A Participatory Modelling Approach to Agents in Industry Using AAS

Nikoletta Nikolova, Cornelis Bouter, Michael van Bekkum, Sjoerd Rongen, and Robert Wilterdink

1 Introduction

With the increasing number of assets being digitized, all of which are expected to be interoperable with each other, the need for well-designed information models grows. Such models describe a given entity in a machine-readable way and enable interpreting, usage and reasoning with data from previously unknown devices. The usual approach to make these information models is through a heavy standardization process in which a large number of organizations all have to come to a common understanding of definitions and approaches, after which all parties implement this understanding in their systems [17]. These standards are usually well-scoped and documented, making them easier to use. For developers, this provides a steady base to build upon without fear of the implementation becoming outdated quickly due to a newly released version of a standard. Unfortunately, these standardization

Cornelis Bouter, Michael van Bekkum, and Sjoerd Rongen contributed equally to this work.

N. Nikolova · C. Bouter · M. van Bekkum (✉)
Data Science, Netherlands Organisation for Applied Scientific Research (TNO), Den Haag, The Netherlands
e-mail: nikoletta.nikolova@tno.nl; cornelis.bouter@tno.nl; michael.vanbekkum@tno.nl; sjoerd.rongen@tno.nl; robert.wilterdink@tno.nl

S. Rongen
Data Ecosystems, Netherlands Organisation for Applied Scientific Research (TNO), Den Haag, The Netherlands

R. Wilterdink
Advanced Computing Engineering, Netherlands Organisation for Applied Scientific Research (TNO), Den Haag, The Netherlands

J. Soldatos (ed.), *Artificial Intelligence in Manufacturing*,
https://doi.org/10.1007/978-3-031-46452-2_12

efforts tend to be slow, often taking multiple years before agreement is reached, for example, ISO standards take 18 to 36 months to be developed [8].

Because of their slow development nature, formal standardization processes are not universally suitable for all applications. For example, when developing a new product, it is not feasible to delay the product launch by months, or even years, because an agreement needs to be reached on the information model. As such, currently, during development, there is little alignment done. Instead, the product developer creates a model with the information that they consider necessary. This is similar to what has been happening in the development of linked data ontologies, where there are few formal standards standardized by W3C (RDF, OWL, SHACL) defining the metamodel. A large number of common use case-specific models made by institutions unrelated to W3C have become *de facto* standards, such as Friend of a Friend,[1] Prov-o,[2] Schema.org.[3] These *de facto* standards are commonly used by individuals designing their models and enable easier alignment between a large number of use case-specific models. This results in a large number of datasets that mostly use domain-specific ontologies, while they can still be related to each other as done with the Linked Open Data Cloud.[4]

We believe these *de facto* standards are vital to quickly defining the semantic model. This belief appears to be shared with organizations in the manufacturing industry, as they have developed their own metamodel to ensure domain models can be aligned with each other. The standardized Asset Administration Shell (AAS) defines how to structure the information model of your asset without specifying what your asset is or what its information model is, cf. [2]. This provides a structure for various organizations to express interoperable models, which may become *de facto*o standards. However, in the Asset Administration Shell community, we detect a lack of tools and methodologies that allow these bottom-up developed AAS submodels to rise to the level of *de facto* standards. This leaves us with formal standards (such as those published by the Industrial Digital Twin Association (IDTA) [7]) and lots of use case-specific models which lack the reuse of domain-relevant models which aids their interoperability.

In this paper, we build upon earlier work for AAS modelling practices [3] and agent modelling approaches [10] to present a set of tools, aiming to help interoperability and standardization. We build on lessons learned from the MAS4AI[5] European project in which software agents were described using an AAS utilizing a reusable model of describing software agents. The methodology referred to in this paper was developed within the European DIMOFAC[6] project. In this work, we aim

[1] http://xmlns.com/foaf/0.1/.

[2] https://www.w3.org/TR/prov-o/.

[3] https://schema.org/.

[4] https://lod-cloud.net/.

[5] https://www.mas4ai.eu.

