Enabling Semantic Interoperability of Asset Administration Shells Through an Ontology-based Modeling Method

Yining Huang, Saadia Douib, Luis Palacios Medinacelli, Jacques Malenfant

Enabling Semantic Interoperability of Asset Administration Shells Through an Ontology-based Modeling Method

Yining Huang
yining.huang@cea.fr
Université Paris-Saclay
CEA, List
Palaiseau, France

Saadia Dhouib
saadia.dhouib@cea.fr
Université Paris-Saclay
CEA, List
Palaiseau, France

Luis Palacios Medinacelli
luis.palacios@cea.fr
Université Paris-Saclay
CEA, List
Palaiseau, France

Jacques Malenfant
jacques.malenfant@lip6.fr
Laboratoire d'Informatique
de Paris 6 (LIP6)
CNRS, Sorbonne Université
Paris, France

ABSTRACT

Industry 4.0 currently prepares a major shift towards extreme flexibility into production lines management. Its goal is to fully automate the process by which series of client orders are transformed into executable production plans on reconfigurable production lines in order to produce series of shorter and highly customized lots of produces in an economically sustainable way. In this paper, we present our first steps towards the design and implementation of an automated I4.0 flexible plant supervision and control system based on MDE concepts within the "Papyrus for Manufacturing" toolset. We show how an MDE approach can aggregate around system modeling tools from the Papyrus platform both I4.0 technologies, such as digital twins and standard Asset Administration Shells, to represent produces, production plans and plant resources, and AI tools such as the MaRCO (Manufacturing Resource Capability Ontology) to provide semantic matching capabilities. More precisely, we address the matchmaking required to find among currently available plant resources the ones able to fulfill the requirements of a production plan. To overcome the limitation of current syntactic-only matching algorithms, we transform AAS concepts and data modeling plant resources into MaRCO ontological concepts and then query the expanded ontology to get the needed resources transformed back into modeling elements. This method has two main advantages (1) to provide semantic descriptions for the AAS models, (2) to complement model-driven engineering tools with reasoning features. This paper showcases this approach through a robotic cell use case. Future work will complete the process *i.e.*, extend our Papyrus-based implementation to generate executable production plans using the identified resources, reconfigure the plant to adapt it to this production plan, then execute the plan and monitor it using a digital twin approach to adapt in case of failure or other run-time incidents.

MODELS '22, October 23-28, 2022, Montreal, Canada

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Software and its engineering → System modeling languages**; **Interoperability**; • **Computing methodologies → Modeling methodologies**.

KEYWORDS

Model-Driven Engineering, Digital Twins, Ontology, Industry4.0, Asset Administration Shell, Semantic Interoperability, Smart Manufacturing

ACM Reference Format:
Yining Huang, Saadia Dhouib, Luis Palacios Medinacelli, and Jacques Malenfant. 2022. Enabling Semantic Interoperability of Asset Administration Shells Through an Ontology-based Modeling Method. In *Proceedings of ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS '22).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Industry 4.0 currently emerges as a comprehensive effort to bring extreme flexibility to production lines management. Our modern economy is seeking ever more customised produces to be made in ever smaller lots. To keep up with this trend and remain competitive, industry is shifting towards fully automated production processes by which client orders are transformed into a production plan and then executed on per lot reconfigured plants, taking the shortest time possible to go from one lot to the next. This automated process requires:

(1) To model produces, their production plans and the plant resources in their finest-grain details to generate timely targeted executable production plans on-the-fly.
(2) For each order in turn, to select currently available plant resources (tools, conveyors, robots, etc.) able to fulfill the requirements of the corresponding production plan.
(3) To reconfigure the plant to efficiently and timely execute each production plan with the selected resources.

(4) To monitor the execution of plans through a closed-loop supervisory control to adapt it upon production failures or incidents.

Developing such production line management systems promises to be a very challenging endeavour, especially in the highly heterogeneous and multi-vendors area of the plant equipment market. Besides well-known challenges in the supervisory control of complex cyber-physical systems, providing a comprehensive representation of produces, production plans and plant resources in an interoperable way among heterogeneous equipments represent a major challenge. To address these challenges, this vision of future flexible plants is currently supported by technical solutions such as (1) the digital twin technology to design structural and behavioral models@runtime, (2) interoperability standards, such as the Asset Administration Shell promoted by I4.0 as a standard interface to digitally represent all of I4.0 elements as well as (3) capability-based modeling which aims at representing and reasoning about production activities, taking into account functional and non-functional properties to generate and operate effective, efficient and economically sustainable production plans.

Digital twin technology [3] provides the direction for future intelligent manufacturing systems with a high degree of autonomy and self-adaptability. Reconfigurable manufacturing systems [24] permit a safe and reliable way to deal with the unforeseen situations that may be encountered on the production line. Despite that if without the interoperability and the adaptability of all the components involved in the production practice, this fantastic idea of reducing costs and improving business profits can only remain in the conception.

The emergence of the AAS [16] standard has put forward the specification of the unified interface of all the participants in a Industry 4.0 digital twin system, making them no longer vendor-dependent and technology-dependent. The adaptability of a system refers to the monitoring, re-planning and reconfiguration ability. An extension of the capability-based engineering [15] has been proposed in [13], where it explains different phases of the production activity and how the AAS models are intervening. Here the advantages of model-driven engineering (MDE) [4] can be manifested, as it can clearly bridge the gap between principles and implementation.

