

Modelling Agents in Industry 4.0 Applications Using Asset Administration Shell

Nikoletta Nikolova^a and Sjoerd Rongen^b

TNO, Anna van Burenplein 1, 2595 DA The Hague, Netherlands

Keywords: Agents, Industry 4.0, Asset Administration Shell, Modelling, Interoperability.


Abstract: Within Industry 4.0, more applications include multi-agent systems and integrated software agents in the newly developed solutions, as they can provide a valuable contribution to the manufacturing processes. This makes it important to create a digital representation of those virtual assets, similar to how this is done for physical ones. The Asset Administration Shell (AAS) has been designed for just this purpose - to create models of assets, containing all the relevant information. Currently, this is standard for physical assets, however, it could be of value to extend it beyond. We propose the usage of the AAS for creating information models of software agents and suggest a generic approach to apply the AAS meta-model to ensure semantic interoperability between them. For this purpose, we outline a structure and a set of specific submodels to group agent attributes, which can provide a description of all relevant information for a given task. We provide two examples of concrete agents and outline how this approach will be further validated within future use cases.


1 INTRODUCTION

In the current digital world, industry is giving a lot of attention to digitization and the creation of smart factories. The transition to Industry 4.0 focuses on the idea of digitizing physical assets and creating smart control mechanisms, which are able to automate production and assist humans in complex tasks. The use of software agents for tasks such as production planning or quality assurance is becoming more common, as this helps in achieving higher efficiency and robustness in the manufacturing process. To facilitate this growing digital ecosystem, a standard has been developed by the Industry 4.0 group, which aims to provide a universal way to describe assets. The Asset Administration Shell (AAS) (Plattform Industrie 4.0, 2019) is an industry-driven initiative to create semantic models of assets and expose standard interfaces to interact with these models. For example for a given physical machine, an AAS model can be created, holding all the relevant information about this asset, ranging from details such as manufacturer, markings, to system outputs, control nodes etc. This provides a way for all machines within a given factory to be described using the same meta-model and thus

makes coupling of the total system simpler. However, when considering smart factories, we no longer have just physical assets active in our manufacturing environment. As software agents are introduced to control and execute different tasks, it becomes essential to model, represent and reason with them as with every other asset. This may enable new software agents to automatically be used in the environment upon being deployed, dynamically determine which agent is best suited for the task at hand, and increase the robustness of the manufacturing control systems. The current developments, however, have not yet been extended to the world of non-physical assets and more specifically digital assets such as software and agents. We consider this to be important, as software solutions fulfill an integral role in Industry 4.0 by taking the available data and transforming this into meaningful insights and actions.

To tackle this challenge, we propose an approach on how the same AAS standard, which is used to model physical assets, can also be used to model digital ones, and more specifically agents. This paper is based on our work in the HORIZON MAS4AI project (MAS4AI, 2020) in combination with the earlier work done in the HORIZON DIMOFAC project (DIMOFAC, 2020). We consider software agents to be assets just as the manufacturing equipment they control. This enables a deeper level of control over

^a  <https://orcid.org/0000-0001-9141-4054>

^b  <https://orcid.org/0000-0002-4302-1050>

factories and better fits the current paradigm of decreasing boundaries between the physical and digital world.

We propose a standard structure, which can be used when creating AAS models for representing software agents. More specifically we provide an outline of the different types of information, which must be represented, and how it can be split between different categories. We consider this important, as at this time there is very limited literature available and even fewer semantic standards on how to model software applications as assets. This makes integration of this software in manufacturing environments complex and the adoption of Industry 4.0 slow. Moreover, an interpretation of the AAS standard applied to software assets is missing, which creates the risk of different initiatives using a different interpretation and through this harming standardization efforts. Therefore it is essential to establish a common understanding.

In this paper, we provide a structure towards how a software agent can be modelled using the AAS. Section 2 provides an overview of the relevant work and theory for this paper. In Section 3 we outline the general structure of the agent model, with Section 4 focusing on the general submodels and Section 5 on the particular submodels, including examples of how two different types of agents can be modelled conceptually. Lastly, in Section 6 we outline how we are going to further test and develop the proposed modelling approach.

