Grant Agreement Number:	957204 (H2020-ICT-38-2020)
Project Acronym:	MAS4AI
Project Start Date:	1st October 2020
Project Full Title:	Multi-Agent Systems for Pervasive Artificial Intelligence for assisting Humans in Modular Production

D3.2 - Agent knowledge access framework

Dissemination level:	PU (Public)
Date:	March 2022
Deliverable leader:	TNO
Contributors:	DFKI, Semaku, Tecnalia, Volkswagen, Flexis, LMS
Reviewers:	DFKI
Type:	Report
WP / Task responsible:	WP3 / Task 3.2
Keywords:	Ontology, AAS, API, Agent knowledge model

Executive Summary

The purpose of this document is to provide detailed descriptions of the knowledge models for the agents, being the ontology and AASs as artefacts of the project. It also details access mechanisms (APIs) for the agents to access the knowledge models.

It concludes with the initial findings on using and implementing the AAS and the ontology in practical settings of the use cases.

Document History			
Version	**Date**	**Contributors**	**Description**
0.1	07-01-2022	TNO	Initial structure
0.5	24-03-2022	All	Draft texts on AAS
0.6	24-03-2022	TNO, Semaku	Draft texts on ontology
0.9	01-04-2022	TNO	Draft version for review
1.0	14-04-2022	TNO, DFKI	Review comments processed, final version

Table of Contents

Table of Figures

1 Introduction

Whereas deliverable D3.1 showed the design of the knowledge models, this deliverable provides an overview and documentation of the actual models that have been developed as software artefacts. Both the ontology and the AAS models for the agents have been formalized and have been made available as tangible software components. The ontology is created as an RDF model serialized in a Turtle format (.ttl), the AAS models have been created as package files (.aasx).

The development of the knowledge models follows the design principles of deliverable D3.1 on reusability and extensibility. This means that the ontology practically integrates and extends existing ontologies (MASON, RAMI) and the AAS agent models are developed as a collection of AAS submodels that can be reused across multiple agents.

In order to be useful, the knowledge models need to be accessible to the use cases, i.e., both need to be hosted in a platform that allows programmatic access via APIs. This document describes how the ontology is hosted in an RDF store in the MAS4AI architecture, whereas the AAS models are hosted in a AAS hosting platform, typically BaSyx.

The main purpose of this document is:

- To describe the schema of the ontology, its concepts and the mechanism to access the ontology via queries in a practical use case.
- Describe the mapping of the models to the standardized AAS metamodel and describe the submodels that have been created for the various agents based on the design in deliverable D3.1.

The document concludes with a reflection on the practical application of the ontology in use cases, several considerations on the continued development of the AAS models in alignment with the use cases as well as a useful solution to lack of findability of AAS (sub)models.

2 Ontology & Knowledge Base

This chapter describes the schema of the ontology, the definition of its concepts, its implementation in an RDF store and mechanisms to access the ontology.

2.1 Definition of ontology elements

This section presents the definitions of the concepts in the ontology and how they are related.

As has been laid out in deliverable D3.1, the MAS4AI ontology extends both the MASON and RAMI ontologies (Figure 1).

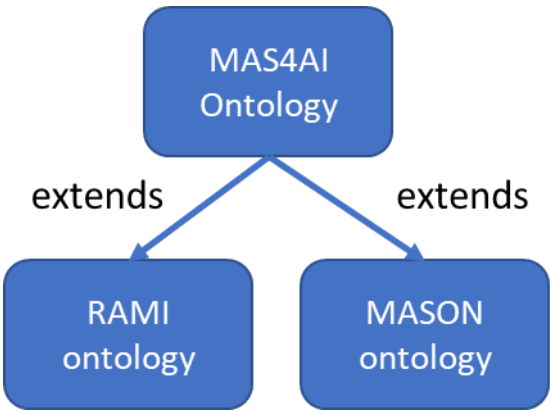

Figure 1 The MAS4AI ontology as an extension of MASON and RAMI

The concepts and properties in the MAS4AI ontology are defined in the MAS namespace at `http://mas4ai.eu/mas4ai_model`. The MAS4AI ontology has five main concepts (Table 1).

Table 1 MAS4AI Ontology concepts

MAS4AI concept	Definition
Agent	A computational process that implements the autonomous, communicating functionality of an application (from FIPA).
MAS4AI Agent	A type of Agent that has skills to perform a task and is implemented by an AAS interface.
Interface	Defined connection point of a functional unit which can be connected to other functional units.

Skill	Ability to produce solutions in some problem domain. An ability that has been acquired by training.
Task	Any piece of work that is undertaken or attempted.