Indeed, though these technologies are very useful building blocks, they need to be brought together consistently and concretely to implement production line management systems. In this paper, we propose to use an MDE approach to do so and present the first steps towards a full-fledged automated I4.0 process based on MDE concepts and implemented within the "Papyrus for Manufacturing"[19] toolset. More precisely, we address the first two steps of the above automated process. Starting from the AAS concept, we develop a Papyrus model of AAS in order to implement as models all of required I4.0 elements: produces, production plans and plant resources. Then, the models can be exploited to implement concretely a capability-based modeling approach.

However, if the AAS standard provides syntactic interoperability for cross-vendor assets, it leaves the major issue of semantic interoperability unresolved. Many research units and groups have realized this semantic gap as a major shortcoming of AAS and studied on to better describe assets or try to propose a solution rather by referring to ontologies [31] or conduct model transformations [28]. However, no solution towards enabling comprehensive semantic interoperability of asset administration shells has been shown yet.

Ontologies indeed appear as a highly relevant approach to bring such semantic interoperability as they exhibit the ability to define semantic models of data combined with relevant domain knowledge, and to formulate inference strategies [25]. The construction of ontologies requires the knowledge of many domain experts, providing a core reusable ontology into Domain Specific Modeling (DSM) will save a lot of time in bringing semantic interoperability to the model [20]. Based on this concept, we are going to present our ontology-based AAS capability modeling approach which enables the semantic interoperability of AAS, as well as the implementation of an automated capability checking procedure in Papyrus modeling framework [14], hence a first step in the concrete realisation of capability-based engineering. In our approach, model-transformations are used to keep aligned a model-based and an ontology-based representation of the assets. By using this tool, we can obtain candidate resource combinations for production line reconfiguration from a well-defined AAS product model. And the result will be justified in a robotic cell use case.

This article is organized as follows, Section 2 presents the backgrounds and related works. Section 3 introduces the capability checking design. Section 4 focuses on the implementation of the capability matchmaking process in Papyrus. Section 5 provides a matchmaking example in a robotic cell use case. Finally, Section 6 concludes and introduces future works.

2 BACKGROUND & RELATED WORKS

2.1 Model-based Digital Twins, Industrie 4.0 & AAS

The purpose of the digital twin technology [3] is to simulate the operation of equipments with the best mathematical and behavioral models. With the rise of IoT and sensor networks, real time data from physical devices and business information have been incorporated into the scope of the model. As the complexity of the system increases, the transformation of the model itself as well as the integrability, connectivity and scalability between models become crucial. Under this premise, the Model-Driven Engineering (MDE) paradigm [4], which revolves around models and focuses on alignments and transformations between models, brings new potential to digital twins. At the same time, the modeling of the asset's entire life cycle at the core of the digital twin also brings self-adaptation and autonomy from design to operation to Model-Based Systems Engineering (MBSE) [21].

Digital twins do not exist in isolation, but are used in the construction of CPS (Cyber-Physical System) [10] and, in our area, CPPS (Cyber-Physical Production System) [30]. Since no unified description for them exists, it is difficult to realize the interoperability and scalability of digital twin models when they are designed and implemented independently. This problem becomes even more significant with the expansion of the system scale. In this context, the reference architecture model for Industry 4.0 (RAMI4.0) [12] and the Asset Administration Shell (AAS)[16] standard were

proposed to provide a unified architectural framework and standardized interfaces for Industry 4.0 (I4.0) systems. The RAMI4.0 can help manufacturing enterprises to open up vertical integration to solve the problem of data flow and transmission from production equipment, production execution, production planning to enterprise business operation management, that is, the connection from Level 0 to Level 4 in the ISA-95 manufacturing pyramid [1]. AAS is the digital representation of an asset, and an asset can represent any element within the I4.0 context.

An AAS as shown in Figure 1 may contain several submodels to describe functional aspects for different use cases by using suitable submodelelements. But these submodelelements do not deal with the semantic representation of capabilities and other submodelelements. So each submodelelement has a semanticId, which might either refer directly to a corresponding semantic definition provided by an external reference (e.g. eCl@ss [7] or IEC CDD property definition [5] or an ontology concept).

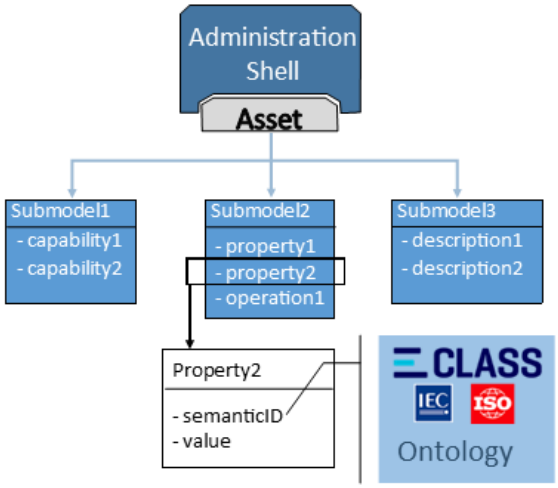

Figure 1: AAS Structure

2.2 Cabability-based Engineering

The adaptability is a crucial issue to ensure the reliability of a digital twin system. To address the problem of adaptability, an approach called capability-based engineering for the flexible production lines has been proposed. First published by Plattform Industrie 4.0 [15], which describes the concept and its operational realization. The term "capability" refers to an abstract description of the function of a production resource, while the ability to achieve a specific effect depends on the asset's "skill". PPR (processes, products and resources) are the three most fundamental building blocks in manufacturing. Capability-based engineering is intended to dynamically deploy resources *i.e.*, rather than specifying the actual production process directly, by defining the capabilities required by the production process of the product and let an automated production line management system find the resources and implement the process.

