh:class(?x, ?c) ∧ mas4ai:hasInterface(?c, ?i)	Reference Element [aas:ReferenceElement]
5	rdf:type(?x, sh:NodeShape) ∧ (not mas4ai:hasInterface)(?x, ?ix) ∧ sh:targetClass(?x, ?c) ∧ sh:class(?p, ?c) ∧ sh:property(?n, ?p) ∧ mas4ai:hasInterface(?n, ?in)	Submodel [aas:Submodel]
6	rdf:type(?x, sh:NodeShape) ∧ (not mas4ai:hasInterface)(?x, ?ix) ∧ sh:targetClass(?x, ?c) ∧ sh:class(?p, ?c) ∧ sh:property(?n, ?p) ∧ (not mas4ai:hasInterface(?n, ?in))	Submodel Element Collection [aas:SubmodelElementCollection]
7	rdf:type(?x, sh:PropertyShape) ∧ (not sh:maxCount(?x, 1))	Submodel Element Collection [aas:SubmodelElementCollection]
8	rdf:type(?x, sh:NodeShape) ∧ mas4ai:hasInterface(?x, ?i) ∧ sh:property(?x, ?p) ∧ sh:datatype(?p, ?d)	Submodel [aas:Submodel]

migrate to RDF-based semantics. Currently, most use cases for RDF focus on descriptive data, which is updated with low frequency and exist fully in the digital domain. Industry, however, needs a model which can be connected directly with the physical world, where the conditions of a machine are tracked and sensors are responsible for constant large data streams using already implemented communication standards. To meet the requirements of industry applications, the Platform Industry 4.0 has put forward its own semantic metamodel - the Asset Administration Shell (AAS) (Bader et al., 2022, 2021). This model is quickly gaining traction in both small and large manufacturing companies. Although the AAS metamodel differs to some extend from RDF, it can still be expressed in RDF (Industrial Digital Twin Association, 2022).

The problem we are now faced with is in having two metamodels available that could reinforce each other, but in practice they are used separately from each other. This leads to the unnecessary recreation of information models which are already available, and the adoption of both metamodels suffers by providing only a sub-set of the required functionality in industry. One of the functionalities often needed, but only offered by RDF, is querying. The AAS server and registries do not support semantic queries, in the sense that there is no straightforward way to look up assets and AAS elements based on their semantic identifier. Based on the challenges identified above, we formulated two questions that are explored in this work:

1. How can RDF-based models be reused in the modelling of Asset Administration Shells, such that
   (a) the logical structure of the original model is maintained;(b) the AAS model includes semantic provenance.
2. How can Semantic Web Technologies, such as RDF and SPARQL, be used to improve the discoverability of assets in a distributed Asset Administration Shell environment, to enable
   (a) integrated querying the AASs and some information model;
   (b) querying in terms of the data model referred to by the semantic identifiers.

In this paper we discuss the merits of both the RDF and AAS metamodels and explore ways how the models can complement each other. We contribute to the current state of the art research by presenting two novel approaches to enhance the interoperability between RDF- and AAS-based models and preliminary results of their implementations. The first approach describes how to incorporate RDF-based data models in AAS modelling, and the second describes how RDF can be combined with AAS to improve discoverability of assets in Industry 4.0. We provide a methodological description and proposed practical integration connected to the development process we are currently in.

We present these ideas based on the experience obtained in the MAS4AI project, in which research is conducted on the application of multi-agent systems (MAS) in modular production environments. This includes the ability to query a general knowledge base describing the system, as well as contacting individual agents and assets via their AAS to coordinate and collaborate on the execution of specific tasks. Within the MAS4AI project the semantic models fulfil a crucial role in enabling a plug & play solution where agents may configure themselves and adapt to the conditions of the framework in which they are deployed. These models enable the adoption of digital twin technology in e.g. the automotive sector, lowering lead times in the low volume and high complexity manufacturing industry, and support predictive maintenance of machines in high volume production, leading to optimized production processes with less downtime. The approaches proposed in this work are tested in collaboration with pilots involved in the project.

The paper is structured in the following way. Section 2 outlines the current state-of-the-art research related to using AAS as a modelling technology in manufacturing and how it can be combined with RDF. Section 3 provides background information and specifications of the two metamodels respectively, including existing methods to combine them. Section 4 describes the two approaches, providing theoretical foundations and definitions. Section 5 describes an implementation of the proposed methods and preliminary results. Section 6 discusses the findings, their limitations, and recommendations for next steps. Finally, Section 7 concludes with a summary of the obtained insights.

2. State of the art

Currently within industry there is a variety of information models and data representation methods which implement little or no general semantics. However, to be able to achieve interoperability and create reusable and meaningful models, it is essential to add a semantic layer. There are different methodologies, such as the one proposed by Hildebrand et al (Hildebrand et al., 2019). that aim to convert existing data sources in factories into RDF (RDF Working Group, 2014). Or the structure provided by the Reference Architecture Model Industry 4.0 (RAMI4.0), in which the information layer explicitly positions the need for information modelling. The German organization Platform Industry 4.0 has defined its own semantic model, the Asset Administration Shell (AAS) (Bader et al., 2021; 2022). This model can be used to describe different types of assets in a production environment and should improve interoperability in industry. The work of Kamburjan et al (Kamburjan et al., 2022). tackles this issue by presenting a method for structural reconfiguration of digital twins using asset models in combination with semantic web technologies. Overall research has been conducted on how to not only add semantics to industry, but also how to bridge the gap between the two commonly used models - RDF and AAS.

The first work that tried to connect the AAS and RDF modelling worlds by Grangel-Gonzalez et al (Grangel-Gonzalez et al., 2016a). focused on how to create an RDF-based model for an AAS. They present a RDFS-based vocabulary, which provides definitions for all AAS components, and a practical example of the semantic mapping for the modelling of a servo motor controller. This work is the first to address the question of how the benefits of RDF can be transferred towards an AAS model. A method is discussed which introduces a semantic layer to an AAS, now commonly called the *Semantic AAS*. This concept is further extended by Bader and Maleshkova (Bader and Maleshkova, 2019), who propose a method to map an existing AAS into an RDF Schema, such that it is possible to query the created model. This way the benefits of having an RDF-based model, such as being able to query and reason about the data, are transferred to the data encapsulated in the original AAS. This work is the first to present a mapping from an XML- to RDF-based model for the purpose of generating a semantic AAS.

RDF is used more commonly as a way to enrich the AAS model by providing a semantic definition for all the elements of the model. This approach, employed by Grangel-Gonzalez et al (Grangel-Gonzalez et al., 2016b)., demonstrates one of the main ways in which RDF can be used in this context. In their work the authors present how to create semantically enriched AASs for the use case of modelling sensors in legacy systems. They argue that combining RDF and AAS can bring benefits such as smooth integration of existing standards like OPC UA, AutomationML, and ECLASS; a unified way to represent and identify relevant entities; and simplified integration of data from different objects.