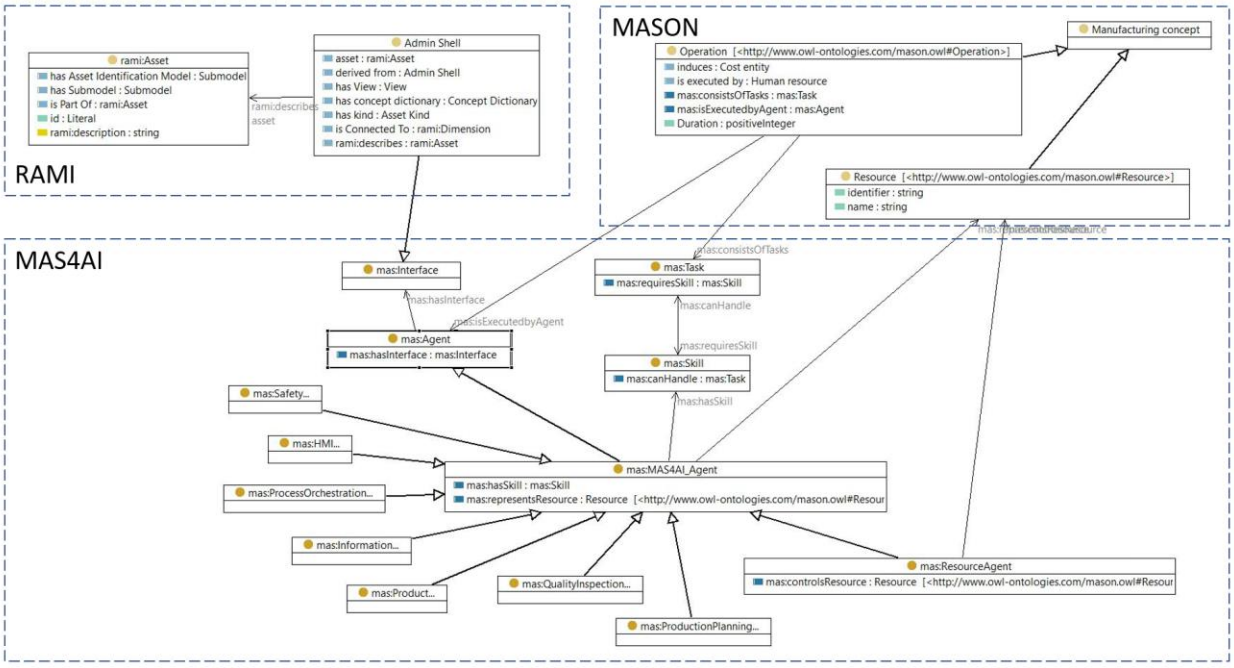

Figure 2 Concepts of the MAS4AI ontology and their relations to MASON and RAMI

The concepts and their properties are related as follows (Figure 2): *Mas4AI Agents* are a subtype of *Agents* that implement the characteristics required by MAS4AI via an *Interface (hasInterface)* that is the *Admin Shell* (AAS). The *MAS4AI agent* possesses a *Skill* (*hasSkill*), which it can use to handle (*canHandle*) a certain *Task*. A manufacturing process is defined as an *Operation* (from MASON) and typically consists of multiple *Tasks* (*consistsOfTasks*). A *Task* in turn requires a certain Skill (*requiresSkill*) to be able to perform that task. An *Operation* is executed by an *Agent* (*executedByAgent*), which can be a human or a *MAS4AI Agent*. A MAS4AI Agent represents a manufacturing *Resource* (from MASON). The *MAS4AI Agent* has eight different (generic) subtypes of Agents for *Safety, HMI/HCC, Process Orchestration, Information, Product, Quality Inspection, Production Planning* and *Resource*, as defined in the mAS4AI deliverable D1.2. A *Resource Agent* specifically has a function to control a *Resource*.

The class view for the MAS4AI namespace concepts is depicted in Figure 3.

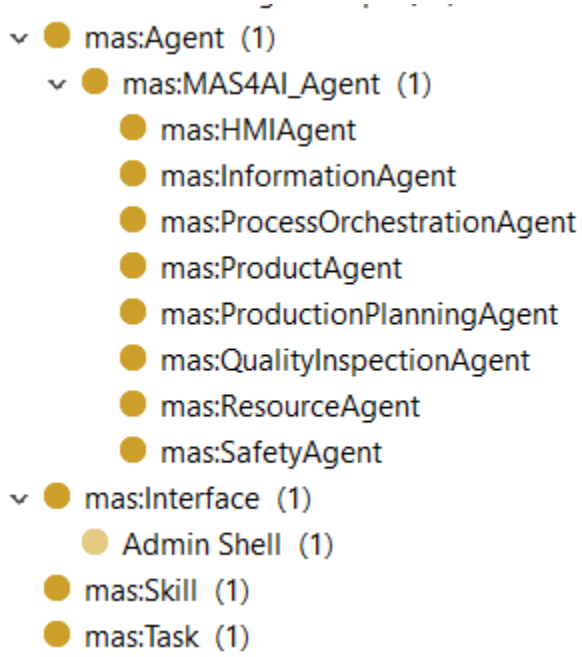

Figure 3 Class Hierarchy view of the MAS4AI ontology concepts

2.2 Ontology cloud instantiation

The MAS4AI ontology is made available to the use cases in an online RDF store, Dydra. Dydra is a cloud-based graph database [Dyd22] that allows programmatic access to the ontology via RESTful APIs. The APIs to the ontology are defined via (stored) SPARQL queries and then turned into REST methods.

The same base ontology can be made available to all use cases in separate environments, which allows extending the ontology and filling it with instance data according to the needs of the use case. This way any information deemed confidential is restricted to the use case alone and there is no interference between use cases on schema and instance data.