Based on this concept, the three key steps of capability-based engineering has been refined in [13] as capability checking, feasibility checking & skill execution. We then introduced a set of AAS modeling tools within a novel capability-based modeling environment [14]. This approach has been proposed to ensure continuous capability-based engineering while minimizing production line downtime.

2.3 Semantic Interoperability in manufacturing

In this article, we expand our previous work by focusing on semantic interoperability of AAS. Semantic interoperability has long been recongnized as a major concern in the field of industrial digital twin systems. This subsection introduces this problem, and then leads to two related works that will be reused in our solution.

The Digital Twin Consortium published a whitepaper [2] on the digital twin system interoperability framework. It introduces seven interoperability concepts that frame the design considerations necessary to make systems interoperate at scale. The article [6] introduces the definition of semantic interoperability in the context of Industry 4.0 and Smart Manufacturing as follows: "Semantic interoperability enables systems to interpret meaning from structured data in a contextual manner. Semantic interoperability relies on ontology-based "contextual metadata" supplementing "data" to form "information" exchanged among connected systems. This ontology must account for metadata exchanged between disparate systems and environments. It represents the highest level of interoperability between connected systems - beyond syntactic interoperability".

[31] provides an overview of various articles and applications of data analysis, expert knowledge, and knowledge-based system drivers in production systems. On top of that, it describes how to use "data analysis" in a production system to create knowledge-based digital twin systems. [22] articulates new concepts of value creation through the use of digital twin decision support services in industrial service ecosystems, and discusses mixed semantic modeling and model-based systems engineering for their implementation. A customizable conversion system for converting ABB Ability™ digital twins to Asset Administration Shell format is presented in [28], showing a real-world example for interoperability in industrial environments.

Ontologies bring to systems engineers and researchers the the high value of semantic interoperability and makes them aware of the importance of combining ontology vocabularies with system model design. [9] introduced an approach of a dynamic mapping of the ontology vocabularies into system models stereotyped by metaclasses defined in a profile. This approach enriches the semantic meanings of system modeling without affecting the definition of existing metamodels.

According to the above work and many other articles not mentioned, the use of ontologies to solve semantic interoperability appears as a common solution in the field. Our idea is to combine ontology-based knowledge representation with the AAS digital twins to achieve the semantic interoperability between digital twins. To achieve this, we rely on two former works described next:

MaRCO [18] provides capability-related ontology for manufacturing systems, and the OML Adapter [8] provides a transformation basis from OWL ontologies to OML and UML models.

2.3.1 Manufacturing resource capability ontology (MaRCO). The OWL-based Manufacturing Resource Capability Ontology (MaRCO) [18] is used to describe the capabilities of manufacturing resources. Ontologies are widely accepted for knowledge representation in specific domains. The expressive power of MaRCO supports the representation of simple resources but also their combination into collaborative resources, hence a good candidate for capability-based engineering. In addition, MaRCO is also provided as a complete capability matchmaking web service [23]. While its implementation language OWL has good knowledge representation features, pure OWL is limited when it comes to querying. To effectively support semantic-based resource selection, SPARQL (SPARQL Protocol and RDF Query Language) [29] has been chosen to implement the capability matchmaking rules. More precisely, SPARQL allows to write queries that combines the capability parameters of several resources to select sets of compatible and covering resources for given requirements. Finally, the use of SPIN (*SP*ARQL *I*nference *N*otation) allows to represent SPARQL queries as knowledge within the ontology and then the SPIN API allows to make inferences and generate new individuals within the ontology.

2.3.2 OML Adapter. The OML (Ontological Modelling Language) [8] is defined by the openCAESAR[1] platform, which is also an ontology description language inspired by OWL and SWRL (Semantic Web Rule Language). OML is a modeling language designed for ontologies, which aims to close the gap between modeling and programming languages. OML is implemented using the Eclipse Modeling Framework (EMF), which gives it a Java API and integration with useful tools such as OML Adapter provided by the openCAESAR project. However, the OML adapter only provides round-trip transformation between OML and UML. More details about this conversion is described in the following section, as we use OML in our platform.