[6] https://dimofac.eu/.

to aid in developing better bottom-up models of sufficient quality to become *de facto* standards and can then be adopted. For this, we present tooling and a methodology.

The rest of the paper starts with an overview of the relevant background and previous research. Afterward, we explain in three consecutive sections the model, methodology, and software tools we developed. Lastly, we provide an analysis of how our work can be used, what its limitations are, and what the next steps are.

2 Background

To achieve interoperability between collaborating parties, an agreement on semantics is needed. An information standard is required to effectively connect systems, organizations, and work. Commonly this is done via formal standardization processes such as those facilitated by ISO, governments, or large sector organizations. It can also be done on a meta-level with a higher level of abstraction. An example is the usage of RDF [15], OWL [12], and SHACL [14] as a standardized way to express domain language in a machine-readable format. In these cases, the metamodel has gone through a strict extensive standardization procedure, but the domain models based on it can be freely published by their authors.

Industry 4.0 introduced the meta-language AAS [13]. It is a semantic model, which provides a structure to describe assets in a machine-readable way including a standardized interaction interface. The standard is young, resulting in a lack of an established community, which actively implements it, or a uniform process for designing, defining, or implementing an AAS. To enable such a task, it is important to consider three main elements: (1) what the models represent; (2) how can they be created; (3) how can they be standardized.

When cogitating on what information models represent, we consider not only physical but also digital assets. Particularly, as agents are becoming more widely adopted in industrial applications, it is imperative to have an AAS modelling strategy. Recent works, such as the one by [11] propose a way to model two types of agents—product and resource agents. Our previous work [10] provides a more general solution to this by presenting a use case-agnostic general agent model structure, which we further extend in this paper.

When developing AAS models, the current approach is focused on using the AAS library provided by the IDTA [7], as it contains the set of the currently standardized AAS models available. As such, this is the first place many AAS users will check when looking for a model. However, the available library is not a complete overview of all AAS submodels that have been made and introducing a new submodel to it is time-consuming. Although the IDTA's contribution to standardization is indispensable for the development of a thriving AAS ecosystem, its approach is top-down. That is to say, when having developed a domain-specific AAS submodel it goes through multiple checks and reviews before being published [1]. While this brings the advantage of high-quality models and a steady base for developers to build their solutions, it is a slow process which is bottlenecked

by the speed at which information models can be reviewed and consensus can be reached. To be able to achieve widely adopted semantic interoperability, it is needed to facilitate also a bottom-up approach.

Currently, there is a lack of common standardization, which is universally used, as shown in the work of [9]. To tackle this, there are multiple different approaches employed toward developing a common method—[5] proposes a solution using OPC-UA, [4] defines an interoperability framework for digital twins, and [6] offer a solution using ontologies. We tackle interoperability from the perspective of what is needed for a standardization approach to happen and how can it be implemented. Specifically, we look at the topic of agents and how their AAS modelling process can be standardized. We develop a set of tools, which can be used to simplify and streamline the process of modelling with AAS. To the best of our knowledge, there is no current common process. We place all those developments in the context of real Industry 4.0 applications.

3 AAS Model for an Agent

Standardization and interoperability require a common approach toward information representation and more particularly modelling. When modelling manufacturing environments, it is important to consider not only physical but also digital assets. More specifically, one may consider these digital assets to be agents responsible for a piece of logic, as such it is essential to have a way to model such an asset. To address this, we provide a general model structure, which can be used when creating a model of an agent using the asset administration shell.

3.1 General Model Structure

The general model structure follows the work we presented in [10]. We provide a set of general and standard submodels which aim to provide a structure, which can be followed when creating any agent model. The model is shown in Fig. 1, in which we add the *Parameterization* and *Configuration* submodels to replace the *Communication* submodel from earlier work. The purpose of this change is to address the difference between the information locally needed for an agent to be defined, and the data, defined by the framework where it is deployed. This creates a concrete distinction between what type of information is needed in the submodels and where it comes from.