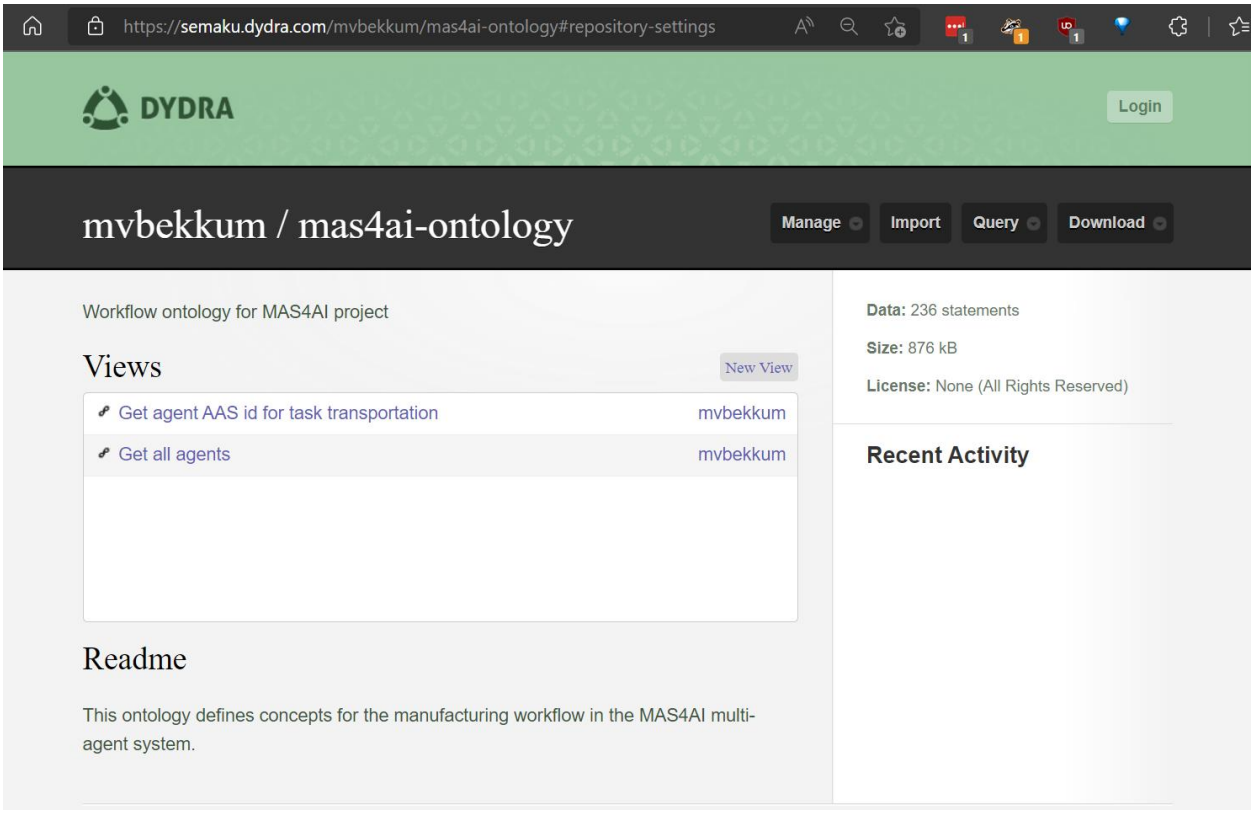

Figure 4 Dydra cloud environment

2.3 Ontology queries

The ontology and the RDF knowledge base aim to provide three specific queries in the MAS4AI framework:

- Provide an overview of all the agent types required, specifically those to be coordinated by the holonic agent.
- Allow to insert an AAS reference ID for an agent currently active.
- Output AAS reference ids for all agents suitable to perform a certain activity.

The queries are stored in the Dydra graph database (Figure 5) to be executed and make use of the advanced reasoning mechanism of RDF, which allows to complex multi-hop inferences over concepts and relations.

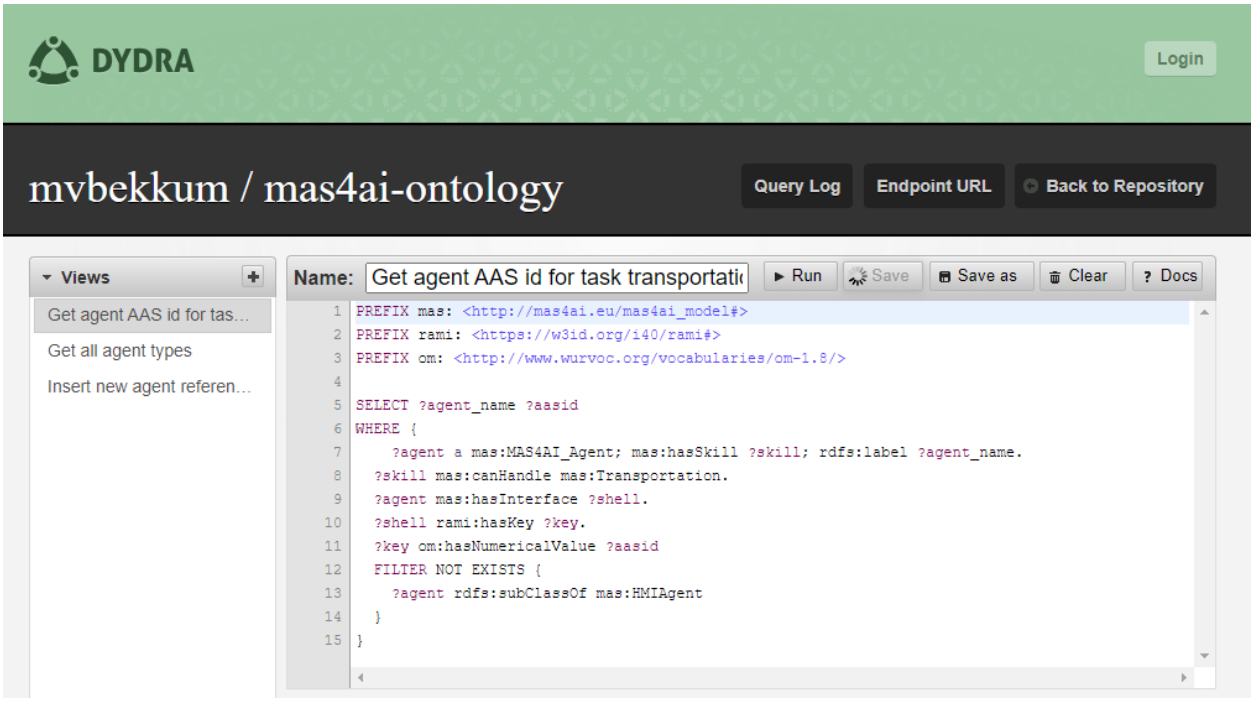

Figure 5 Stored SPARQL queries in Dydra

Query 1: Overview of agent types

The first query requests all agent types required for spawning within the current holon agent (Figure 6), that are a subtype of the MAS4AI agent type. z

```
PREFIX mas: <http://mas4ai.eu/mas4ai_model#>
PREFIX rami: <https://w3id.org/i40/rami#>

SELECT ?agenttype
WHERE {
   ?agenttype rdfs:subClassOf mas:MAS4AI_Agent
}
```

Figure 6 Query: list agent types

The response lists the eight agent types:

?agenttype
http://mas4ai.eu/mas4ai_model#HMIAgent
http://mas4ai.eu/mas4ai_model#InformationAgent
http://mas4ai.eu/mas4ai_model#ProcessOrchestrationAgent
http://mas4ai.eu/mas4ai_model#ProductAgent
http://mas4ai.eu/mas4ai_model#ProductionPlanningAgent
http://mas4ai.eu/mas4ai_model#QualityInspectionAgent
http://mas4ai.eu/mas4ai_model#ResourceAgent
http://mas4ai.eu/mas4ai_model#SafetyAgent