3 MODEL BASED CAPABILITY CHECKING

Figure 2 shows the whole process of the capability-based engineering approach [15]. In a model-based Digital Twin production system, each resource (or asset) has its own representative AAS provided by different stakeholders (product and process designers, equipment supplier, integrator, etc.). The AAS contains the technical descriptions (nameplate), the simulation models, the operational data and other business information. The resource pool of a plant contains all the resources as well as the system layout design. During the design phase, the system architect specifies the products and their manufacturing processes. The rounded rectangles in the figure represent different levels in the automation pyramid. From top to bottom, they are representing the manufacturing operation management (level3), the monitoring and automated control (level2), and the manipulation of production processes (level1). In the latter level, the "Asset Administration Shells" (or digital twins) are considered as models@runtime since they are continuously updated to represent the assets real time status. Capability checking takes the PPR capability models as input and computes the possible resource combinations that may achieve the production. During the feasibility checking step, these combinations and environmental contexts will be simulated to validate the selected resources combination against their current constraints. Then the next step automatically supervises the skill execution of the selected models by the reconfiguration plan. The supervisor deploys the selected resource pool models according to the system reconfiguration plan obtained through the capability-based reconfiguration phase. During the whole process execution, the supervisor monitors the status of all asset models and will re-plan the production process in a timely manner when abnormalities are detected. The overall goal of our work is to gradually implement all of these steps, shown in Figure 2, in the model based engineering toolset "Papyrus for Manufacturing".

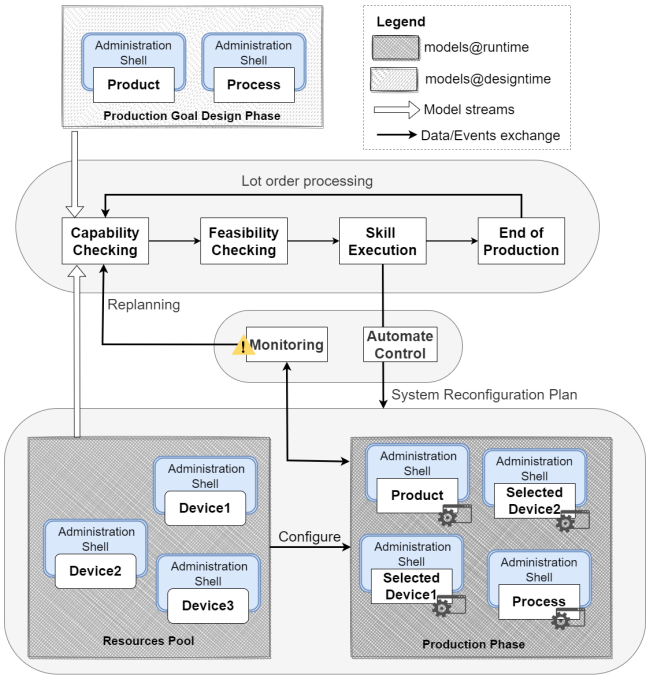

Figure 2: Capability-based reconfiguration approach

The rest of this section presents in details our capability checking module. A concrete example to describe this capability checking process, is how to select a device that can provide transporting capability from the alternative resources when an object needs to be moved in the production process. In order to simplify the presentation, we consider only design time models as input. Since runtime models will contain similar meta data to the ones of the design time models, the capability checking module will have a similar behavior when interacting with the two types of models.

3.1 Capability Checking Design Flow

As shown in Figure 3, the capability checking module interacts with the AAS models to set/get their semantics and then to trigger the capability matchmaking reasoner in order to compute the best

[1] http://www.opencaesar.io/oml/

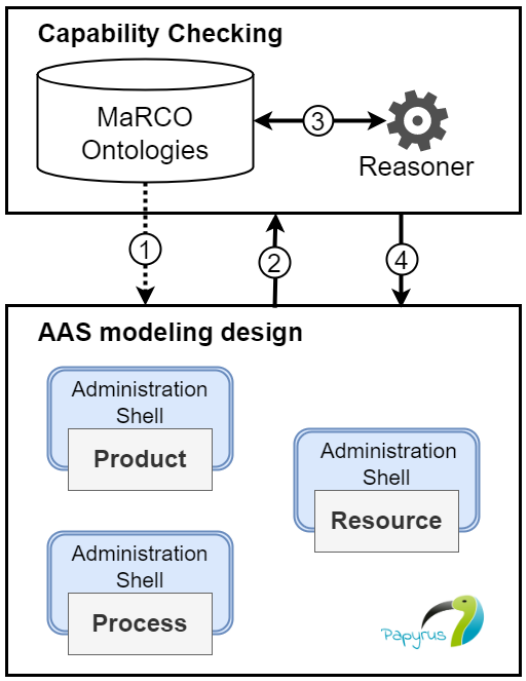

Figure 3: Capability checking architecture

resources matching the requirements of each production process. The four stages depicted in Figure 3 are:

(1) The designer annotate to the AAS models with semantic definitions (semanticIds) from the MaRCO ontologies.
(2) The designer automatically transforms the AAS models (Product, Process, Resources) into MaRCO individuals.
(3) With the input individuals, the automated reasoning engine is triggered to match the capabilities required by the process with the capabilities provided by the resources.
(4) Finally the capability checking module returns the match-making result to the designer.

Since there was an expert's commitment on ontology concepts, the ontology will not frequently change over time. Consequently, the first stage (OWL to UML profile conversion) only needs to be performed once, as long as the ontology concepts do not changed. The second, third and fourth stages will be repeated, whenever a PPR model update occurs. All the actions represented by the arrows shown in Figure 3 are automated, system architects only need to define and select the required production models.