Another solution to represent semantics in industry data has been proposed by Textor et al (Textor et al., 2021). In their work the authors introduce the BAMM Aspect Meta Model (BAMM), which represents a digital twin including semantics. An *aspect model* contains both runtime and non-runtime data about a given asset, and is in that way similar to the AAS. BAMM represents an aspect model defined through an RDF vocabulary and SHACL rules. It can represent information about a digital twin in the shape of a graph. Recently, there have been developments towards converting BAMM to AAS, so that the AAS models can benefit from the already expressed semantics .[1]

Ocker et al (Ocker et al., 2021). provide a framework for checking compatibility between components by representing existing Digital Twins as semantic AAS models and converting them to RDF. They use the new representation to run set of checks using SHACL shapes and SPARQL queries. Here the AAS serves as a connection element between the physical asset, and the semantic model that can be reasoned over. This supports our position, that the AAS and RDF can be used to complement each other. The concept of transforming AAS models to different formats and increasing its capabilities is further explored in the work of Braunisch et al (Braunisch et al., 2021)., where an approach is

[1] https://industrialdigitaltwin.org/en/news-dates/idta-and-omp-collaborate-3837

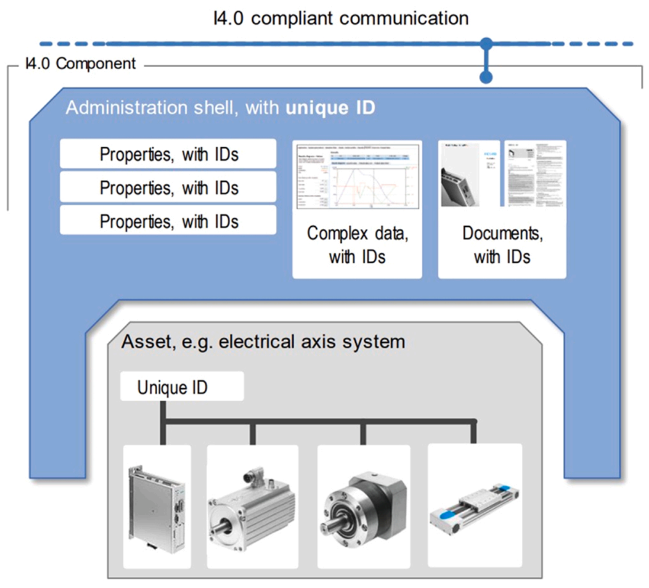

Fig. 1. Structure of an AAS (Bader et al., 2022, 2021).

presented towards producing SDKs in different languages, however do not yet tackle the topic of RDF aside from mentions for future research.

At the moment of writing, to the best of our knowledge, there is no other work which discusses how AAS models can be created from RDF-based schemas, as we propose in this paper. The current state-of-the-art looks at how to create an RDF representation of an AAS (Semantic AAS) or how an AAS model can be mapped onto an RDF-based schema, but not the other way around - the generation of a basic AAS model from an already existing RDF-based model. The development which can be considered similar is the generation of AAS descriptions from BAMM models.[2] This, however, is limited to models based on BAMM and the approach cannot be replicated for more general RDF-based models. Our approach goes further and allows one to reuse the large amount of already existing semantic models in RDF and reference to their URIs directly. This would greatly increase the number of easily implementable AAS models, while preventing divergence in semantic models by remodelling the same concepts.

Similarly, the notion of using RDF next to AAS as a storage for different types of information has not been investigated. What is significantly different in this approach compared to the current state-of-the-art is that, instead of taking one of the semantic models as starting point and mapping it into the other, the method focuses on how the RDF and AAS can be used in parallel. The main benefit of this approach is that it can combine the benefits of both metamodels, by reusing data and functionalities offered by the other metamodel. As the RDF and AAS models represent different sets of data, one describing the environment and providing high-level descriptions (RDF) and one with more operational machine data (AAS). In this case the RDF graph provides a kind of search index for AASs that goes beyond the standard AAS capabilities, and makes it more straight-forward and efficient to query and discover AASs. This enables the usage of AAS in larger scale applications. Both of those novel approaches are important in improving the integration of semantics within industry 4.0 models.

3. The structure of the metamodels

In this section we outline the formal structure of the two metamodels RDF and AAS by providing relevant definitions, important specifications and other details which are key to understanding the methodologies

[2] https://github.com/OpenManufacturingPlatform/sds-sdk/issues/113

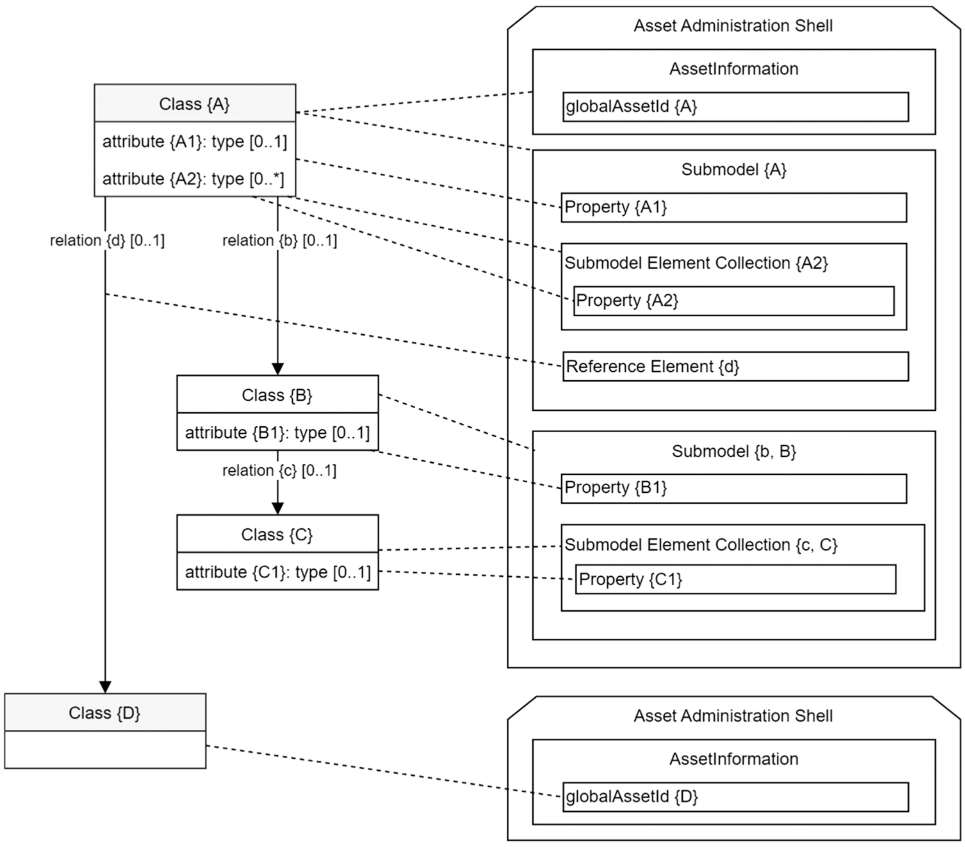

Fig. 2. Overview of mapping from UML class diagram to components in the AAS.

presented in this work. Furthermore, we also discuss the current most common approach towards using RDF and AAS within Industry 4.0.

3.1. *The resource description framework*

The resource description framework (RDF) (RDF Working Group, 2014) is a standard for describing resources and defining ontologies in the semantic web. It provides a high level meta language, commonly extended with the Web Ontology Language (OWL) (OWL Working Group, 2012) or Shapes Constraint Language (SHACL) (RDF Data Schapes Working Group, 2017). The metamodels can be used to describe concepts, their properties and the relations between them. They are structured as graphs, often referred to as knowledge graphs due to the extensive context they can provide for even a single instantiated data point. These semantic web ontologies, or RDF graphs, may adhere to FAIR (Wilkinson et al., 2016) data principles. The usage of (dereferenceable) unique resource identifiers (URIs), in combination with ontologies, make data findable, accessible, interoperable and reusable.