Query type 2: updates (reference IDs) for agents

The second type of request is an update query in order to register the reference ID used by the active agent(s). These updates can e.g. be made by the holon, i.e., the registration of its own reference. In the example below, we have added a Product agent to the repository, with an AAS reference id 'refid3456'.

```
INSERT Data {
   mas:new_agent a mas:ProductAgent; mas:hasInterface mas:AAS_new.
   mas:AAS_new a rami:AdminShell; rami:hasKey mas:Key_new.
   mas:Key_new a rami:Key; om:hasNumericalValue mas:refid3456
}
```

Figure 7 Query: insert new agent with reference

Query type 3: query for agents to perform a certain activity

The third type of query is a request for agent types that have the capability to perform a certain activity. In the example below, we have stated the query for the reference id of an agent that can perform a transportation activity, with the agent not being an HMI agent.

```
PREFIX mas: <http://mas4ai.eu/mas4ai_model#>
PREFIX rami: <https://w3id.org/i40/rami#>
PREFIX om: <http://www.wurvoc.org/vocabularies/om-1.8/>

SELECT ?agent_name ?aasid
WHERE {
    ?agent a mas:MAS4AI_Agent; mas:hasSkill ?skill; rdfs:label ?agent_name.
    ?skill mas:canHandle mas:Transportation.
    ?agent mas:hasInterface ?shell.
    ?shell rami:hasKey ?key.
    ?key om:hasNumericalValue ?aasid
    FILTER NOT EXISTS {
        ?agent rdfs:subClassOf mas:HMIAgent
    }
}
```

Figure 8 Query: list agent that can perform transportation

The response lists a transportation agent with its associated AAS reference identifier.

?agent_name	?aasid
Transportation agent 1	refid1234

3 AAS

As the Asset Administration Shell (AAS) is an important part of the semantic modelling in the MAS4AI framework, this chapter further explores what the AAS is, how it is used and what kind of models have been created using it.

3.1 Mapping to AAS schema structure

The Asset Administration Shell (AAS) serves as an abstract meta model in which specific concepts are expressed. Through the standardization of this meta language, it is possible to automatically interact with the specific concepts without having any pre-existing knowledge about them.

Core within the AAS metamodel is the usage of submodels to group properties and relations, identifying these concepts using a URI as semantic ID and reusing already existing concepts or (sub)models.

3.1.1 URI Strategy

In accordance with FAIR data standards, the AAS makes use of globally unique and persistent identifiers [1]. Within the AAS these are called *"Semantic identifier".* These identifiers are used to uniquely identify data and the way they are created reflects their intended usage. The approach regarding these Universally unique identifiers (URIs) is described in the URI Strategy below. This strategy is applied to both the AAS modelling, and the RDF ontology modelling, as these two are parts of the same modelling process.

We opted to use meaningful URIs for our model, meaning the ontology classes and properties, as well as the AAS submodels and properties. Rationale being that although opaque URIs are easier to keep persistent, even when the concepts they identify change, we are seeing that this leads to confusion by developers. This is especially true within the industry domain where semantic modelling is relatively new and little experience with URIs is present. Using meaningful URIs allows developers to get a feeling for the data they are getting just by looking at the URI. This makes it easier to select that data based on the URI, opposed to for example a provided label. Instantiations of the model are however identified with opaque URIs to prevent reuse of a single identifier for, for example, two different individuals of the same type.

Wherever possible the concepts in the MAS4AI model are aligned with, or reused from, existing models, such as RDF ontologies or pre-existing AAS models. In these cases, the URI or semantic ID from these models are of course presented in the MAS4AI model. However, not all modelling

[1] FAIR principals: https://www.go-fair.org/fair-principles/f1-meta-data-assigned-globally-unique-persistent-identifiers/

concepts are already defined in a pre-existing standard. In these cases, the URI uses the project specific namespace: `https://mas4ai.eu/model`, as this is managed by the project itself and there is no risk of namespace conflicts.

The current MAS4AI model does not yet fully adhere to this URI strategy, as per the used modelling approach [2] assigning semantics, and thus proper URIs is part of the second iteration phase, after the conceptual modelling and evaluation of these. As such, not all models and concepts have already completed the Industry 4.0 compliance phase.

3.1.2 MAS4AI Information Representation

The AAS modelling in this project focusses on providing generally applicable and reusable AAS submodels to describe agents in a multi agent system. This means there is a strong focus on the core concepts which occur in every use case, and a relatively smaller amount of attention is given to the detailed properties such a concept may have. The goal in this project is to provide use cases and agent developers with a starting point for their AAS modelling, which they can take as is and easily expand upon. As such, we develop many distinct submodel templates which each cover a different piece of information. This allows easily including one submodel, without this pulling in concepts which are not of interest to a given use case.

In the modelling extra attention was given so far to the relations between submodels. As these can be difficult to identify when only using one or two submodels and having such relations pre-defined aids in the discovery of new submodels by following the references from one to the other model. Upon instantiating a submodel template, the relationship between the model and another model should be instantiated as well, linking two concept instantiations together.

This combination of submodels together define an asset, which in this project is specifically an agent. This differs from how, for example, the RDF model defines an instance. Whereas RDF uses the notion of a “type” to explicitly define what an instantiation is, the AAS instead defines everything of an instance, except for its type. In IT modelling this is sometimes called duck typing [3], from the saying "If it walks like a duck and it quacks like a duck, then it must be a duck". This is a more flexible approach as it is possible to only add the submodels which are needed for the specific agent. However, this makes it more difficult to determine what sort of agent one is dealing with. Within the MAS4AI framework it is still possible to find what the type of an agent is by querying the RDF store, in which this type is explicitly given.

[2] AAS modelling approach: https://ieeexplore.ieee.org/abstract/document/9610633
[3] https://en.wikipedia.org/wiki/Duck_typing

3.1.3 Submodels

Within the AAS, a submodel is grouping of elements on both a conceptual and technical level. Within an AAS the submodel is the highest-level modelling construct and provides a way to structure the concepts within the model. For example, one may have a submodel “Identification” which contains all the properties and references needed to identify an asset. This makes it easier for the person using the model to understand what the properties in a model are supposed to be used for.