3.2 Modeling Languages Mapping

Our platform relies on the alignment of models (in UML) and ontologies (in MaRCO). This alignment requires a comprehensive definition of mappings between concepts in the different languages used to express our models and ontologies.

3.2.1 General concepts mapping. In our approach, AAS models are UML models extended with an AAS-UML profile. To achieve the transformations between AAS and OWL models (stage 1 and 2 in Figure 3), we have used the OML adapter for Papyrus . We chose to use the OML adapter because it allows automatic conversion of the ontology concepts into a UML Profile, and its extraction from UML back to OWL. OML is a language to describe ontologies, where adapters for transformations from OML to UML, and UML to OWL are provided. In this context OML can be seen as an intermediate language to enable the conversions.

Table 1: General Concept Mapping

OWL	OML	UML Metaclass	UML Profile
Class	Aspect/ Concept	Abstract Class/ Class	Stereotype
Individual	Instance	Instance specification	Stereotype applicable element
Object Property	Relation Entity	Association/ Property	Stereotype attribute
Data Property	Property	Property	Stereotype attribute
Cardinality, MinCardinality, MaxCardinality	exactly min max	Multiplicity	Multiplicity

To pave the way to models alignment, we defined a mapping between OWL, OML, UML general concepts and the AAS-UML profile concepts (Table 1). As the output of the OML Adapter is a UML profile, we decided to add the OML and UML profile column to this table. The classes in OWL are represented as aspects and concepts in OML. The aspect refers to the abstract class, while the concept refers to the class. They are all transformed to stereotypes of a UML profile. An individual in OWL refers to an instance in OML and instance specification in UML, this represents a UML element to which a stereotype is applied to. The OWL object properties are represented in OML as relation entities and refer to associations or properties in UML. The OWL data properties refer to properties both in OML and UML. The object properties and data properties are transformed to the attributes of a stereotype.

3.2.2 AAS & MaRCO vocabularies mapping. In order to use OML for our MaRCO-specific semantics, the mappings between general concepts are not enough. Hence, we defined transformation rules between the vocabularies of MaRCO and AAS in Figure 4. MaRCO concepts are on the left, while AAS concepts are in the middle. As presented in Table 1, the OWL classes are transformed to stereotypes in a UML Profile. A subset of MaRCO concepts have been chosen, such as *Resource*, *ProductElement* and *Activity*, which are transformed to stereotypes that can be applied to the AAS models of resources, products and processes. The *Capabilities* or *ProcessTaxonomyDescriptions* stereotypes should be applied to AAS capabilities. Then the object properties *requiresProcessCapability* and *hasCapability* refer to the attributes of these stereotypes. And the value type of these attributes should be *Capabilities* or *ProcessTaxonomyDescriptions*. Each stereotype may contain two different types of attributes, one is the scalar properties such as weight or depth which refers to the data property parameter, the other is object