2 BACKGROUND

Within industry the challenge of semantic interoperability has largely been tackled so far through the definition of static standards in lengthy standardization processes. This top-down approach often takes a long time to deliver implementable standards, which is at odds with the increasing speed of innovation and development in industry. To mitigate this issue, the Industry 4.0 Alliance introduced the Asset Administration Shell (AAS) (Plattform Industrie 4.0, 2019). This provides a digital representation of an asset, which it describes through the usage of multiple standardized submodels, each of which contains a set of elements in the form of properties, relations, references to other AAS (sub)models etc. The structure of an AAS is shown in Figure 1.

Within the AAS, concepts are given a Semantic Identifier, which is globally unique and may be reused across different models and instantiations of AAS models to relate to the same concepts used in different places, similar to how the URI works within

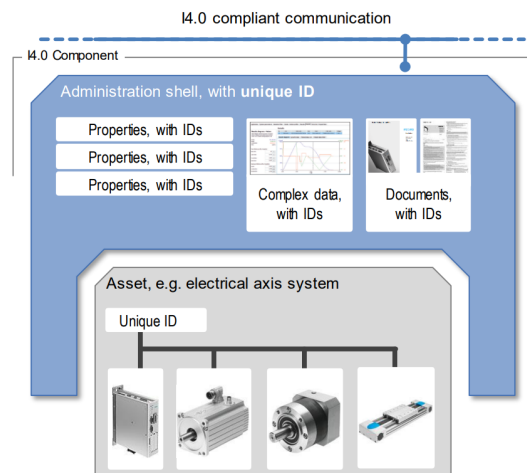


Figure 1: General structure of an AAS model.

the semantic web. Furthermore, the AAS provides standardized interfaces to interact with these semantic models (Bader et al., 2022)).

The AAS initially gained traction as a way to digitally represent physical assets such as manufacturing equipment. However, with the implementation of industry 4.0 concepts, the boundary between the physical and digital world is blurring and increasingly there is a need to integrate with digital assets in a similar way as one would integrate with physical assets. As an example, one may think of a planning algorithm or the logic controlling a machine. These pieces of software themselves are something to interact with and to a large extent responsible for the competitive position of a company through facilitating predictive maintenance or the optimal utilization of available resources. Moreover, these systems are not implemented once and then left for years or decades, as the manufacturing equipment itself may be. They are actively updated, new software is added and the amount of interactions between these systems can quickly grow. We consider these pieces of software to be agents in a multi agent system, which supports the large amount of interactions and provides a way to structure the various applications in a holonic architecture.

Using a multi agent system to support Industry 4.0 and overall the use of agents in such setting has been researched before. (Hoffmann, 2019) argues there is a need to represent the vast amount of information within a factory described in a simple way. The work presents an approach to integrate smart entities (agents) in the shop floor and corresponding model creations using OPC-UA. Similarly, (Vogel-Heuser et al., 2020) presents an overview of the different challenges within an industry 4.0 systems and argues in favour of using multi-agent systems to tackle these.

To implement agents within industry systems it is important to determine what types of agents are useful given the use case. The work of (Cruz Salazar et al., 2019) outlines patterns when it comes to the basic (referred to as *mandatory*) set of agents required within a factory - Resource Agent, Process Agent, Agent Management System and Communication Agent. The authors further argue about the importance of having such agents on the shop floor, as they execute basic functionality.

Given the strive towards interoperability enabling these agents to collaborate, it is important to define a way to model agents used within the manufacturing process. This entails a methodology to model agents and mapping the standardization approach to the one used for other assets. The work of (Sakurada et al., 2022) explores the idea of considering agents as assets when applying the AAS in Industry 4.0 applications. But, it does yet not look at how these agents would be modelled exactly.