Besides this structuring for a person using the submodel, it is also used when accessing the AAS. An application may ask which submodels exist, or which properties exist within a submodel. This makes it possible to find related properties, which exist within the same submodel and get all these even when initially only one of the properties was known to be required.

Finally, AAS submodels play a vital role in the reuse of modelling components. As it is easy to export, import and reference AAS submodels they often provide the scope of reuse. Instead of reusing all different properties, or having to refence an entire AAS, one may only reuse a submodel from another project. This reuse reason is the primary reason why in the MAS4AI modelling work various submodels have been defined. Each of the submodels contains a set of related properties which can be reused as a whole submodel at once for a function described on the submodel level. For more details on how to use the AAS, and submodels within it, please see: details of the administration shell [DAAS1].

In the next chapter, the modelling structure based on submodels is reused in how the modelling outcomes are presented. That is to say, the properties and references are presented grouped by submodel. As they stay together when reusing a MAS4AI submodel.

3.2 Definitions of AAS elements

In this chapter the different AAS concepts are presented, structed along the agent, and then the submodels of that agent

3.2.1 Standard MAS4AI Elements

Every MAS4AI Agent has a set of standard models, which are universal. The set of those models encapsulates the general information that describes the standard characteristics of the created algorithms. Furthermore, the standard models contain information needed to integrate the agent within the MAS4AI framework, such as agent status.

NAMEPLATE

The nameplate model is a standard AAS submodel that was originally intended to be used for the representation of physical assets. As it contains several properties of relevance to digital assets

it is currently used within MAS4AI agents as well. However, as MAS4AI AAS models are about software assets instead of physical assets, a different type of model is eventually needed specializing the nameplate model to be used for agents.

IDENTIFICATION

The identification submodel is a standard AAS submodel that was originally developed for use with physical assets. It contains information regarding manufacturer, revisions, logos etc. Like the *Nameplate* submodel, here a lot of the information already present can be reused for software assets. However, a separate software (agent) focused version will be developed within the next section of the project to better match the software agents, like the nameplate model.

AGENT STATUS

The agent status submodel encapsulates all the relevant information regarding the status of the agent. This data is required by the MAS4AI framework and is created to fulfil this technical need. In the upcoming period it will be assessed to what extend this should be a separate model and whether having an AAS submodel created to facilitate implementation decisions is desirable. Alternatively, the agent status model may be absorbed in the agent lifecycle model.

AGENT LIFECYCLE

The lifecycle of an agent is required as defined by the FIPA Agent Management Specification [4]. It is needed for the agent to function within the MAS4AI framework and contains the current stage in the agent's lifecycle, such as active or waiting. Currently, only few components of the FIPA specification have been reused within semantic AAS models. As parts of this are already handled by other components in the MAS4AI framework. However, it may be that in the next phase of the project more FIPA components need to be implemented within the AAS Models to support all multi agent functionality.

MAS4AI AGENT

The MAS4AI Agent submodel contains information, that is specific for the MAS4AI framework. Given the specific use case this is where the reference to the RDF store is described. Aside from that, any other specific information that is required and is MAS4AI specific will be hosted here. Currently only two properties are known to be needed, but as more are expected these are already placed within their own submodel.

[4] *FIPA Agent Management Specification: http://www.fipa.org/specs/fipa00023/XC00023H.html#*

Table 2 MAS4AI framework specific properties

Name	Type	Value From	Description
RDF instance URI	Reference Element / Property	User/Other Program	Represents the connection to the RDF store that contains information about the other agents and their capabilities.
Part of Holon	Property	User/Other Program	Specifies what holon the agent is part of.

CONFIGURATION PARAMETERS

Every agent uses some sort of configuration parameters that are specific to it and required when initializing the system and/or when starting a particular task that is to be executed. The exact content of those parameters is highly dependent on the specific agent. For example, this submodel can contain the amount of memory available to a software agent or the goal of the agent. As such, only a high-level sub model is currently provided which can be filled with properties by the developer.

AVAILABLE AGENTS

The available agents submodel contains the set of agents that are currently available to execute a given task. This is especially important for agents such as the process orchestration agent, which is required to determine to which resource to give a certain task. The exact information is stored in the agent status and lifecycle models of the other agents and is referenced from the available agents submodel. However, during implementation it may be found that this information can easier be acquired through other processes, such as querying the RDF store and retrieving other agents' statuses from there. In that case, this model will be deprecated.

OTHER SUBMODELS

Aside from the seven general submodels defined in the sections above, it is possible that other submodels are needed by all agents in a specific use case. As this is highly use case specific and cannot be generalized directly to apply to all use cases, it was decided to not address this further in this iteration of the model's design.

3.2.2 Product Agent Submodels

The standard Product Agent model consists out of three sets of submodels – Standard MAS4AI submodels, Physical Components and Outputs.

PHYSICAL COMPONENTS

The main information within the product agent is a description of the actual product – technical information, specifications, sketches. The extend of detail within every submodel depends on the exact product and use case.

Table 3 Product Specifications

Name	Type	Value From	Description
Technical Data	Submodel	User/Other Program	Contains set of technical specifications about the product
Product Manufacturing Instructions	Submodel	User/Other Program	Set of instructions regarding how the product needs to be manufactured
Product Design	Submodel	User/Other Program	CAD files, Sketches and other relevant design properties
Requirements	Submodel	User/Other Program	Set of given requirements concerning the exact product to be made
Other Specifications	Submodel	User/Other Program	Depending on the product there may be more specifications present

Aside from the product specifications, which describe its physical properties and manufacturing process, the agent also tracks the status of the product development. The table below shows the properties that are used to describe the corresponding cycle stages.

Table 4 Lifecycle of the product

Name	Type	Value From	Description
Cycle Stage	Property	Process Orchestration Agent	Represents the current stage in which the production is
Current Production Status	Property	User/Other Program	Current Production Status of the production manufacturing process

OUTPUT

The output of the product agent is a report regarding the current product status. For all other relevant information, the agents can directly refer to the product agent.