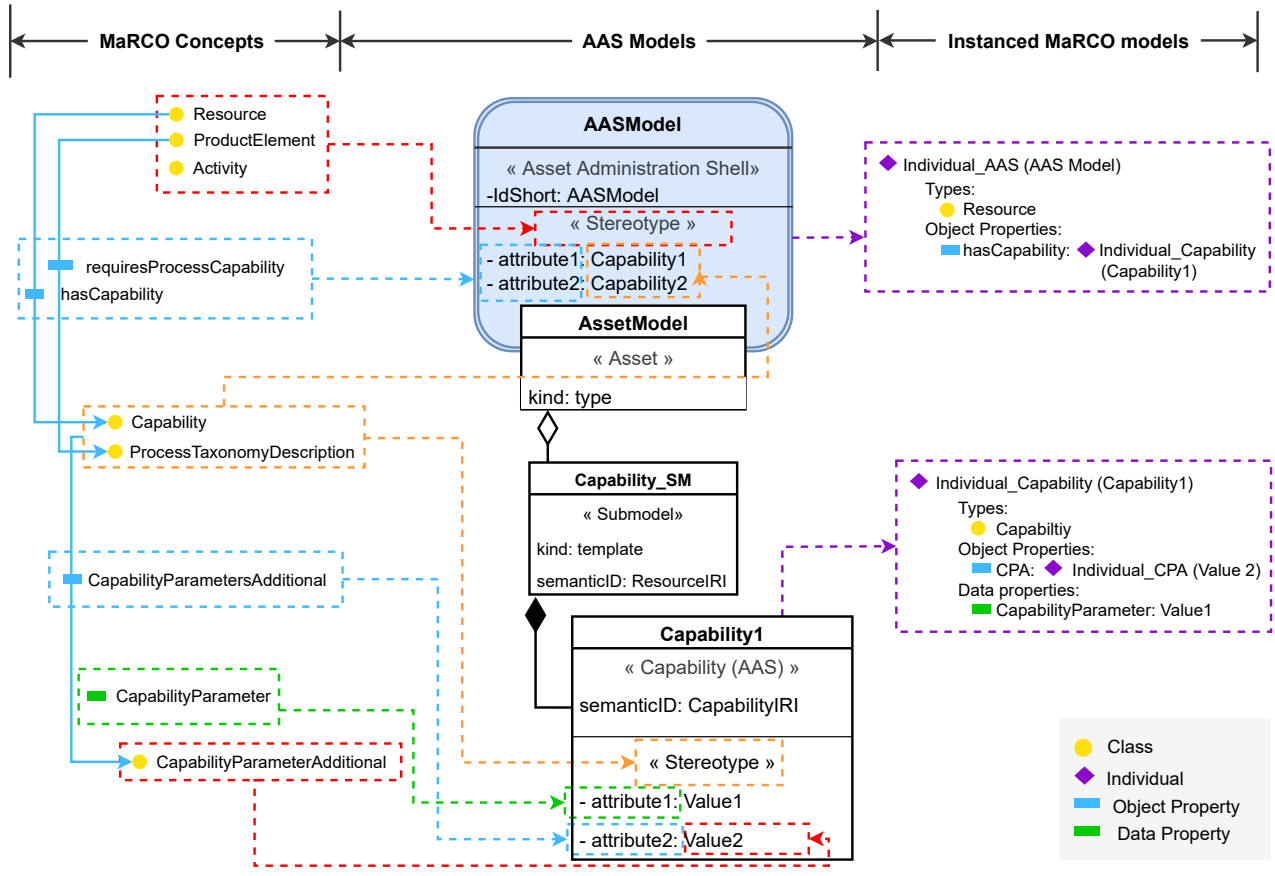

Figure 4: AAS Marco vocabularies mapping

properties that points to other stereotyped elements in the model package. By applying stereotype to an AAS model, the semanticId of the AAS model will be associated with the vocabulary IRI in MaRCO ontology.

Once we have annotated the AAS model content with the stereotypes coming from MaRCO ontology, these stereotype applicable elements will be regenerated back to OWL individuals. So an AAS class is transformed to an OWL individual, and its type is either a *Resource*, a *ProductElement* or an *Activity*. It may contain object properties *hasCapability* or *requiresProcessCapability* with the value of individual capabilities as defined in the AAS model.

4 IMPLEMENTATION

The capability checking implementation involves three different modules as shown in Figure 5: (1) the OML Adapter for the ontology concept conversions between different file natures, (2) the capability matchmaker for inferences, and (3) user interface module for launching capability checking requests and displaying the reasoning results in Papyrus4Manufacturing [19]. Both modules, OML Adapter and capability matchmaker, take MaRCO ontology as input. The final output of the entire capability checking process is the computed list of combinations of resources that can fulfill

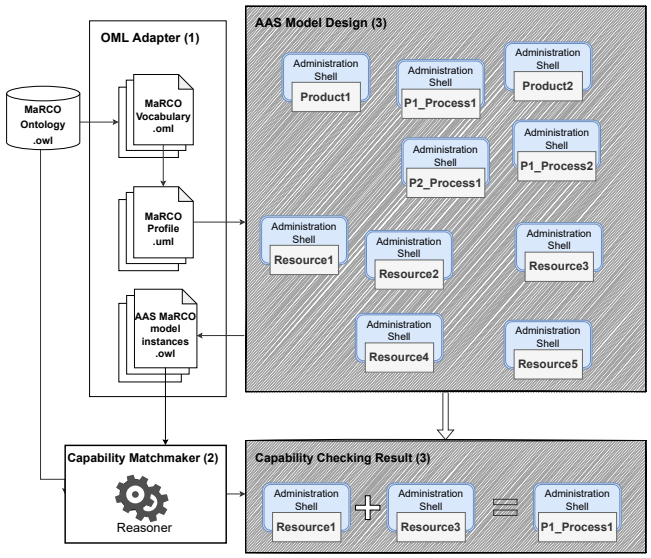

Figure 5: Technical architecture of capability checking

the capabilities required by the process to achieve the product production.

4.1 OML Adapter Module

Figure 6 illustrates the four steps applying the OML adapter:

(1) Choose a subset of OWL-based MaRCO ontology concepts and define corresponding OML Vocabularies.
(2) Convert the OML vocabularies to a UML profile.
(3) Apply the generated MaRCO UML profile to AAS models.
(4) Regenerate the specified AAS models to MaRCO compliant instances in an OWL file.

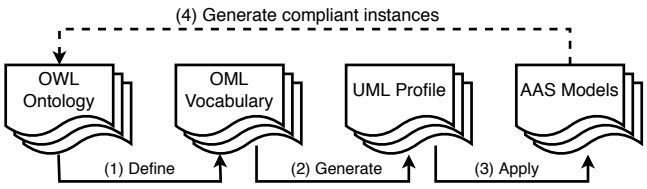

Figure 6: OML Adapter workflow

The first three steps correspond to the first stage introduced in Section 3.1, which enriches AAS models with semantic annotations in the manufacturing capability domain. Since there was experts' commitment on the ontology, once generated, the UML profile generated from the ontology can be reused for all the actions afterwards. The fourth step refers to the second stage in the capability checking architecture (Figure 3), that generates the MaRCO concept instances from the AAS system model for further inferences.