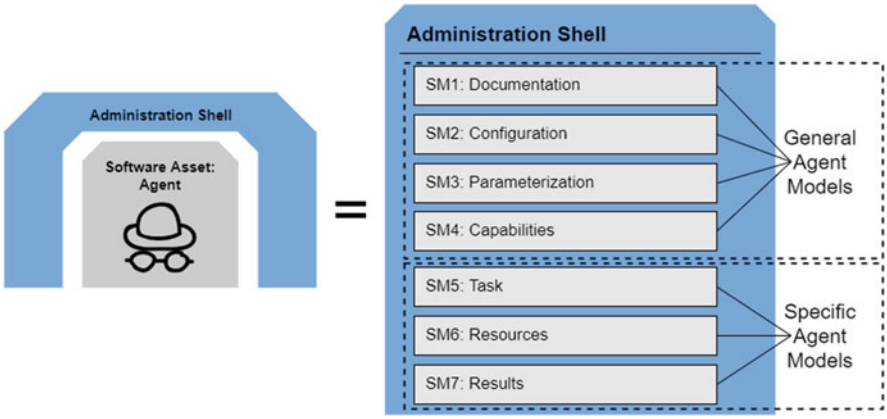

Fig. 1 General agent structure, which contains seven different distinct submodels, split into two categories

3.2 Generic Submodels

The set of generic submodels contains information, which is considered task agnostic and is needed for correct tracking and connection of the developed agent. The definitions follow from the work of [10].

- ***Documentation:*** The Documentation submodel contains information, which is relevant for describing any background and specifications for the agent. Example properties for this would be details about the developer, version, required software, language, algorithms, etc.
- ***Configuration:*** The Configuration submodel contains information, which is needed/provided by the framework or agent management system, where the agent will live. It provides information such as where the proper interface connections can be found, how to configure, etc. This is information which is determined by the framework and is usually shared between multiple agents.
- ***Parameterization:*** The Parameterization submodel contains information regarding the exact parameters that an agent needs to be able to initialize. Those parameters are determined by the exact algorithm and construction of the agent and are specific for a particular agent type.
- ***Capabilities:*** The Capabilities submodel contains information about what the agent can do. This can be done in various ways, including in combination with RDF [16].

3.3 Specific Submodels

Specific Submodels [10] are those, which depend on the use case. They contain information, which is determined by the exact situation and setting where the agent is to execute its work.

- ***Task:*** The task submodel provides a description of the exact task that the agent has to execute. Depending on the case there can be a single task such as *"Moving,"* but it can also be a set of multiple sub-tasks.
- ***Resources:*** The resources submodel aims to wrap in one place the connections to all resources and corresponding relevant information from them, which the agent would need. This would, for example, contain properties such as *"Machine Capability"* and *"Operator Availability."*
- ***Results:*** The results submodel presents the type of results and corresponding details that the agent provides after its task is executed. There can be multiple results such as *"Analysis," "Plan,"* etc.

3.4 Usage

The concept of the general agent structure is to serve as a base skeleton model, which provides a clear split and indication regarding what type of information needs to be contained in an agent model. The aim is to use the structure, when creating models of new agents, starting from concretely specifying the submodels to filling them in. To provide an approach for this, we have defined a methodology based on the work of [3], described in the next section.

4 Methodology for Developing an AAS

When creating a rich semantic information model there are several aspects to consider. To support proper modelling practices, we suggest the usage of a well-defined process which ensures no steps are missed.

We propose a methodology based on earlier work by [3], which we extend to make it more applicable when modelling not only physical assets but also software assets. The methodology is additionally extended by identifying four phases, described below and visually represented in Fig. 2.

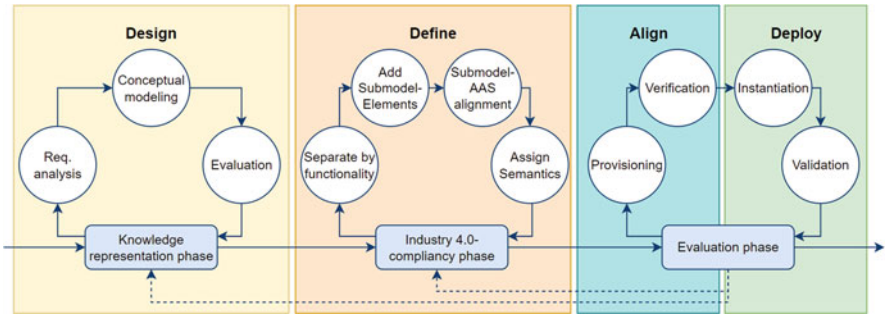

Fig. 2 Updated AAS Development Methodology, visualizing the four separate steps—design, define, align, deploy