The work of (Ocker et al., 2021) comes closest to the idea of modelling agents for a factory using the AAS. The authors provide a suggestion regarding how in particular the Planning and Resource agents can be modelled using the AAS for multi-agent systems (MAS), further pointing towards how the actual MAS can be build using those agents. Our work continues the direction Ocker et al., however we take a broader look at agent modelling. We choose to focus on how a general structure for an agent model could look, providing an outline of the different types of models and do not consider specific agent types or their definitions. Our method aims to provide a structure which can be used regardless of specifics for agents and can be applied in multiple cases. To the best of our knowledge, there is no general agent model structure at the time of this work. Therefore we investigate and propose how this problem can be tackled within the next sections.

3 STRUCTURE OF THE AAS FOR AGENTS

When describing software agents in a generic way using the AAS as a semantic standard, there are a couple of design decisions to consider. What is the structure of the model, what information should it contain and how should this be grouped. Our model aims to provide maximum flexibility in describing agents, while also providing enough structure to support reuse, constraint by what the AAS meta model provides. The process of populating an agent is placed out of scope for now.

3.1 General Structure of Agent Model

Within an AAS model, one may define properties within reusable submodels whose values can be read through a standard API. As such, the main goal of the model is to make it clear to any user or application which properties they are interested in. As there is no direct approach to do this, we considered two different approaches, which can be offered to support this.

An intuitive approach to present information of a given software application (in our case an agent) may be to look at the model elements as agent inputs, outputs or static description of the agent itself. Within these three categories, properties may then be defined. Resulting in an input, output and description submodel which has an arbitrary amount of properties. Such an approach makes it very simple for developers to know which properties to use when interacting with the agent, but at the same time makes the model highly specific, loosing the interoperability between agents, as it is not possible to create reusable submodels for different agents. That is to say, all agents have the same submodels, but these may be completely different in their actual content.

As we strive for interoperability in the models to be created, we instead consider to group relevant properties based on their semantic meaning and intended usage. This provides us with models regarding the creation of the agent, the connectivity details, work order schedules (for a planning agent specifically), etc. This leads to a greatly increased amount of submodels, some of which may contain less information than others. At first impression this may appear more complex, but it allows reusing of well defined models across different agents, resulting in faster development, easier reuse and the ability to better match to domain specific information models.

Our suggested approach provides a possible structure for how such semantic grouping can be formed for agents. We build our suggestion based on the needs of different use cases represented in the MAS4AI project, where we build an initial set of reusable submodels, used to describe an agent. Unlike standard submodels, we did not provide a complete set of submodel elements, but instead proposed a set of groupings which can be used to structure all the relevant information and provide a base point for developers to start from.

We split the agent model structure into two groups of three types of components each, as shown in Figure 2.

An Agent has a set of Generic Agent Submodels, which contain information such as documentation, communication, capabilities. This is considered

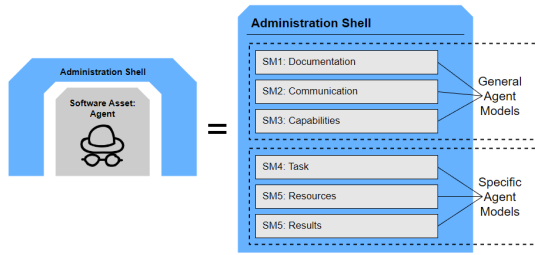


Figure 2: General structure of an agent AAS.

to be information, which should be provided with every agent and should thus contain similar type of data disregarding the exact task. The second set of Submodels is Specific Agent Submodels, which refers to the type of information that an agent needs based on the specific task it needs to execute - such as Resources, Analysis, Task etc. Each of those sets of submodels is further evaluated in the following two chapters.

3.2 Development of AAS Model

The general structure of the agent shown in the section above provides an overview regarding how the model should look and what type of information can be included. When it comes to the way the model should be filled, we refer the readers to the methodology of (Bouter et al., 2021). Their work proposes a clear and robust approach towards iterative AAS model development, as shown in Figure 3.