Table 5 Product Agent Output

Name	Type	Value From	Description
Product Status	Reference element	Generated from the Agent's process	Description of the current status of the product (for e.g. such as *In process, To be started)*
Production Status	Property		Notes the exact status of the production – at which step the product is – *Milling, Welding, Quality Inspection* etc.

3.2.3 Resource Agent Submodels

The Resource Agent AAS has four groups of submodels – Standard MAS4AI models, Task(s), Physical Components and Output(s). Each set contains multiple different submodels describing separate elements.

TASK

The tasks of the resource agent can come from two different place. Originally it would receive information from the process orchestration on the task to execute. However, depending on the use case it could be that there is also another entity which can provide a task (user, external program, other agent etc.)

Table 6 Resource Agent Task Descriptions

Name	Type	Value From	Description
Execution Task	Reference element	Process Orchestration Agent	Exact task that the agent needs to execute following the made planning
Other Task	Property	User/Other Program	Task for the agent provided in a relevant form

PHYSICAL COMPONENTS

The resource keeps track of a set of information, which is considered relevant for the given resource. The definition of a resource encapsulates a diverse set of assets – could be a machine, a logistics vehicle, a certain department or another relevant factory element.

Table 7 Manufacturing resources of Resource Agent

Name	Type	Value From	Description

Operator information	Reference element	HMI Agent	Information regarding the operator of the given resource – it can contain information about skills or other relevant operator information.[5]
Components	Submodel	User/Other Program	List of relevant components for the type of resource (for e.g. types of tools a robot arm can use)
Materials (Raw)	Submodel	User/Other Program	The set of materials that the resource has access to / can use (such as steel etc.)
Manufacturing Equipment	Submodel	User/Other Program	Description regarding the exact equipment with the corresponding specifications
Energy	Submodel	User/Other Program	Information about the energy consumption of the resource

Table 8 Department Specifications

Name	Type	Value From	Description
Storage	Submodel Element Collection	User/Other Program	Provides information about the relevant information regarding the storage facilities of a given department – what is present there, what space is available etc.
Other Relevant Information	Property/ Submodel Element Collection	User/Other Program	Depending on the exact department there may be other relevant information needed (such as e.g. available operators)

Table 9 Transport Resource

Name	Type	Value From	Description
AVG	Submodel	User/Other Program	Describes the relevant parameters of an AVG.

[5] The operator data, which is allowed to be gathered, is highly dependent on the exact GDPR agreement, as it may contain personal information.

Logistics	Submodel	User/Other Program	Describes the relevant parameters of a logistics resource (e.g. truck id, parameters, type)
Other Type of Vehicle	Submodel	User/Other Program	Describes the relevant parameters of a specific type of vehicle.

OUTPUT

The resource agent's output provides information about the status of the corresponding process. Again, depending on the type of exact resource, the type of output will differ – for manufacturing resource such as a welding machine, the output would be production status.

Table 10 Output of Resource Agent

Name	Type	Value From	Description
Manufacturing Status	Property	Generated by the agent's internal analysis	Status of the current manufacturing process (such as *complete* etc)
Manufacturing Parameters	Property		Manufacturing parameters if relevant (e.g. exact grinding wheels needed for the next production step)
Transportation Status	Property		Status of the transportation process (such as *in transit*)
Department Status	Property		Status of the department's given task

It is important to note that it is possible to have multiple types of output, as they highly depend on the exact type of resource. It can also be the case that the output is not a property, but a set of them in the form of a submodel element collection.

3.2.4 Planning Agent Submodels

The standard Planning Agent AAS has four sets of submodels – Standard MAS4AI models, Task(s), Available Information and Output(s). Each set contains multiple different submodels describing separate elements.

TASK

The task of the planning agent is provided in the shape of a certain set of order(s) and possibly corresponding product design. What exactly the order contains highly depends on the use case as different companies have varying personalized forms that they use.

Table 11 Task for Planning Agent

Name	Type	Value From	Description
Product Design	Reference element	Product Agent	Provide information about the product design of the product(s) which are contained in the order
Order(s)	Property or Submodel Element Collection	User/Other Program	Information about the exact set of orders that is provided to the planning agent. Depending on the case this can be a single property or a submodel element collection with more information included.

AVAILABLE INFORMATION

The planning agent requires information from different agents to be able to complete its task. It refers to the details of the required product, the available resources and their status.

Table 12 Available information for the planning agent

Name	Type	Value From	Description
Product	Reference element	Product Agent	Information about the specifications of the product(s) within the given order
Resources	Reference element	Resource Agent	Information about the set of resources, available to the agent
Status	Reference element	Resource Agent	The stations of every resource of interest
Other specifications	Property (or Collection)	User/Other Program	Any other relevant information and specifications that are unique/specific to the exact agent and/or use case.

OUTPUT PLANNING

The output of the agent is an optimized plan. This can be about machines, operators of these machines or the logistical capabilities associated with the manufacturing process. Currently the plan's format is left unspecified, as this is under development in work package 4.

Table 13 Output of the Planning Agent

Name	Type	Value	Description
Machines Plan	Property	Generated from the agent's analysis	Set of plans for the required machines
Transport Plan	Property		Set of plans for the required logistics services
Operator Plan	Property		Set of plans for the required operators

3.2.5 Process Orchestration Agent Submodels

The standard Process Orchestration Agent AAS has four sets of submodels – Standard MAS4AI models, Task(s), Available Resources and Output(s). Each set contains multiple different submodels describing separate elements.

TASK

The task of the process orchestration agent is to execute the created plan from the planning agent. It can also be the case that there is another specific task (such as handle deviation or resource problem) depending on the exact use case.

Table 14 Tasks of the Process Orchestration Agent

Name	Type	Value From	Description
Resource Plan	Reference element	Planning Agent	The exact order execution plan as created by the planning agent
Other Task	Property	User/Other program	Task defined in an appropriate format (depending on the situation it can also be an element of different type)

AVAILABLE RESOURCES

To create a Resource usage plan the agent needs to have information about the availability of the different resources.