The RDF graphs can be queried or manipulated using SPARQL (SPARQL Working Group, 2013). Given, that within the semantic web data models are considered first class citizens, the SPARQL query language can be used to query the data model (ontology) used, just as easily as it may be used to query the instance data being described in a graph. This flexibility in combining instances, models and other meta data provides a powerful way of representing and using the knowledge modelled in the graph.

RDF-based ontologies are already used in a variety of domains such as public governance (P. O. of European Union, 2017), life science (Belleau et al., 2008), or descriptions of ”common knowledge” (Wikidata gorup, 2019). Moreover, these different initiatives relate to each other and may reference each other’s definitions, as is presented well by the linked open data cloud (LOD cloud group, 2020).

3.2. *The Asset Administration Shell*

The Asset Administration Shell (AAS) is an information model which describes an asset, such as a manufacturing resource, a product, a piece of software, a process, etc. This description is a vital part of a digital twin in which both the data, as well as the models and applications using that data, require a semantic description to be reused for various applications. The structure of the created model, shown in Fig. 1 contains a collection of different generic and specific submodels, which together create the complete asset description. Each submodel contains a set of submodel elements of different types. For example, datatype properties with a literal value are modelled as a Property, while references to other AASs (elements) or external objects are modelled using a Reference Element. Multiple submodel elements can be bundled in a Submodel Element Collection. Each submodel template can be reused between different AASs, contributing to a library of common and standardized semantic data-models.[3] For more details about the different AAS element types and their usage we refer to the specification (Bader et al., 2022) and RDFbased ontology.[4]

Concepts and relations in the AAS can have a semantic identifier, which is a global unique URI, that can be used to refer to external concept definitions, which can be a standardized submodel from some library or another type of data model, like an ontology. Besides these metadata, the AAS also allows adding supporting documents to the shell,

[3] https://admin-shell-library.eu/ is a repository containing template models
[4] https://github.com/admin-shell-io/aas-specs/blob/master/schemas/rdf/rdf-ontology.ttl

such as 3D models or user manuals. Finally, the AAS models can be uploaded to an AAS server which exposes standardized APIs for interaction. Clients can interact with the AAS model, as well as its instance data using REST.[5] In the background the AAS server can integrate to a machine for example using the common OPC UA[6] protocol to synchronize data between the physical asset and its digital twin AAS and also control the asset. For more detailed specifications, please refer to Bader et al (Bader et al., 2022, 2021).

Despite the structures and semantics supported by the AAS metamodel, it lacks features one may expect when used to RDF-based ontologies. Specifically, the AAS is not strong typed, meaning it can be difficult to determine what type of individual one is looking at. RDF offers us a model to add typing to individuals and solve this issue. Next to this, Bouter et al (Bouter et al., 2022). concluded that the AAS model is currently not suitable for querying large sets of AASs. Mainly because interaction happens through the standardized, but less flexible predefined API. Lastly, as the AAS is still relatively new, the tools for model creation are not mature and the set of standardized models is still limited.

3.3. *RDF as a semantic base for AAS*

The AAS metamodel has an element specifically for defining semantic identifiers. In most cases those identifiers are used to refer to external definitions in standards like ECLASS (Belyaev et al., 2021). Ideally, a semantic identifier is provided for every element in the AAS, which refers to organization internal or external standardized data models. The semantic identifiers are conceptually similar to the URIs used in RDF modelling and serve to reference a larger semantic definition from a small local model definition. The most basic way to bridge from an AAS to RDF model is by using already existing URIs as semantic identifiers. This allows users of the AAS to find and use the concepts defined on the semantic web for further definitions or relations. This is part of the AAS modelling approach, originally presented by Bouter et al (Bouter et al., 2021). Within the MAS4AI project, this method is applied to add references from AAS concepts to the larger standards on which the models are based, such as the FIPA multi-agent modelling standard .[7] However, as the current AAS tooling is limited in how it makes use of these semantic identifiers, for example, current tooling does not support derefencing them to present the external data, the usage of this approach in practice is limited to purely semantic applications. Developers in practice often don't need, or see the use of, using these semantic identifiers beyond them being just a unique identifier used within the AAS model. Moreover, even when adopting this approach fully, it is limited due to the lack of support for using the semantic identifier in the AAS API, making it difficult to use for anything other than referencing from an AAS model to a larger semantic knowledge base.

4. Methodology

We propose two methods for integrating RDF and AAS, which leverage the benefits of both metamodels. The two described methods aim to keep semantics easily usable through the standard model and interfaces provided by the AAS, while also building on the rich semantics available in RDF based ontologies. In this section we outline the foundation of our ideas and in the next section we explain the preliminary implementation which we are developing in the MAS4AI project.

4.1. *Generating an AAS template from RDF-based data models*

Creating AAS models can be difficult, due to a lack of tooling, it being a niche domain and the general complexity of information modelling. As such, tools that make it easier to make an AAS model, or reuse already existing information models would be highly beneficial in speeding up development, improving model quality and ensuring semantic interoperability between models. One of the ways to aid the creation of AAS models would be by supporting the easy reuse of existing models. Pakala et al (Pakala et al., 2021). propose a mapping from the Web of Things Thing Description, which is an RDF-based data model, to AAS submodels. Similarly one of the AAS developer tools contains a solution to import an AAS from a model defined according to the BAMM.[8] We take this idea a step further and argue that a general RDF-based semantic model (ontology) can be used to generate a basic AAS (template). An automated conversion from a semantic model to an AAS helps to reuse existing knowledge and ensures the semantic references are aligned with existing data models. The generated AAS should contain the information from the semantic model it is generated from, but can be extended with additional (standardized) submodels for aspects that are not covered by the existing model.

The starting data model can be in different shapes and formats, so we describe some fairly generic mapping rules below that are always applicable. The generic rules can be formalized and tailored to the specific modelling language that is implemented, for example OWL 2 (OWL Working Group, 2012) or SHACL (RDF Data Schapes Working Group, 2017). Fig. 2 gives an overview of the mapping concept and rules described below, following Requirement 1a.

1. The class for which we generate an AAS is mapped to a submodel. The other classes are only considered if they are in the domain or range of a relation.
2. For relations two main cases can be distinguished:
   a. If the relation has as domain the class for which an AAS is generated, it is mapped to a submodel.
   b. Otherwise, the relation and its range class are mapped to a submodel element collection. This collection can in turn have nested submodel elements mapped from the relations with domain the class corresponding to this collection.
3. Relations that have as range another class for which an AAS is constructed, are mapped to a reference element.
4. Attributes have the most straightforward mapping, as they have as range a value with a certain type. In general, attributes therefore map one-on-one to a property submodel element.
5. Relations and attributes with multiplicity bigger than one are embedded in a submodel element collection.

Besides the mapping rules described above, it is important that the AAS elements contain a reference to the (unique) identifier for the class, relation, or attribute it was mapped from. This reference ensures that it is possible to find the semantic definition and context of the element (Requirement 1b. Besides that, the semantic identifiers can also be used for integrating data from different AASs using the approach we describe in Section 4.2.

The main advantages of this approach are that throughout the conversion process the identifiers from the ontology are preserved and the URIs from the ontology are used as semantic identifiers within the AAS model. This allows the continued usage of the RDF graph as a semantic base if desired. Furthermore, it aids modellers and speeds up the modelling process by generating (partial) AAS templates from already existing (formal) models.