4.1 Phases

The methodology is separated into four phases covering the implementation of a set of AAS models. The methodology starts with the Design phase which covers the knowledge representation necessary to describe the factory concepts underlying this problem. This phase requires no special knowledge about the AAS or other Industry 4.0 technologies. It is an application-independent phase of modelling the relevant factory assets and their properties. The domain expert is safeguarded from having to consider Industry 4.0 requirements. A further specification of the consideration of the four sub-steps is described in [3].

The methodology concludes with an evaluation phase separated into two parts: (1) the alignment of the existing data models being used in the factory with the developed AAS models and (2) the deployment of the tooling for which the AAS interoperability was established. The former is called verification since it is a process we can formally check: if the factory data elements have been aligned with the AAS data elements, both are aligned. The deployment is the validation step because it cannot be formally verified. We consider the AAS model validated when the tool the user had in mind at the start of the process can be built using the AAS models.

4.2 Agent Modelling

In the case of agent modelling, the standard process described above is directly applicable thanks to the two added states—provisioning and instantiation. Furthermore, the process can be split into four distinct stages, referring to the different implementation stages of agent development.

1. ***Design:*** The design stage is the use case specification moment. This is, for example, the point in time when we aim to identify the types of agents which

would be relevant/useful to the particular situation, such as *product agent*, *resource agent*, *planning agent*, etc.

2. ***Define:*** The define stage is the template generation phase. At this moment the developers can create/choose standard submodels, which can be used to provide a skeleton structure for the developed agents. More particularly, the aim is to fill the general agent structure, which was described in Fig. 1. For the General Agent Submodels, this would mean identifying the template structure to be used, whereas, for the Specific Agent Submodels, it would require identifying their exact definition and expected content.
3. ***Align:*** The align stage focuses on filling in the submodels in the created templates with any needed properties, mainly details such as exact parameters needed for the agent to operate properly. At the end of this stage, it is expected that the Agent templates contain all the relevant information.
4. ***Deploy:*** Lastly, the deployment stage takes care of the spawning of the agent. That includes filling all properties (such as Agent ID, Task ID, etc.) through the framework or agent management system, which is used. Since multiple entities of a single type of agent can be active, this is the phase where they are created and the corresponding models are filled.

5 AAS Model Repository

One of the key components of interoperability is enabling seamless sharing and distribution of developments. In the case of AAS models, there is currently no official software, which provides a simple and user-friendly interface for visualization and sharing. This is important since the correct handling of this process can enable collaboration between different parties and support the smooth distribution of work. Therefore, to close the loop, we developed an online public repository (https://admin-shell-library.eu/). It focuses on providing a way to share models between parties and visualize them in a user-friendly manner.

5.1 Functionality

The main functionalities of the repository are (1) visualization and (2) distribution, both focusing on making the AAS models' development and cooperation easier. Currently, the majority of development happens behind closed doors in silos, with distribution and information sharing only happening at the last step of the process. It is very important to enable ways for cooperation, especially since the increased interest in the AASs also creates accidental duplication of work.

One of the main challenges for distribution is the lack of direct sharing possibilities when it comes to AAS models. Presently, the standard process requires downloading from a source (such as GitHub) and running an extra program (such as AAS package explorer) to open and view a model. This can hinder the development

process, since more steps are required and hence more time and focus are needed to review a model. What the AAS repository enables is direct link sharing. Once a developer uploads their model online, it is possible to get a link, which leads to a web page visualization of the AAS. The link can be shared with other parties and removes the need for any software installation, which can significantly simplify the sharing and collaboration process, which are key for creating general and reusable models.