Here we will briefly introduce some concepts from the MaRCO ontology involved in this capability matchmaking process. The MaRCO ontology is composed of several distributed ontologies [18]. By using the OML Adapter, a subset of MaRCO vocabularies was transformed into a UML profile that can be applied to AAS models as stereotypes, including different sub-classes of the concepts appearing in 3.2.2. The capabilities are separated into simple capabilities like *Moving* and combined capabilities like *PickAndPlace*, and these capabilities have parameters to describe their characteristics. The combined capabilities are compositions of simple or other combined capabilities, these information are defined in the Capability Model ontology. The resource model stereotypes define different resource types, including atomic resources (*DeviceBlueprint* and *IndividualDevice*) and different resource combination types including *DeviceCombination*, and the combination at the *FactoryUnit* level. The concepts of *Product*,*Process* and a selection of *ProcessTaxonomyDescription* have been included in the UML profile as well.

4.2 Capability Matchmaker Module

The capability matchmaker is responsible for resource combination and combined capability computation, as well as the matchmaking reasoner which aligns the corresponding capabilities between production processes and resources. The implementation of this module reuses as much as possible other existing open-source projects. First of all, the MaRCO ontology and the associated SPARQL queries and SPIN rules come from the open-source MaRCO ontology[17]. The functionalities of ontology read and write is provided by Jena semantic web framework[2] and the SPARQL queries can be executed by Openllet reasoner[3]. As for the reasoning process of SPIN rules, it is realized by SpinAPI (provided by TopBraid[4]), which aims at encouraging the adoption of SPIN in the domain.

The pre-defined SPARQL queries update the capabilities for the individual devices and compute combined capabilities for the device combinations. The SPIN rules integrated in the Parameter Rule ontology will be executed in order to infer these novel capabilities' parameters. The matchmaking reasoner deals with the matching between capabilities required by the process and capabilities provided by the newly updated resource system. During this process, not only are the capabilities matched at the name level *has capability match*, but also the adaptations of the parameters *canBeImplementedWith* are computed. These reasoned relationships and inferred elements will be saved in a separate file.

4.3 User Interface Module

This user interface exists in the form of an Eclipse plug-in that ties the above two modules together and establishes a relationship with the model in the modeling environment. The usage scenario we envisage is shown in Figure 7. First, the user defines the production process in Papyrus4Manufacturing [19], and triggers the capability checking function through a right-click menu "Capability Checking", from which he/she can select the product for which the capability checking must be performed. This command sequentially invokes the OML Adapter, the capability matchmaker and the results retrieving module. After a series of processing, the results are returned to Papyrus4Manufacturing by a popup window, providing the user with a list of devices to choose from. Finally, the user selects a set of equipment combinations and then performs the feasibility checking (which is out of the scope of this paper, as said earlier).

The result retrieval aims to integrate and extract the results of ontology inferences, return them to the user, and save them for later use. The result of capability matchmaking shows the *DeviceBlueprints* that can realize the capability, however in our actual application, the production process is actually realized by the device instances (*IndividualDevices*). In this step, we need to find qualified device instances and device combinations through SPARQL queries, so we still need the help of ontology and SPARQL query processing tools previously mentioned. The results will be sorted out via a popup window for the user to choose from. The selected information will be included as input in feasibility checking which the second step of the capability-based engineering.

It is worth mentioning that, if we look back to the capability-based engineering approach in Figure 2, the capability checking in this article only involves the period from the design process stage to the period before it is put into production. It does not include the re-capability checking triggered by the monitoring process after the

[2]https://jena.apache.org/
[3]https://github.com/Galigator/openllet
[4]https://www.topbraid.org/spin/api/

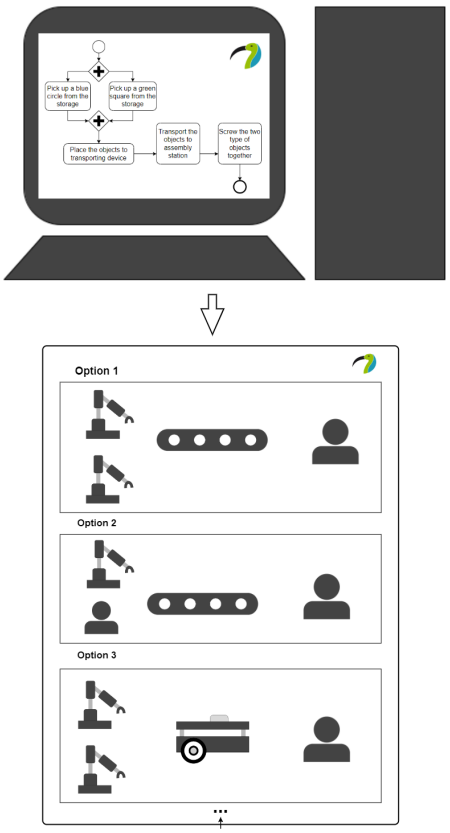

Figure 7: User interaction scenario

production line is put into production. This capability checking procedure is only the first part of our vision to implement a capability-based engineering digital twin system. Papyrus4Manufacturing provides an AAS (Asset Administration Shell) modeling environment for manufacturing where digital twins (AASs) of assets can be modeled and automatically deployed to BaSys[11]. The AASs communicate with physical devices via the OPC UA protocol. The robotic cell, considered in this paper, has already an OPC UA communication layer [26] which connects the devices with the AASs' executable BaSyx code. The resource combination options obtained in this step will be simulated and verified during the feasibility checking process. Finally, the selected resources should be deployed automatically.