[5] https://restfulapi.net/
[6] https://opcfoundation.org/about/opc-technologies/opc-ua/
[7] http://www.fipa.org/index.html
[8] https://openmanufacturingplatform.github.io/sds-documentation/ sds-developer-guide/tooling-guide/java-aspect-tooling.html#mapping-aas

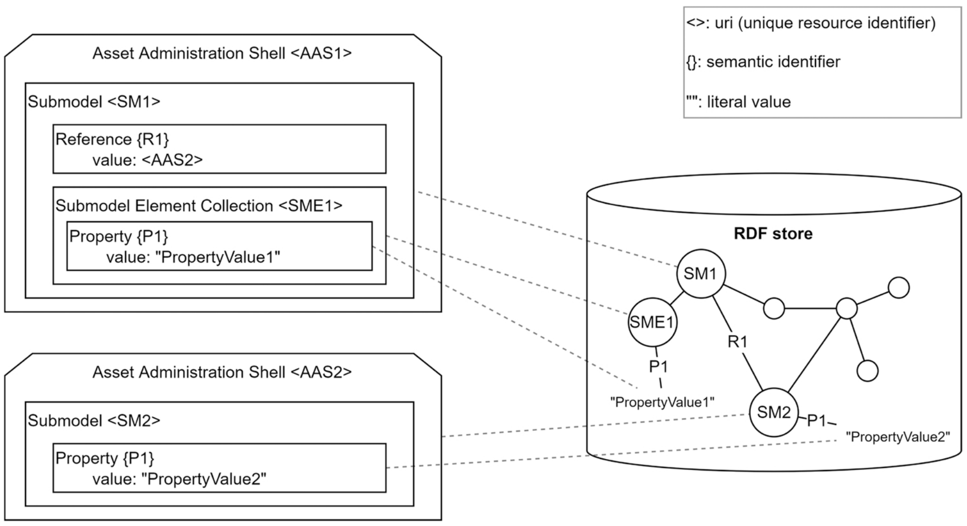

Fig. 3. Relating AAS elements to an RDF graph.

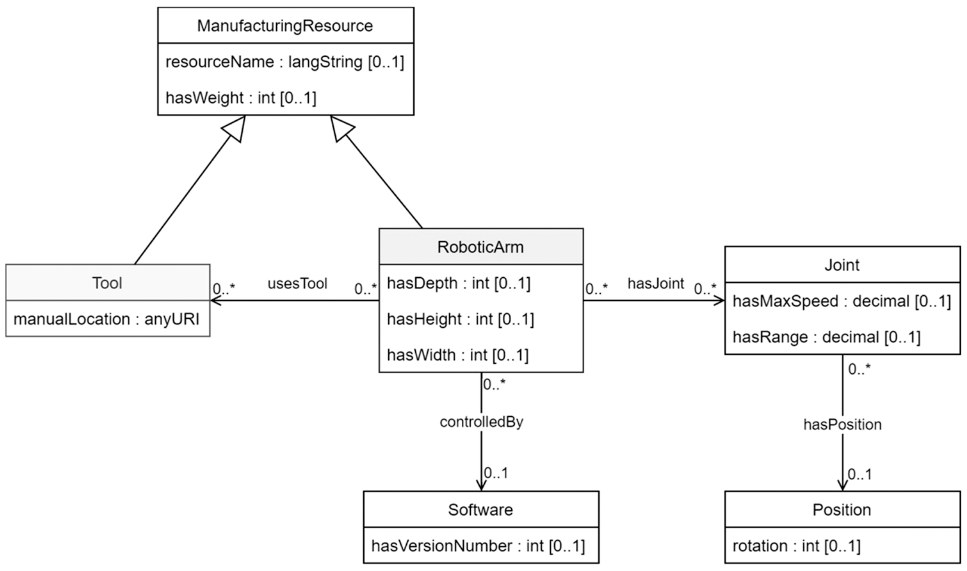

Fig. 4. UML diagram of the example robotic arm model (gray classes are the assets for which an AAS is generated).

4.2. RDF store as a search index for AASs

Whereas the method described above aids in the design time of an AAS model, this approaches focusses on the interaction that can happen between the two metamodels at runtime. They can be used together in a single deployment where both modelling methods reinforce each other. Bouter et al (Bouter et al., 2022). explore the question of how to query AASs and conclude that currently the AAS is not suitable to be queried in large numbers. For a single AAS it is possible to convert it to an RDF graph (Bader and Maleshkova, 2019) and then query it. However, the proposed model is an almost one-on-one mapping of the AAS metamodel, which contains much indirection and make it difficult to query as a graph. We argue that the AAS data can be mapped to an RDF graph according to a custom RDF-based model, when semantic identifiers are present that refer to this model.

When there is a need to search through a large set of AASs, we argue that an RDF store can be of added value. Let us consider the use case of a MAS controlled manufacturing environment. Each agent in the environment is represented by an AAS, combined with a real-time connection to the physical or digital factory component. Different agents will implement a different set of skills depending on their type. To find an agent that has the exact required capability, we can create a dedicated graph containing the relevant information needed for the MAS to query. Using the RDF store we can easily find the small set of AASs of interest and then connect to these directly for more detailed information about the agent.

More general, we propose to use an RDF store to create an integrated view of the assets and their properties exposed via an AAS. The integrated view can then be used to answer more complex questions (Requirement 2a), like the one described above. One of the main advantages of RDF is its natural graph structure, which gives the flexibility required for integrating data in a dynamic agent-based modular

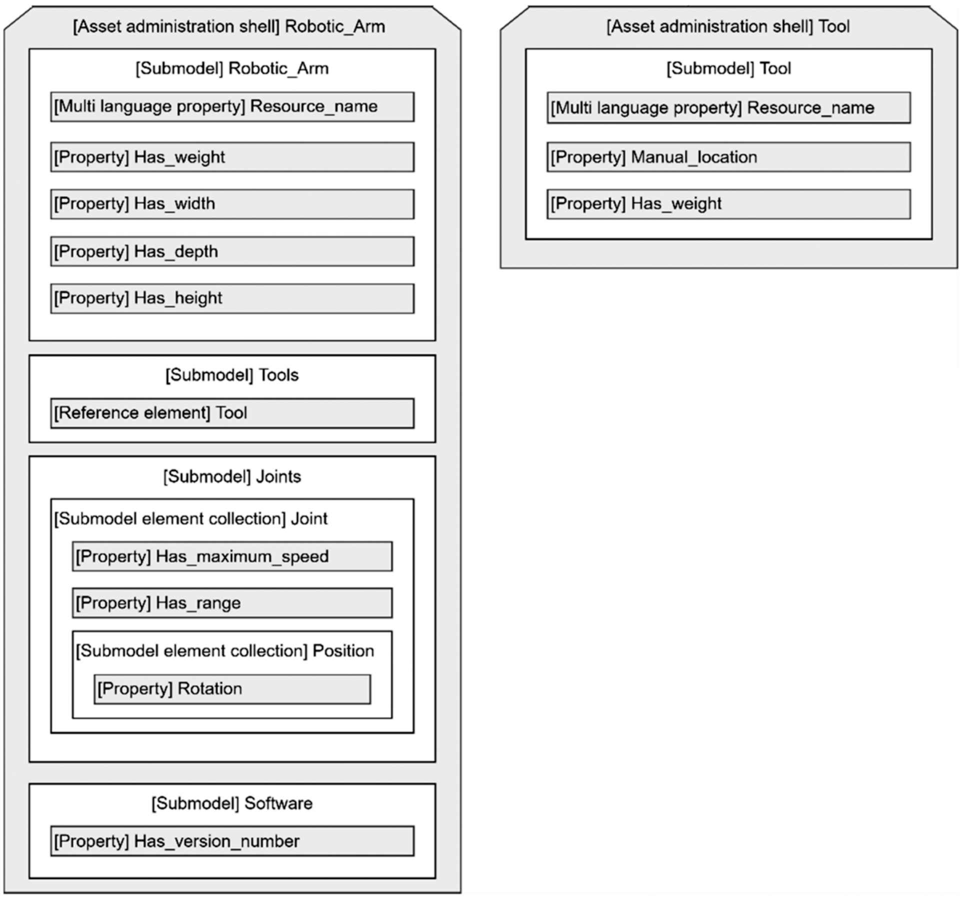
[Asset administration shell] Robotic_Arm
[Submodel] Robotic_Arm
[Multi language property] Resource_name
[Property] Has_weight
[Property] Has_width
[Property] Has_depth
[Property] Has_height
[Submodel] Tools
[Reference element] Tool
[Submodel] Joints
[Submodel element collection] Joint
[Property] Has_maximum_speed
[Property] Has_range
[Submodel element collection] Position
[Property] Rotation
[Submodel] Software
[Property] Has_version_number
[Asset administration shell] Tool
[Submodel] Tool
[Multi language property] Resource_name
[Property] Manual_location
[Property] Has_weight