In general, a full AAS contains multiple Submodels, which each contain several (nested) `SubmodelElements`. For each of these Submodels and `SubmodelElements`, `semanticIds`, descriptions and Concept Descriptions should be maintained. Moreover, multi-language `SubmodelElements` exists, which can have several values attached for the various supported languages. Because of this size and complexity, a model can have, working collaboratively on an AAS template should be supported with proper visualization. Currently, the most commonly used tool is also the one used for model creation—the package explorer. While this is useful software, it can be unintuitive for non-accustomed users and increase complexity when looking at a model (especially for external parties, who do not work with this program). The repository provides a web interface, which removes the need for separate software tooling for viewers.

5.2 Working Principle

The working principle of the repository is visualized in Fig. 3. The figure visualizes the different interactions that a user can have with the repository and the high-level corresponding processing steps.

- ***Upload**:* A user can upload a model to the repository by filling in the metadata and providing the corresponding `.aasx` file. This model will be checked for errors by the system and if no errors are found, it would be uploaded to the repository.
- ***Modify**:* A user can modify the metadata of a model or upload a newer `.aasx` version at any point.
- ***Visualize**:* The model visualizes multiple details of the `.aasx` model. It provides collapsible submodels, each containing the corresponding elements. There is a possibility to show concept description and most importantly example data. This makes it easier for a viewer to understand the whole process of the models. If example data contains references to other models included in the repository, they can easily be followed via Web links. Especially for complex composite models, this can make a significant difference.
- ***Share**:* Any of the models can be shared via a link, which directly links to the relevant AAS file.
- ***Access**:* Model access is managed by a combination of groups/users and roles. Each model can have several groups or individual users assigned. Additionally,

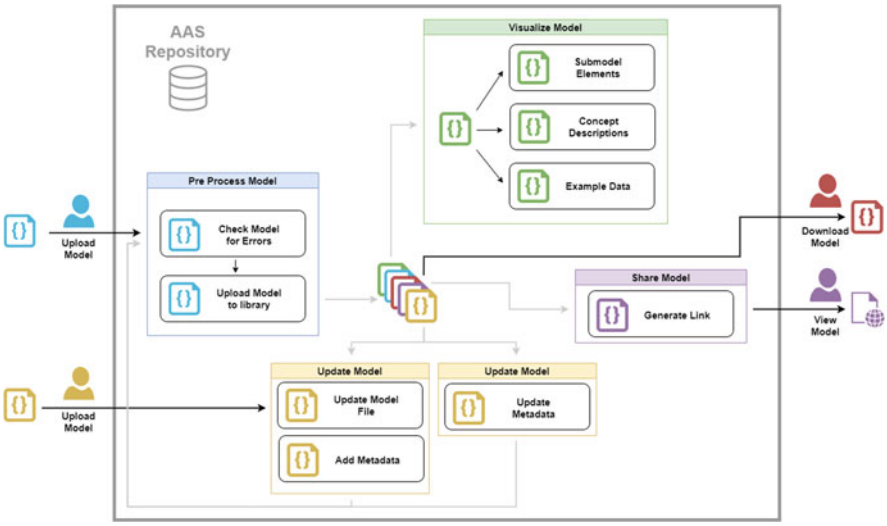

Fig. 3 Repository working principle, where black arrows represent interaction with the user and grey represent internal connections

for each user or group a role must be specified, i.e., guest, member, or editor. The combination of group/user and role then determines the effective access level. This ensures that during each step in the development process, the models can be shared with appropriate access levels and no sensitive information is publicly available.

6 Discussion

In this paper, we presented three ways of improving the creation of bottom-up standards which may rise to the level of *de facto* standards. Firstly, we presented the notion of a generic asset structure, such as for software agents, consisting of multiple submodels. Secondly, we presented a methodology on how to create your AAS models, and finally, we propose a repository for these templates which may in the findability and adoption of good AAS submodels.

6.1 Maturity of the AAS

Although we have presented a number of ways to improve the usage of the AAS and to ensure semantic interoperability we do not believe this will on its own lead to wide-spread adoption and an active ecosystem of AAS users. We believe

it is important to nurture an environment in which proper AAS modelling can thrive. However, before this technology blossoms, we believe there are still several improvements that may be made. This is to be expected given the immaturity of the AAS technology but that does not make them less pressing to tackle.