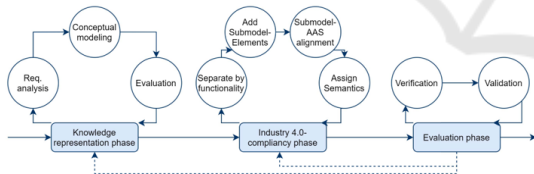


Figure 3: Overview of AAS modelling methodology (Bouter et al., 2021).

The iterative structure assures that the created model contains all the relevant information for an asset, in our case an agent, and helps provide a structure for the development process.

4 STANDARD AGENT SUBMODEL

Within the description of every agent, similar to a machine, there is a set of information that must always be represented, no matter the specific application type. When looking at the current set of provided

standard AAS submodels, developed by the International Digital Twin Association (IDTA), every created asset model uses the Nameplate and Technical Information submodels. This already encompasses all the relevant information in relation to all the specifications of the machine. When looking towards software, there is a similar set of requirements that must be provided within such a model. We consider, that there are (at least) three different submodels, which should be standard for every agent.

4.1 Documentation

Following the current modelling approach, when creating an AAS for a physical asset, an agent needs to have documentation presented in its model. The current IDTA standard submodels Nameplate and Technical Information are created with purely physical assets in mind. These contain properties which are not relevant for an agent, such as *CE Markings*. For a software component there are other relevant components, such as Software Version, Programming Language etc. This means that the current standard models are insufficient for the modelling of software agents. However, the recently released IDTA Software Nameplate (IDTA, 2022) provides a good starting point to cover most documentation needs.

4.2 Connection

An agent, whether placed within a multi-agent system or another type of architecture, contains a set of predefined connections, communication channels and protocols. These may differ slightly based on the exact implementation details and use case considered. One example of the information that such a submodel would encapsulate can be chosen following the FIPA Agent Management standard (FIPA, 1996). For example, such a submodel would contain information such as:

- *Agent Status, Agent Lifecycle*
- *Communication Channels and Protocols*

Depending on the exact situation the information within this submodel can be determined based on what the (multi-agent) system needs and requires for functions. This can also mean that there are some highly specific properties (such as trigger for activating certain agent property).

4.3 Capabilities

Every agent has a set of capabilities describing what tasks it is able to perform. We consider, when defin-

ing such submodel as standard, that it is crucial to define the exact capabilities of a given agent, as they are specific to the developed algorithm. This is not a new idea, as for example within ISA-95 every component, which falls under *Manufacturing Operations Management* has a *Capability Information*. Similar to that, Industry 4.0 also argues in favor of that within the work of (Bayha et al., 2020), as the white paper presents an approach to provide information about the offered functionality of a system by combining property descriptions and ontology. The capabilities description of an agent do not have to be limited to a single submodel, as depending on the exact situation it may be beneficial to split the information. An example for data, which would be encapsulated in such a submodel could be:

- Planning capability - what type of planning can the agent actually do, such as *deviation planning*, *re-planning*, *production planning*, *personnel planning* etc.
- Quality assessment - what type of quality check does the agent actually do, such as *measurement evaluations*, *deviation detection*, *error analysis* etc.

5 EXTENDING FOR SPECIFIC USE CASES

Besides the standard agent submodels, an agent can also have ones, which are further tailored to the actual usage and environment it operates in. Within industry applications there are numerous different use cases where agents are tasked with various assignments. We propose a further generalization of how those components can be considered and grouped based on the type of knowledge they carry.

- **Task** - every agent has a particular task that it has to execute and therefore has a related set of information that is required for it.
- **Resources** - set of information to which the agent has access to. This can be other machines, software, agent, resources etc.
- **Results / Output / Analysis** - set of information and/or analysis which the agent produces based on all the incoming data and task specifications.

These three submodels would contain the general set of information that an agent may need. The exact submodel elements can differ depending on the particular use case. To illustrate this we present how this planning method could be used to create general AAS models for two different agents within a factory.

Planning Agent. First, we consider a planning agent, which is responsible for optimizing the overall work plan to be executed by the other agents. Such an agent can have a variety of different tasks, all related to the scope of the planning assignment. For the small created example, visualized in Figure 4, we provide just a sample of the possible properties, which can be present within a given model and how those map to the structure presented above:

- **Task** - Current production plan, Set of orders.
- **Resources** - Maintenance schedule, Personnel shifts, Personnel capabilities, Machine capabilities.
- **Results/Output/Analysis** - New production planning.