Fig. 5. AAS templates generated from the robotic arm ontology (the AssetInformation components are omitted for brevity).

production environment. For example, to deal with a situation where new types of agents are integrated into the system.

We envision a future system that populates the RDF store with data from the AAS according to a data model which references semantic identifiers used on elements in the AASs. Fig. 3 gives an impression of how AAS elements map to statements in the RDF store (Requirement 2b). To keep the RDF store up to date, we need to synchronize data when updated in the AAS. For this purpose, an event driven approach can be used, where events are generated when certain AAS elements change (Bouter et al., 2022). With a synchronization mechanism in place the RDF graph can be used as a digital shadow that the agents can use for decision making.

5. Implementation

In the previous section, we described the theoretical foundation behind our proposed methodologies. In this section we will present our current implementation and corresponding results as achieved within the MAS4AI project. As this is an active area of development we present only the current work and in Section 6 we will evaluate what are the next steps.

5.1. Generating an AAS template from RDF-based data models

To support the development of AAS models, and support semantic interoperability between different models we developed a tool to automatically convert from an RDF-based model to an AAS template model. The implementation can generate an AAS template from SHACL-based models. SHACL (RDF Data Schapes Working Group, 2017) is a more recent addition to the semantic web stack and we observe that it is used more regularly in the manufacturing domain. Furthermore, we argue that it fits better with the (current version of the) AAS metamodel, which has only limited expressiveness, for example it does not support complex value constraints that can be defined in OWL (OWL Working Group, 2012) ontologies. The tool implements the mapping rules proposed in Section 4.1. Table 1 gives an overview of the most important rules implemented by the tool. The right column shows the AAS element to which resources in the SHACL Shapes graph are mapped that adhere to the rule defined in the left column.

Rule 1 defines the objects to map to an AAS. The custom property mas4ai:hasInterface is used to mark the node shapes (representing an asset) for which the tool should generate an AAS. Rule 2 and 3 defines that datatype property shapes should be mapped to either a Property or MultiLanguageProperty, if the datatype is a language string. Property shapes that refer to a class that is mapped to an AAS, a ReferenceElement is created (rule 4). If the shape refers to a class that is not mapped to an AAS, then the tool maps it to a Submodel if the property has domain an object that is mapped to an AAS (rule 5) and to a SubmodelElementCollection (SMC) otherwise (rule 6). Property shapes with cardinality not equal to one are nested in a SMC, so rule 7 defines that for those shapes an SMC will be created. Rule 8 defines the creation of a Submodel for all datatype properties with domain an ’AAS class’. All rules to construct the components are defined in SPARQL CONSTRUCT queries. Subsequently, a set of SPARQL INSERT queries are used to add the relations between the components. The tool is implemented using Python RDFLib.[9] The source code and SPARQL queries are available on Github.[10]

We demonstrate this tool using a simple robotic arm ontology (see Fig. 4), however, the underlying logic is generic and supports the model constructs described in Table 1. Besides the robotic arm, we can also tell the tool to create an AAS for the Tool class. Running the publicly available code we find the AAS models provided in Fig. 5. Both the shapes and the resulting AAS templates can be found in the Github repository.

As expected, we find two AAS models, one simple model for the tool containing only one submodel and a slightly more elaborate AAS model for the robotic arm containing more submodels and properties. Although this may not provide a full description of the asset it provides a starting point to the modeller which is fully aligned with an earlier model, making it simple to reuse pre-existing standards and nudging the modeller into AAS modelling best practices, such as breaking up concepts into separate submodels.

Within real life manufacturing environments there may be a vast amount of assets, varying from software agents to manufacturing equipment to the products being made. As such, speeding up the modelling process may make the difference between proper semantic models being prohibitively expensive to make and them being the easy solution to representing some concept in the organisation.

5.2. RDF store as a search index for AASs

In the MAS4AI project we designed a solution that implements the method described in Section 4.2. The solution contains an RDF graph according to a semantic model, as well as an AAS server containing the AAS models for individual assets. These AAS descriptions may be fast to update, contain a large number of data elements and can be integrated with operational systems, such as manufacturing equipment or software agents. However, the AAS does not support directly querying for assets matching certain properties. For this, the RDF store is used, it describes the environment and the slower to update, or completely static, asset data. These data can be queried to find individual assets of interest, which identifiers point to the AAS model of these assets. This provides the flexibility, interoperability and reasoning power of RDF for the purpose of discovery. While also leveraging the ease of integration and compatibility of the AAS.

The downside of this approach is that when updating metadata it is not as clear where this new information should be stored - is it to be updated in the AAS, the RDF, or both? This may differ dependent on the implementation and, although it could be determined by querying the model, we designed a generic solution which abstracts this synchronisation away from the individual agents and instead constructs the RDF graph based on the AAS model. This approach works by duplicating relevant information from its AAS representation to the RDF knowledge base to enable further analysis. The primary advantage is that it simplifies integration for agent developers, who only need to bother with the integration of the AAS and the asset it represents. In our implementation we assume that the AASs are the source of which a subset is synchronized to the RDF store. The current implementation only supports pull-based updates of the data in the RDF store, and the complete AAS representation has to be recreated when an AAS element is updated. Note that every AAS is stored in its own graph context, which makes it straight-forward to insert, update, or delete the representation, for example using the Graph Store Protocol.[11]

Algorithm 1. Construct RDF graph for AAS elements for $e \in AAS$ do.

```
$e \in AAS$ do
    $s \leftarrow$ semanticIdentifier($e$)
    if $s \in G_{model}$ then
        if rdf:type($s$, owl:Class) then
            $G_{AASs} \leftarrow G_{AASs}$+( iri($e$), rdf:type, $s$ ) else if
        rdf:type($s$, owl:DatatypeProperty) then
            $G_{AASs} \leftarrow G_{AASs}$+( iri($e$), $s$, value($e$): valueType($e$) ) else if
        rdf:type($s$, owl:ObjectProperty) then
            $G_{AASs} \leftarrow G_{AASs}$+( iri(parent($e$)), $s$, iri($e$) ) end if
    end if end for
```

Algorithm 1 gives a simplified view of the logic to construct an RDF graph for AAS elements and their semantic identifiers. The algorithm loops over all elements in the AAS ($e \in AAS$), if the semantic identifier (s) related to the element exists in the given RDF-based model (G_{model}), then depending on the type of the semantic identifier a different statement (subject, predicate, object) is added to the graph G_{AASs}. The following functions are used:

- iri(e): get the identifier iri of the element e;
- value(e): get the value of the property element e;
- valueType(e): get the value type of the property element e;
- parent(e): get the parent element of the element e in the AAS.