Table 15 Available resources to the Process Orchestration Agent

Name	Type	Value From	Description
Manufacturing resources	Reference element	Resource Agent	Information about the status of all manufacturing resources
Department	Reference element	Resource Agent	Information about the department(s)
Logistic Resources	Reference element	Resource Agent	Information about the logistic resources
Product	Reference Element	Product Agent	Possible relevant about the product(s) that need to be manufactured

OUTPUT

The output of the process orchestration agent is a set of tasks to specific other agents that are needed to be fulfilled. All those tasks together form the resource usage plan.

Table 16 Output of the Process Orchestration Agent

Name	Type	Value From	Description
Resource Usage Plan	Property	Generated from the agent's analysis	Resource usage plan detailing the exact resources to be used to execute the created production plan
Execution task	Property		Exact execution tasks towards the HMI agent
Quality Task	Property		Exact quality task to be communicated towards the Quality Agent
Execution Task	Property		Exact execution task towards the Resource Agent

3.2.6 Quality Agent Submodels

The standard Quality Agent AAS has four sets of submodels – Standard MAS4AI models, Product of Interest, Current State and Output(s). Each set contains multiple different submodels describing separate elements.

PRODUCT OF INTEREST

The quality agent focuses on analysing the quality of a given product. Therefore, the task that it receives from an external party can point to the exact product of interest and/or further specifications that must be taken into account.

Table 17 Quality Agent tasks

Name	Type	Value From	Description
Quality Task	Reference element	Process Orchestration Agent	Description of the exact product of which the quality needs to be evaluated
Other Task	Property	User/Other Program	Information about any other type of quality task (such as specifically predefined by the user task)

CURRENT STATE

To access the quality, the agent needs to receive information regarding the expected and current product measurements. Furthermore, to be able to track what the quality is related to, the agent also keeps track of the resource that the product is related to (e.g. the exact machine that is used to produce the current state of the product).

Table 18 Status Elements of the Quality Agent

Name	Type	Value From	Description
Expected Product Measurements	Reference element	Product Agent	Expected product measurements as given by the technical specifications of the product
Current Product measurements	Reference element	Product Agent	Current product measurements as measured after the product has been produced / passed certain production steps
Other Relevant Measurements	Reference element	Product Agent	Information about other relevant measurements that are needed to determine the product quality
Resource Status	Reference element	Resource Agent	The status of the resource that is used for the production of the product of interest
Other Relevant Measurements	Reference element	Resource Agent	Information about other relevant measurements from the

			resource (such as defined tolerances of the machine)

OUTPUT

The main output of the quality agent is a set of quality analysis parameters. It provides the current quality and depending on the settings it can also provide a trigger alert to other agents if the quality is below a certain threshold.

Table 19 Output of Quality Agent

Name	Type	Value From	Description
Current Quality	Property	Generated by the agent's internal analysis	Information about the current product quality given in a certain format (if there are multiple measurements this element can be extended to a submodel element collection)
Quality Feedback	Property		Send towards a particular resource if the quality issue is determined to originate from there.
Action Trigger	Property		Sends towards the process orchestration in cases when the quality is below a certain threshold

3.2.7 Safety Agent Submodels

The standard safety Agent AAS has four sets of submodels – Standard MAS4AI models, Task(s), Safety Elements and Output(s). Each set contains multiple different submodels describing separate elements.

TASK

The safety agent focuses on assessing the current safety level of given process within a factory. The scale of it can differ depending on the exact use case. The exact task is provided by either another agent, user or algorithm and can have different formats. In certain cases, it can also be relevant to define certain risk threshold which is a reference value when performing the analysis.

Table 20 Task of the Safety Agent

Name	Type	Value From	Description
Safety Task	Property	User/Other Program	Describes the type of task that the safety agent receives. Depending on the situation that can be a direct communication line to the process orchestration agent.
Risk Threshold	Property	User/Other Program	A pre-defined threshold of risk, which is used to after that determine the need for triggering a safety alert.

SAFETY ELEMENTS

To assess the safety levels the agent requires information about the current state of the different safety component (sensors) within the system. Depending on the analysis, the agent may also need information about the exact way the product will be developed so that it can assess any risks from e.g. hazardous materials required.

Table 21 Safety elements information

Name	Type	Elements		Description
Resource Safety	Submodel Element Collection	Safety Sensor	Reference Element	References to a given resource agent to extract information about its safety sensor(s) and their current state
		Safety Settings	Reference Element	References to a given resource agent to extract its safety settings
Product Requirements	Submodel Element Collection	Risk Level	Reference Element	Extract from a given product agent the risk level that corresponds to the given product
		Other Parameters	Reference Element	Extract any other relevant parameters from the product agent that are needed for the safety task

OUTPUT

The output of the safety agent is a certain set of analysis results. It can contain a risk level value, a plan for what steps need to be done given the risks and if required a certain set of alerts in case the calculated risk is above the given threshold.

Table 22 Safety Agent output

Name	Type	Value	Description
Risk Level	Property	Generated from the agent's analysis	This describes the risk level in the current task/system within a set of specific parameters.
Safety Plan	Property		For specific situations it may be required to create a safety plan that can be used to decrease the risk level or assure safe work. In this case the safety plan would contain this information.
Safety Alert	Property		When the risk is above a certain specific risk threshold, the agent may trigger a safety alert towards other agents.

3.2.8 HMI Agent Submodels

The standard HMI Agent AAS has four sets of submodels – Standard MAS4AI models, Task(s), Available Resources and Output(s). Each set contains multiple different submodels describing separate elements.

TASK

The HMI agent may receive tasks from the Process Orchestration Agent, as part of the tasks from the current resource plan. Or the HMI agent may receive a separate other task that can originate from another different entity.

Table 23: Task of the HMI agent

Name	Type	Value From	Description
Execution Task	Reference Element	Process Orchestration Agent	Defines a specific task from the process orchestration agent
Other Task	Property	User/Other Program	A particular task that the agent must execute.

Available Resources

The HMI agent can require different information depending on the exact type of action it is required to execute. However, the focus is on getting information about the machine of interest, its operator and the instructions for manufacturing or other needed information from separate agents.