Models
To construct reusable and interoperable models, it is essential to provide clear templates and corresponding specifications. Currently, the AAS implementations do not allow for specifying what components are needed for a minimal implementation. This is especially important for future uses, as it would provide a way for users to know what is the least information they need to fill in for their model to be viable.

Model Constraints
To make the usage of semantic models more beneficial to business users and give implementers more guidance on what parts of the AAS model are mandatory and which are optional, it would be beneficial to add the ability to constrain the instantiation of the AAS model. A basic implementation of this would support defining cardinality constraints to the AAS model, defining which properties are mandatory, and which are optional. This could then be extended with value contents and potentially even some simple reasoning, to implement basic business logic in the model. For example, the intended use of a creation date and maintenance date of an asset may make it impossible to maintain the asset before it was created. However, the current AAS standard does not allow for a way to express such constraints. In semantic modelling, we have been in a similar situation not that long ago with the adoption of the Web Ontology Language (OWL), a powerful modelling language which did not get adopted as quickly in business as it could have been due to a lack of easily implemented value constraints. The creation of the Shape Constraint Language (Shacl) largely solved this problem and has increased the adoption of RDF in business contexts.

Standardization
The methodology and the repository synergetically combine to improve the bottom-up standardization of the AAS. The methodology facilitates the identification of additional submodels that are lacking among what is currently available, as well as aiding the user in following the AAS paradigm. The repository facilitates the sharing of work-in-progress submodels with other users, such that the community can swiftly adopt new work. This approach complements the top-down approach of a comprehensive IDTA standardization procedure.

In top-down standardization, the authority of the standardization body ensures the quality of the approved models. When adopting a free-for-all approach of submitting submodels to the repository, the quality should be ensured differently. Firstly, the methodology functions as a way to increase the shared knowledge of the modelling paradigm. Secondly, statistics about the usage of the various models can indicate which models turn into *de facto* standards, thereby giving a measure of their quality. Additional research and implementation work may be needed to tailor the repository for this purpose.

The increased accessibility of the repository can aid the various groups who can be expected to share their models. The methodology is primarily intended for user integrators leveraging the AAS for improved interoperability to support a factory application. Another audience that may use the repository is the machine manufacturer who wants to share instantiated and specialized templates for their machines. These submodels similarly benefit from a bottom-up approach without a mandatory procedure.

Collaborative Modelling
Currently, the process of making an AAS model is primarily a solitary effort in which the modeller still needs to actively try to reach out for input. However, the whole point of standards is that they align between different parties. As such, adding more tools to facilitate collaborative modelling would be beneficial for the creation of a proper AAS. The presented repository and development methodology already aid in this, but we believe further steps could be taken to support a shared model discussion and increase iteration speed during the development process.

7 Conclusion

The methodology and the repository described in this paper provide us with means to develop AAS models as *de facto* standards for non-physical assets from bottom-up, community-based efforts. The MAS4AI project has shown that the methodology can easily accommodate particular use case requirements by allowing for straightforward bottom-up extensions to the models. An industry use case in MAS4AI on implementing planning agent software based on ISA-95 models has shown that bottom-up standardization is instrumental. The required models were created by applying the aforementioned methodology in a collaborative effort by modelling experts, domain experts, and software developers and have led to successful integration and deployment in the use case.

The use of the repository has similarly led to clear benefits in both the MAS4AI and the DIMOFAC project: sharing the AAS model templates with all stakeholders in an easy-to-use, intuitive way has made them more accessible to all parties and has promoted discussion and feedback on their contents, thus ensuring more widespread support for the models.

Acknowledgments The research leading to these results has been funded by the European Commission in the H2020 research and innovation program, under Grant agreement no. 957204, the “MAS4AI” project (Multi-Agent Systems for Pervasive Artificial Intelligence for Assisting Humans in Modular Production Environments) and under grant agreement No. 870092, the DIMOFAC project (Digital & Intelligent MOdular FACtories).