5.2.1. Discoverability example

We demonstrate the advantage of an RDF graph as a search index on AASs with a small example, based on the robotic arm AAS template presented in Section 5.1. Suppose we have a production environment with multiple robotic arms and instantiations of the robotic arm AAS

[9] https://github.com/RDFLib/rdflib/
[10] https://github.com/gitmpje/shacl2aas
[11] https://www.w3.org/TR/sparql11-http-rdf-update/

template. Next to that we have a simple (SKOS[12]) taxonomy of different tools. Subsequently, we apply the mechanism proposed above and for example the ex:hasTool, ex:toolType, and ex:maximumWeight properties are indexed in the RDF store. Now, we can easily find all AASs for the robotic arms with some kind of ex:Gripper that can carry a weight above a certain value using one SPARQL query on the integrated data set that includes a property path on the SKOS taxonomy.

```
 PREFIX ex :       http :// example . org/

SELECT ?Asset
WHERE {
        ?Asset a ex : RoboticArm ; ex : hasTool            [ ex :
                toolType/skos : broader+ ex : Gripper    ; ex
                :maximumWeight ?maxWeight ; ]             .
        FILTER( ?maxWeight >= 200 )
 }
```

In comparison if we would only rely on the standard capabilities of the AAS server and registry (Bader et al., 2021), we would have to make a request to all AASs to retrieve their type and weight, and subsequently make the comparison to our defined values.

5.2.2. *Business analytics example*

The RDF store contains an integrated overview of all the assets and can be used to obtain managerial insights, which often involve aggregation. A typical example of this is to count the number of assets that can do a certain operation. The query below could be used to count the number of robotic arms for every type of tool.

```
PREFIX ex: <http://example.org/>

   SELECT ?Type ( COUNT(DISTINCT ?Asset) AS ?count)
   WHERE {
      ?Asset a ex:RoboticArm ;
         ex:hasTool/ex:toolType ?Type .
   }
   GROUP BY ?Type
```

5.2.3. *Operational insights*

When the RDF graph contains real-time information about the state of the assets, this information can be used to obtain insights for operations planning. For example, the query below lists all robotic arms with an ex:TwoFingerGripper, that are idle and can be used to plan certain operations on.

```
PREFIX ex: <http://example.org/>

SELECT ?Asset
WHERE {
   ?Asset a ex:RoboticArm ;
      ex:hasTool/ex:toolType ex:TwoFingerGripper ;
      ex:status ex:Idle .
}
```

5.2.4. *Reasoning*

One of the benefits of using RDF in combination with (OWL) ontologies is the reasoning capability. This could be applied, for example, for the conversion of unites of measure. Suppose that grippers from different suppliers are used that define their specification using different units of measure. If those units are defined using a units of measure

[12] https://www.w3.org/TR/skos-reference/

ontology like qudt,[13] then a reasoner can be used to automatically convert the metrics to the same unit and compare the specifications.

Another use case for reasoning and the application of ontologies is the alignment of vocabularies. In a scenario where tools from multiple suppliers are used, the suppliers will often use a different vocabulary for describing the tool. The alignment of the vocabularies can be done based on an ontology, which can for example define that both *ex:maxWeight* and *ex:maximumCarryWeight* are sub properties of *ex:maximumWeight*.

5.2.5. validation

SHACL can be sued to define constraints on the data that is available through the graph. A typical is the constraint on the cardinality of certain properties. For example, we can say that every robotic arm should have exactly one status out of the set of possible statuses.

```
PREFIX ex: <http://example.org/>
PREFIX sh: <http://www.w3.org/ns/shacl#>

ex:StatusShape a sh:PropertyShape ;
  sh:path ex:maximumWeight ;
  sh:minCount 1 ;
  sh:maxCount 1 ;
  sh:or (
    [
      sh:class ex:Idle ;
    ]
    [
      sh:class ex:Operating ;
    ]
  ) .
```

Besides data validation SHACL can also be used to define and validate operation constraints. In combination with logic problem solvers, it can also be used to generate configurations (Bischof et al., 2018). The example shape below defines that a robotic arm can only have at maximum one *ex:TwoFingerGripper* tool.

```
PREFIX ex: <http://example.org/>
PREFIX sh: <http://www.w3.org/ns/shacl#>

ex:GripperTypeShape a sh:NodeShape ;
  sh:targetClass ex:RoboticArm ;
  sh:property [
      sh:path ex:hasTool ;
      sh:maxCount 1 ;
      sh:class ex:TwoFingerGripper ;
  ] .
```

[13] https://www.qudt.org/doc/DOC_VOCAB-UNITS.html

As these examples demonstrate the an RDF graph which can be queried, or even used for data validation enables valuable use cases in Industry which cannot be implemented using the AAS alone.

In the MAS4AI project we test the proposed design, and specifically the advantage of the using the RDF store for the purpose of AAS discovery. Whether the AAS to RDF implementation as described in Section 5.1 is used is left up to the agent developers as it is an optional implementation of the data synchronisation implementation of the framework. The RDF graph for searching is deployed in the pilot lines which all contain multiple agents of multiple different types as well as potentially a number of physical assets. This situation is representative for a production environment regarding the interaction between RDF and AAS and the difficulty in finding and integrating with specific assets.

6. Discussion and limitations

In this paper we explored how the meta languages of RDF and AAS may be combined to support industry use cases. We found that the AAS metamodel fulfils similar semantic interoperability requirements as RDF-based ones, however both approach the interoperability challenge from different angles. The AAS model is designed to provide a standardized interface to data describing assets, while RDF is a more general framework offering more flexibility compared to the AAS. As the AAS model is tailored to the manufacturing domain, it makes it easier for engineers and domain experts to develop a model that fits their needs. Furthermore, the AAS concept in general goes beyond the modelling of assets and supports easy integration with commonly used machine interfaces.

As there is already experience and existing work on RDF-based semantic models, we believe that the AAS model development can benefit from this. As a first step in this direction we propose a mapping from semantic models to elements in the AAS. Allowing generation of AAS models based on already existing RDF ontologies. This aids developers in defining AAS models with meaningful semantic identifiers that can be dereferenced to get the semantic definition and context of the AAS element. This provides a solution to question 1 as posed in the introduction.

Next to the alignment of semantic RDF-based models with AAS models, we found that combining both concepts in a single solution design can be of added value in modular production environments. In the MAS4AI project an agent-based digital twin is developed, which is used to control a (simulated) modular production environment. For this system to work, complex questions have to be answered regarding the environment in which the agents operate. The current AAS framework does not efficiently support this. Introducing a semantically interoperable RDF store provides the needed capabilities and flexibility to answer these more complex queries. As demonstrated in a number of examples, this approach allows us the solve the problem posed in question 2 in the introduction.