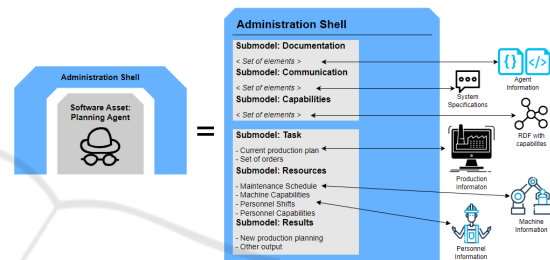


Figure 4: Structure of AAS model for a planning agent.

Quality Inspection Agent. Second, we look at a quality inspection agent, which needs to assess the level of quality of a product. The structure is visualized in Figure 5, and for it, the following information can be present:

- **Task** - Quality inspection task, Quality targets
- **Resources** - Product specifications, Machine specifications.
- **Results/Output/Analysis** - Quality analysis, Quality measurement.

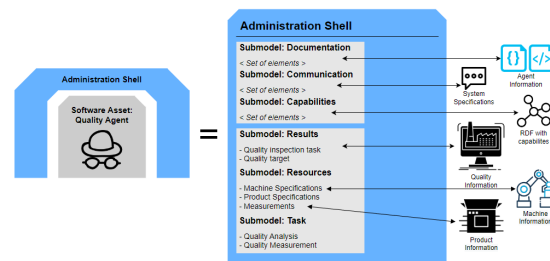


Figure 5: Structure of AAS model for a quality agent.

The provided examples¹ are in no way exhaustive of all the information such an agent would contain but

¹For access to the examples developed in the MAS4AI project, please contact the authors or go to <https://admin-shell-library.eu/>

aim to provide an idea of how the intended structure would look like for different situations. Depending on the particular use case the type of functionality for an agent can change, for example, a planning agent can have a varying set of tasks, each with its own set of requirements (the information needed for logistics planning is different from a maintenance plan).

6 OUTLOOK

In the previous sections, we presented our view on how agents should be modelled using AAS, providing a general structure that can be followed, which is not connected to a particular use case. This work is based on the HORIZON MAS4AI project, in which real-life use cases are investigated. The next step for the further development of this agent modelling methodology is to test how well it fits when applied to these use cases. Within MAS4AI we will test this with eight different general types of agents (product agent, resource agent, process orchestration agent, planning agent, safety agent, information agent, quality agent, and human-machine interaction agent) in up to six varying use cases. This provides a solid evaluation environment for whether our idea lines up with the real needs of different manufacturing entities. Our proposed approach would follow the structure shown in Figure 6.

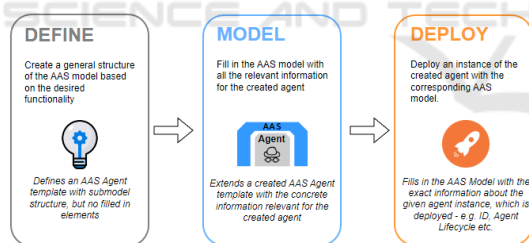


Figure 6: Steps in AAS generation.

There will be multiple different variations of the same type of agent, for example, the planning agent. This would provide a suitable way to test whether the same agent in different settings and with different requirements, would still be able to be modelled following the provided structure. Furthermore, we plan to test how this process works when the use case directly tries to implement the methodology so that we can get a clear overview of not only the usefulness of this methodology but also the understandability and logic for external parties. This is important as eventually this work is aimed towards helping manufacturing companies further implement agent modelling in their factories following the Industry 4.0 specifica-

tions. We plan to use the results from those use case pilots to determine the suitability of our approach and any potential missing elements.

Aside from that, another point we plan to further evaluate is how the ISA-95 standard can be further properly represented within the visualized model structure. In order to be able to achieve interoperability of the created models, it is important to follow a standard when creating the information definitions. Therefore, we plan to further define the methodology to align better with the standard.