References

1. 4.0 PI: Structure of the asset administration shell. continuation of the development of the reference model for the Industrie 4.0 component (2016) . Tech. rep. https://industrialdigitaltwin.org/wp-content/uploads/2021/09/01_structure_of_the_administration_shell_en_2016.pdf
2. Bader, S., Barnstedt, E., Bedenbender, H., et al.: 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)
3. Bouter, C., Pourjafarian, M., Simar, L., et al.: Towards a comprehensive methodology for modelling submodels in the industry 4.0 asset administration shell. In: 2021 IEEE 23rd Conference on Business Informatics (CBI), vol. 02, pp. 10–19 (2021)
4. Budiardjo, A., Migliori, D.: Digital twin system interoperability framework. Tech. rep. Digital Twin Consortium, East Lansing, Michigan (2021). https://www.digitaltwinconsortium.org/pdf/Digital-Twin-System-Interoperability-Framework-12072021.pdf
5. Cavalieri, S.: A proposal to improve interoperability in the industry 4.0 based on the open platform communications unified architecture standard. Computers **10**(6), 70 (2021). https://doi.org/10.3390/computers10060070, https://www.mdpi.com/2073-431X/10/6/70, number: 6 Publisher: Multidisciplinary Digital Publishing Institute
6. Huang, Y., Dhouib, S., Medinacelli, L.P., et al.: Enabling semantic interoperability of asset administration shells through an ontology-based modeling method. In: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. Association for Computing Machinery, New York, NY, USA, MODELS '22, pp. 497–502 (2022). https://doi.org/10.1145/3550356.3561606
7. IDTA: Industrial digital twin association (2023). https://industrialdigitaltwin.org/en/
8. ISO: Target date planner (2023). https://www.iso.org/files/live/sites/isoorg/files/developing_standards/resources/docs/std%20dev%20target%20date%20planner.pdf
9. Melluso, N., Grangel-González, I., Fantoni, G.: Enhancing industry 4.0 standards interoperability via knowledge graphs with natural language processing. Comput. Ind. **140**, 103676 (2022). https://doi.org/10.1016/j.compind.2022.103676, https://www.sciencedirect.com/science/article/pii/S0166361522000732
10. Nikolova, N., Rongen, S.: Modelling agents in industry 4.0 applications using asset administration shell. In: Proceedings of the 15th International Conference on Agents and Artificial Intelligence (2023). https://doi.org/DOI:10.5220/0011746100003393
11. Ocker, F., Urban, C., Vogel-Heuser, B., et al.: Leveraging the asset administration shell for agent-based production systems. IFAC-PapersOnLine **54**(1), 837–844 (2021). https://doi.org/https://doi.org/10.1016/j.ifacol.2021.08.186, https://www.sciencedirect.com/science/article/pii/S2405896321009563. 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021
12. OWL Working Group: Owl 2 web ontology language document overview (2012). Tech. rep., W3C. https://www.w3.org/TR/owl2-overview/
13. Plattform Industrie 4.0: Details of the asset administration shell from idea to implementation (2019). https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/vws-in-detail-presentation.pdf
14. RDF Data Shapes Working Group: Shapes constraint language (SHACL) (2017). Tech. rep., W3C. https://www.w3.org/TR/shacl/
15. RDF Working Group: RDF 1.1 primer (2014). Tech. rep., W3C. https://www.w3.org/TR/rdf11-primer/
16. Rongen, S., Nikolova, N., van der Pas, M.: Modelling with AAS and RDF in industry 4.0. Comput. Ind. **148**, 103910 (2023). https://doi.org/https://doi.org/10.1016/j.compind.2023.103910, https://www.sciencedirect.com/science/article/pii/S016636152300060X

17. Toussaint, M., Krima, S., Feeney, A.B., et al.: Requirement elicitation for adaptive standards development. IFAC-PapersOnLine **54**(1), 863–868 (2021). https://doi.org/https://doi.org/10.1016/j.ifacol.2021.08.101, https://www.sciencedirect.com/science/article/pii/S2405896321008508. 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021