6.1. Limitations

The current implementation which we provided in Section 5 provides a proof-of-concept of the combination of AAS and RDF metamodels. In this work we focused on explaining the approaches and their applications, therefore the evaluation of the approaches and their implementations in terms of performance and scalability is limited. However, the first results show that the approaches are promising and a more extensive evaluation in an experimental and practical setup would be a next step.

The RDF to AAS mapping in Section 5.1 only covers relatively simple data models, and for example not the more complex constraints that can be expressed using SHACL and even more using OWL2. This limitation is mainly due to the limited expressiveness of the AAS metamodel, which does not support complex value constraints. However, as the AAS metamodel is still in development it might be possible to map more complex ontologies and constraints to similar constructs in the AAS model. An alternative is to only provide the semantic identifier that can be dereferenced to discover the definition and constraints for the class or property at hand. As we argued before, it would be of added value if AAS tooling would support dereferencing the semantic identifiers, and optionally also automatic validation of the constraints defined in the referenced semantic model.

The main limitation of the second approach, using an RDF graph as a search index for AASs, is that it depends on properly designed models that align with each other. In most cases this requires sufficient knowledge of RDF and AAS modelling, which might not be present in every organization. However, note that the concept proposed in Section 4.1 can help to align an AAS model with an RDF-based model. Additionally, the current framework and implementation do not support push-based and targeted updates of AAS elements and therefore there is no guarantee that changes in the AASs are directly synchronized to the RDF store.

6.2. Future work

Both described approaches will be tested in a number of real-life use cases in different manufacturing settings. More specifically, the method presented in Section 4.2, combining the RDF and AAS framework, will be applied in an agent-controlled simulated production environment as part of the MAS4AI project. This provides an opportunity to evaluate how the solutions behave in a scaled up test case with assets representing a manufacturing environment. In this work we present the approaches and demonstrate their functionality in a small proof-of-concept, a more extensive experimental evaluation of both approaches is required to validate and quantify the applicability. In the experiments the data size should be considered, as well as the complexity of the models and data set. Specifically for the conversion from RDF-based to AAS models the performance and scalability using different techniques can be evaluated. The current version implements all rules in SPARQL, but another approach could be to use SWRL rules in combination with a reasoner.

Independent from testing the implementations presented in Section 5, we would also like to extend the described methods and increase their usability in different settings. For example, an extension of the RDF to AAS mapping to support more complex graph patterns and mapping from OWL-based models. Another improvement that should be investigated is a more efficient synchronization mechanism to keep the RDF store up to date with the data exposed by AAS servers. One solution to consider is a push-based and targeted update of data in the RDF store when an AAS element is updated. The current AAS specification does not support such a mechanism yet, but the BaSyx eventing extension[14] can be used for prototyping.

Beyond the scope of the currently suggested solutions, we propose two research directions for future work. Firstly, one may define the standard interfaces exposed by AAS servers on an RDF store, allowing interaction with an RDF graph as if it's an AAS and providing the ease of integration of the AAS, without limiting the expressiveness and capabilities of an RDF based knowledge graph. Secondly, we recommend the development of AAS modelling tools which dereference URIs, which should make it easier to reuse predefined concepts in the creation of AAS models and stimulate the use of proper semantic identifiers.

Finally, we identified the need to take this research and explore further how it supports ongoing standardization processes. Specifically, we are seeing the AAS metamodel being extended and surrounding tools being developed to make it more powerful as a modelling language. For example, the work by Bayha et al (Bayha et al., 2020). provides an approach on how to model capabilities in the AAS, which fits well with the use case we envision for the RDF store solution described in Section 4.2. More research should be conducted on how these metamodels can converge and what this means for implementations and usage.

[14] https://wiki.eclipse.org/BaSyx_/_Developer_/_Extensions_/_Eventing

7. Conclusion

In this paper we formulated two challenges and questions on the integration of semantic technologies and the Industry 4.0 Asset Administration Shell. Subsequently, we propose and explore two novel approaches that addresses those questions. The first challenge is how the Industry 4.0 AAS metamodel can be combined with semantic RDF-based metamodels. We demonstrate the conversion from RDF-based models to AAS models, such that the structure of the original model is maintained and the AAS model is annotated with references to the original model using the semantic identifiers that are part of the AAS metamodel. This reduces the modelling effort when developing a new AAS model for an asset and avoids duplicating information, as the AAS model can refer to information and knowledge represented in an RDF graph. The second challenge is the semantic discovery of assets. We present an approach, which demonstrates how RDF and AAS can be used next to each other. Specifically, we showed how a solution combining both modelling paradigms and their frameworks provides more (semantic) capabilities compared to using the frameworks in isolation. The RDF store can be used to execute more advanced queries that touch multiple AASs and possibly also an (RDF-based) information model.

We propose both approaches connected to our work in the MAS4AI project and suggest implementations based on the pilots in this project. Initial results suggest that these methods meet the requirements and are usable by developers of agents in the MAS4AI framework, while providing the additional functionality required by the pilot lines. Therefore, we think the proposed solutions can also be of interest in other settings, so in this work we generalize them. However, we recognize that a broader evaluation of the approaches is necessary, so we hope this work inspires others to apply and validate the approaches in their (industrial) setting and experiment with the generation of AAS models based on existing RDF models. Next to that, we intend to continue the development and validation efforts with real-life manufacturing use cases in the MAS4AI and possible follow-up projects.

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.

Data Availability

Data will be made available on request.

Acknowledgements

This project is supported by the European Union‘s Horizon 2020 research and innovation program under grant agreement No. 957204, the project MAS4AI (Multi-Agent Systems for Pervasive Artificial Intelligence for assisting Humans in Modular Production). We specifically would like to thank our colleagues working in the MAS4AI project for their support by providing insights in their use cases and application of the AAS.

References

Bader, S., Maleshkova, M., 2019. The semantic asset administration shell. In: International Conference on Semantic Systems. Springer, Cham, pp. 159–174. https://doi.org/10.1007/978-3-030-33220-4_12.

S. Bader, B. Berres, B. Boss, A. Gatterburg, M. Hoffmeister, Y. Kogan, A. Köpke, M. Lieske, T. Miny, J. Neidig, A. Orzelski, S. Pollmeier, M. Sauer, D. Schel, T. Schröder, M. Thron, T. Usländer, J. Vialkowitsch, F. Vollmar, S. Madanska, Details of the Asset

Administration Shell. Part 2 - Interoperability at Runtime - Exchanging Information via Application Programming Interfaces (Version 1.0RC02), 2021. URL: https://www.plattform-i40.de/IP/Redaktion/EN/ Downloads/Publikation/Details_of_the_Asset_Administration_ Shell_Part2_V1.html.

S. Bader, E. Barnstedt, H. Bedenbender, B. Berres, M. Billmann,B. Boss, N. Braunisch, A. Braunmandl, E. Clauer, C. Diedrich,B. Flubacher, W. Fritsche, K. Garrels, A. Gatterburg, M. Hankel, S. Heppner, M. Hoffmeister, L. Janicke, M. Jochem, C. Ziesche, 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), 2022. URL: https://www.plattform-i40.de/IP/Redaktion/EN/ Downloads/ Publikation/Details_of_the_Asset_Administration_ Shell_Part1_V3.html.