Table 24: Available resources to the HMI agent

Name	Type	Value From	Description
Manufacturing Resource	Reference element	Resource Agent	Reference to the manufacturing resource that is of interest to the HMI agent
Operator Information	Reference element	Resource Agent	Information about the human operator/participant within the interaction
Manufacturing Instructions	Reference element	Product Agent	Instructions how the product of interest can be manufactured
Other Specifications	Reference element or Property	Other Agent / User/Program	Other relevant information that is essential to the HMI agent

Output

The exact output of the agent depends on the particular task that it needs to execute, as the definition of HMI is very broad. We define two main elements – product state, in the situation where there is a product resulting from the human-machine cooperation, and operator information in case there is an assessment happening.

Table 25: Output of the HMI agent

Name	Type	Value	Description
Product State	Property	Generated from the agent's analysis	Describes the state of the product that the HMI agent was assisting with (for example *done* for the manual assembly of a product)

Operator Information	Property		Any relevant information about the Operator (such as a new assessment of the level of the given skill)
Other specifications	Property		Any other relevant information that can be provided from the agent.

3.2.9 Information Agent Submodels

The standard Information Agent AAS has four sets of submodels – Standard MAS4AI models, Current Plan, Current State and Output(s). Each set contains multiple different submodels describing separate elements.

TASK

The task of the information agent is generally to gather information about all the processes and resources that are of relevance in the current moment within the factory. That is why the initial input of the agent is the current production and resource plan.

Table 26 Information Agent Input

Name	Type	Value From	Description
Production Planning	Reference element	Planning Agent	Current production plan linked to the planning agent
Resource Plan	Reference element	Process Orchestration Agent	Current resource plan linked to the process orchestration agent

REFERENCE ELEMENTS

The agent gathers information from every running agent that is currently active and of interest (given the created planning).

Table 27 Reference Elements within the Information Agent's input

Name	Type	Value From	Description
Product Status	Reference element	Product Agent	Information about the current product status in relation to manufacturing process

Resource Status	Reference element	Resource Agent	Information about the current status of the resource (current state and possible errors tracking)
Current Risks	Reference element	Safety Agent	Information about the risk evaluation from the safety agent
Current Quality	Reference element	Quality Agent	Information about the quality of the current products of interest from the quality agent(s)

OUTPUT

The exact output of the information agent depends on the type of analysis it is required to do. Therefore, it is broken down into three main categories deviation, production state, and other analysis.

Table 28 Information Agent's output

Name	Type	Value	Description
Deviations	Property	Generated from the agent's analysis	Information about detection of any deviations within the production process
Production State	Property		Information about the current production state of the created plans/orders/products
Other Analysis	Property		Any other relevant analysis that can be used within the exact use case.

3.2.10 Other Relevant Submodels

The actual content of an AAS model can highly vary depending on the specific situation. Just in MAS4AI the different use cases already present highly diverse situations, which come from different settings and have different information requirements for certain points.

3.3 Access / API

In order to use all of these models, they need to be accessible to applications. For this, an application programming interface (API) is needed. The AAS models themselves don't yet provide

such an API until they are hosted on an AAS-server. This server allows interaction with the models through a pre-defined API specification[6].

In the MAS4AI framework, the open-source Basyx middleware is used as AAS-server. It adheres to the API specification[7] and can host multiple AASs and submodels on a single server. The Basyx AAS server allows agents to ask questions like: which submodels does a given AAS have? Or: what is the value of a given property? The response to these questions depends on the AAS model, but the way of asking them is always the same. Allowing the usage of a standard and stable API, using different, of changing, semantic AAS models.

[6] AAS API Specification: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part2_V1.html
[7] Basyx REST specification: https://app.swaggerhub.com/apis/BaSyx/basyx_asset_administration_shell_http_rest_api/v1

4 Conclusions

4.1 Application of the ontology

The current instalment of the ontology provides a sound, generic knowledge base for all use cases. During the application of the ontology in use cases in work package 6, it has to be extended on a use case specific basis. The real-world application requires that specific activities and perhaps specific types of agents as subclasses of current types need to be included.

In addition, other types of queries may need to be defined, again depending on the requirements of the use cases. The current set of queries are related to the setup that has been defined as a starting point for the architecture (cf. deliverable D2.1). A more extensive role of the ontology could similarly require other/more requests from the RDF store. This could perhaps include a setup where the ontology specifies configuration parameters for a configuration-based agent setup, rather than having (pre-)programmed agents.

4.2 Application of the AAS

During the development of the AAS models a couple of things were discovered which will be taken into consideration during the next phase of the MAS4AI project. The preliminary conclusion after developing a number of AAS models is that there are currently very few relevant models to be found.

This issue is caused by three distinct root causes. Firstly, the AAS is relatively new and as such there are very few AAS models already developed in general. There are multiple projects underway to solve this issue, such as the DIMOFAC project [DIMO22], but also the MAS4AI project itself.

Secondly, the models which are already developed are targeted at physical assets. For example, the commonly used “nameplate” model describes a number of properties of relevance to agents, but also contains a section on exterior markings on a product. Which is, of course, not relevant to a digital asset. This places MAS4AI in a greenfield situation where there is little existing AAS models to consider, but also leaves MAS4AI with few AAS standards to build upon, forcing the interpretation and incorporation of external non-AAS models instead. This issue is in part the reason to divide the MAS4AI models into a large number of submodels, which improves reuse of individual models and increases the chance of the developed models to be of relevance to future developments.

The final issue encountered has to do with the findability of AAS submodels. Although the AAS standard describes registries which can be used to share AAS templates and models, in practice only very limited public AAS registries exist. Instead, getting access to models involves knowing

someone who happens to be working on a project where they already developed something. This serendipity based AAS discovery is not sustainable in the long run and is only functional due to the small community of actors creating these models. To solve this issue, MAS4AI collaborates with the DIMOFAC project to develop a web application which can be used to store, explore and share AAS templates and instances. This is based on the already existing DIMOFAC platform.

5 References

[Dyd22] Dydra, http://docs.dydra.com/dydra

[DIMO22] *Digital Intelligent MOdular FACtories*. HORIZON DIMOFAC Project, https://dimofac.eu/

[DAAS1] *Details of the Asset Administration Shell – part 1*, https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html