Finally, by deploying these agents and connecting them to the MAS4AI framework using an AAS representation. Other applications can interact with an AAS as if it's an active entity which responds, initiates actions and adapts to its environment. This closely matches the functionality of an Active AAS as described in AAS literature, despite not fully matching the often shared conceptual diagrams putting the logic within the AAS. However, this semantic description of agents may provide a different, but valid way of implementing an active AAS.

7 CONCLUSION

In this paper we presented our position on how an agent can be modelled using the Asset Administration Shell. We propose a systematic approach outlining what the standard groups of information for an agent could be when creating a model. Our structure provides a way to represent the corresponding inputs and outputs in the same way one may model a machine with materials and products going in or out. That is to say, an Administration Shell of an agent vs that of a physical asset are not fundamentally different. We found this to be advantageous in separating complexity, as often the software development and semantic modelling are usually performed by different specialists. This approach simplifies development by considering these as two different components, instead of adding the logic to the semantics and centralizing all complexity.

The model structure that we presented provides a standard overview of how the information can be presented without specifying solution-specific components. To address how the approach can be used we also developed examples, two of which are presented here. These examples provide an overview of how a particular agent can be modelled. In the upcoming year we intend to test the approach and models for validation in the MAS4AI project's use cases. Finally, we aim to further develop standard submodels which may speed up future agent development as

well as providing a catalyst of agent submodel standardization.

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) and grand agreement No. 870092, the project DIMOFAC (Digital & Intelligent MODular FACTories). We specifically would like to thank our colleagues working on both projects for their support by providing insights into their use cases and application of the AAS.

REFERENCES

- Bader, S., Barnstedt, E., Bedenbender, H., Berres, B., Billmann, M., Boss, B., Braunsch, N., Braunmandl, A., Clauer, E., Diedrich, C., Flubacher, B., Fritsche, W., Garrels, K., Gatterburg, A., Hankel, M., Heppner, S., Hoffmeister, M., Jänicke, L., Jochem, M., and Ziesche, C. (2022). *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)*.
- Bayha, A., Bock, J., Boss, B., and Diedrich, C. (2020). Describing capabilities of industrie 4.0 components. Technical report, Industry 4.0.
- Bouter, C., Pourjafarian, M., Simar, L., and Wilterdink, R. (2021). Towards a comprehensive methodology for modelling submodels in the industry 4.0 asset administration shell. In *2021 IEEE 23rd Conference on Business Informatics (CBI)*, volume 02, pages 10–19.
- Cruz Salazar, L., Ryashentseva, D., Lüder, A., and Vogel-Heuser, B. (2019). Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns. *The International Journal of Advanced Manufacturing Technology*, 105.
- DIMOFAC (2020). Digital intelligent modular factories.
- FIPA (1996). Foundation for intelligent physical agents.
- Hoffmann, M. (2019). *Smart Agents for the Industry 4.0: Enabling Machine Learning in Industrial Production*.
- IDTA (2022). Digital nameplate.
- MAS4AI (2020). Multi-agent systems for pervasive artificial intelligence for assisting humans in modular production.
- Ocker, F., Urban, C., Vogel-Heuser, B., and Diedrich, C. (2021). Leveraging the asset administration shell for agent-based production systems. *IFAC-PapersOnLine*, 54(1):837–844. 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021.
- Plattform Industrie 4.0 (2019). Details of the asset administration shell from idea to implementation.
- Sakurada, L., Leitao, P., and la Prieta, F. D. (2022). Agent-based asset administration shell approach for digitizing industrial assets. *IFAC-PapersOnLine*, 55(2):193–198. 14th IFAC Workshop on Intelligent Manufacturing Systems IMS 2022.
- Vogel-Heuser, B., Seitz, M., Salazar, L. A. C., Gehlhoff, F., Dogan, A., and Fay, A. (2020). Multi-agent systems to enable industry 4.0. *at - Automatisierungstechnik*, 68(6):445–458.