A. Bayha, J. Bock, B. Boss, C. Diedrich, S. Malakuti, Describing Capabilities of Industrie 4.0 Components, Technical Report, Platform Industrie 4.0, 2020. URL: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Capabilities Industrie40_Components.pdf?_blob=publicationFile&v=2.

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

F. Belleau, M.-A. Nolin, N. Tourigny, P. Rigault, J. Morissette, Bio2RDF: Towards a mashup to build bioinformatics knowledge systems, Journal of Biomedical Informatics 41 (2008) 706–716. URL: https://www.sciencedirect.com/science/article/pii/S1532046408000415. doi:https://doi.org/10.1016/j.jbi.2008.03. 004, Semantic Mashup of Biomedical Data.

A. Belyaev, C. Block, B. Boss, C. Diedrich, P. Juhel, W. Hartmann, O. Hillermeier, N. Ondracek, S. Pfeifer, F. Scherenschlich, J. Schmelter, Modelling the Semantics of Data of an Asset Administration Shell with Elements of ECLASS, Technical Report, ECLASS, 2021.URL: https://eclass.eu/fileadmin/Redaktion/pdf-Dateien/ Broschueren/2021–06-29_Whitepaper_PlattformI40-ECLASS.pdf.

Bischof, S., Schenner, G., Steyskal, S., Taupe, R., 2018. Integrating Semantic Web Technologies and ASP for Product Configuration (September). ConfWS 53–60.

Bouter, C., Pourjafarian, M., Simar, L., Wilterdink, R., 2021. Towards a comprehensive methodology for modelling submodels in the industry 4.0 asset administration shell. 2021 IEEE 23rd Conference on Business Informatics (CBI). IEEE, pp. 10–19, 10.1109/CBI52690.2021. 10050.

C. Bouter, R. Wilterdink, R. Hindriks, C. Leeuwen, Representing the virtual: Using aas to expose digital assets, in: Third International Workshop On Semantic Digital Twins (SeDiT 2022), 2022. URL: https://ceur-ws.org/Vol-3291/.

N. Braunisch, M. Ristin-Kaufmann, R. Lehmann, H.W. van de Venn, Generative and model-driven sdk development for the industrie 4.0 digital twin, in: 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE Press, 2021, p. 1–4. URL: https://doi.org/10.1109/ETFA45728.2021.9613164. doi:10.1109/ETFA45728.2021.9613164.

I. Grangel-Gonzalez, L. Halilaj, G. Coskun, S. Auer, D. Collarana, M. Hofmeister, Towards a semantic administrative shell for industry 4.0 components, in: 2016 IEEE Tenth International Conference on Semantic Computing (ICSC), IEEE, 2016a, pp. 230–237. doi:10.1109/ICSC.2016.58.

Grangel-Gonzalez, I., Halilaj, L., Auer, S., Lohmann, S., Lange, C., Collarana, D., 2016b. An rdf-based approach for implementing industry 4.0 components with administration shells. 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE,, pp. 1–8. https://doi.org/ 10.1109/ETFA.2016.7733503.

Hildebrand, M., Tourkogiorgis, I., Psarommatis, F., Arena, D., Kiritsis, D., 2019. A Method for Converting Current Data to RDF in the Era of Industry 4.0. Springer,, Cham, pp. 307–314 doi:10.1007/ 978-3-030-30000-5_39.

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M., 2004. SWRL: a semantic web rule language combining OWL and RuleML, technical report. W3C. URL: https://www.w3. org/Submission/SWRL/.

Industrial Digital Twin Association, Aas specs, https://github.com/admin-shell-io/aasspecs/tree/master/schemas/rdf, 2022.

Kamburjan, E., Klungre, V.N., Schlatte, R., Tarifa, S.L.T., Cameron, D., Johnsen, E.B., 2022. Digital twin reconfiguration using asset models. In: Leveraging Applications of Formal Methods, Verification and Validation. Practice: 11th International Symposium, ISoLA 2022, Rhodes, Greece, October 22–30, 2022, Proceedings, Part IV. Springer-Verlag, Berlin, Heidelberg, pp. 71–88. URL: https://doi.org/10.1007/ 978-3-031-19762-8_6. doi:10.1007/978-3-031-19762-8_6.

LOD cloud group, The linked open data cloud, 2020. URL: https://lod-cloud.net/.

MAS4AI Consortium, Multi-agent systems for pervasive artificial intelligence for assisting humans in modular production, 2020. URL: https://www.mas4ai.eu/.

G. v Nederveen, R. Beheshti, P. Willems, Building information modelling in the netherlands: a status report, in: W078-Special Track 18th CIB World Building Congress, volume 361, Salford, United Kingdom, 2010, pp. 28–40. URL: https://www.irbnet.de/daten/iconda/ CIB18802.pdf.

F. Ocker, B. Vogel-Heuser, H. Schön, R. Mieth, Leveraging digital twins for compatibility checks in production systems engineering, in: 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2021, pp. 103–107. doi:10.1109/IEEM50564.2021.9672892.

OWL Working Group, OWL 2 Web Ontology Language Document Overview, Technical Report, W3C, 2012. URL: https://www.w3.org/ TR/owl2-overview/.

P. O. of European Union, Eu open data portal sparql endpoint, 2017. URL: https://data.europa.eu/data/datasets/ eu-open-data-portal-sparql-endpoint?locale=en.

H.K. Pakala, C. Diedrich, K. Oladipupo, S. Käbisch, Integration of asset administration shell and web of things, 2021. doi:10.25673/39570, online.

RDF Data Schapes Working Group, Shapes Constraint Language (SHACL), Technical Report, W3C, 2017. URL: https://www.w3.org/ TR/shacl/.

RDF Working Group, RDF 1.1 Primer, Technical Report, W3C, 2014. URL: https://www.w3.org/TR/rdf11-primer/.

Schröder, M., Schulze, M., Jilek, C., Dengel, A., 2021. Bridging the technology gap between industry and semantic web: Generating databases and server code from rdf. In: Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART). SCITEPRESS, pp. 507–514. https://doi.org/10.5220/ 0010186005070514.

SPARQL Working Group, SPARQL 1.1 Overview, Technical Report, W3C, 2013. URL: https://www.w3.org/TR/2013/ REC-sparql11-overview-20130321/.

Technical Committee: ISO/TC 59/SC 13 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM), Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries — Part 1: Data schema. International Organization for Standardization, Standard, International Organization for Standardization, Geneva, CH, 2018. URL: https://www.iso.org/standard/ 70303. html.

Technical Committee: ISO/TC 59/SC 13 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM), Information container for linked document delivery — Exchange specification — Part 1: Container, Standard, International Organization for Standardization, Geneva, CH, 2020. URL: https://www.iso.org/standard/74389. html.

A. Textor, S. Stadtmüller, B. Boss, J. Kristan, BAMM Aspect Meta Model, in: International Semantic Web Conference (ISWC) 2021: Posters, Demos, and Industry Tracks, 2021. URL: https://ceur-ws. org/Vol-2980/paper415.pdf.

Wikidata gorup, Wikidata, 2019. URL: https://www.wikidata.org/.

Wilkinson, M., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., Bonino da Silva Santos, L.O., Bourne, P., Bouwman, J., Brookes, A., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C., Finkers, R., Mons, B., 2016. The FAIR guiding principles for scientific data management and stewardship. Sci. Data 3. https://doi.org/10.1038/sdata.2016.18.