5 ROBOTIC CELL USE CASE

A robotic cell (LocalSEA) use case is now presented to demonstrate the entire process of capability checking. The AAS modeling of this example has already been described in [14]. In this senario, a new product has been designed and the system architect want to configure the production line with the help of Papyrus4Manufacturing toolset. The production resources consist of two Niryo Neds, one conveyor belt, one TurtleBot3, two human workers, two storage units, and an assembly workstation. Niryo Ned is a robotic arm that includes a six-axis arm to realize *PickAndPlace*, a camera to realize *LocatingVisual*. The conveyor belt owns the capability *Transporting*. The TurtleBot3 Waffle is a mobile robot that can achieve *Transporting* capability as well. Ideally, a human could replace any type of device, with abilities including *PickAndPlaceFlexible*, *Transporting*, and *Hammering*.

Next, The AAS model of the product and its production process needs to be defined, including the information of the product and the manufacturing capabilities required by the process. The product defined in our robotic cell example is the assembly of two objects of different colors and shapes. Therefore, the corresponding production process (Figure 8) is as follows:

- Detect and grasp the two types of required pieces from two different storage unit in parallel and place them on the transporting device.
- Transport the required parts to the assembly area
- Complete the screw action

It is represented as a BPMN [27] process diagram. The capabilities required by this manufacturing process are: *PickAndPlaceFlexibles*, *Transporting* and *Hammering*.

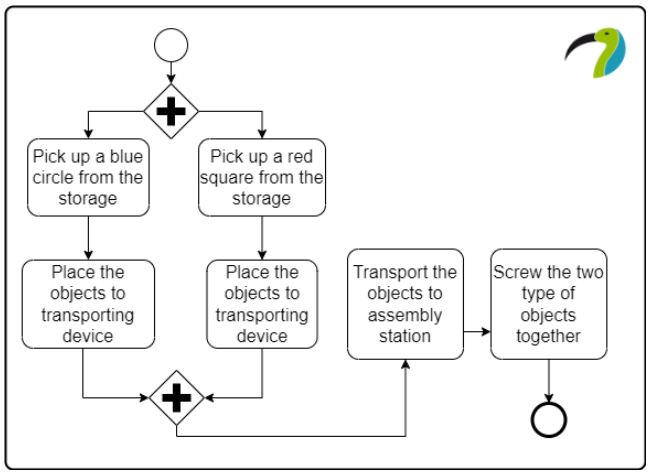

Figure 8: LocalSEA BPMN production process

During the design phase, the MaRCO Ontology profile is applied to the LocalSEA models, as presented in 3.2.2. Also, the AASs have applied stereotypes corresponding to the different types of *Resources* existing in MaRCO. Figure 9 is an example of different types of devices existing in LocalSEA and their attributes. The stereotype *DeviceBlueprint* is applied to "AASHumanOperator_type" contains the information about a human operator in LocalSEA. The capabilities mentioned above are attached to AAS capabilities owned by the "AASHumanOperator_type" as stereotypes. Alice, an instance of human operator type, is defined as an *IndividualDevice*, so the attribute *hasDeviceBlueprint* is set to "AASHumanOperator_type". And the last model shown in Figure 9 "AASNiryoNed1" is a *DeviceCombination*.

Once the user selects the product to produce, the rest of capability checking process are automated by a right-click command. Firstly, the OML Adapter is automatically called in order to transform the AAS models into MaRCO instances. The resulting *AASs.owl* file

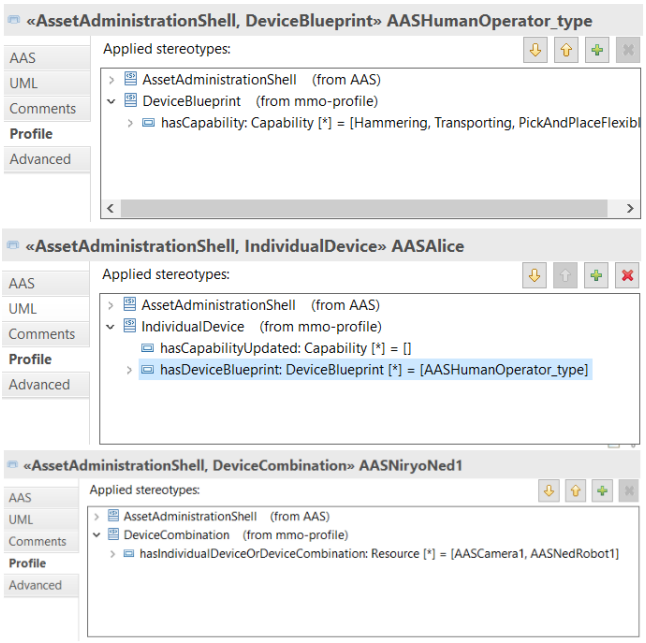

Figure 9: Different types of LocalSEA Resources

contains all the AAS model capability-related information. Then the capability matchmaker takes the resources and product descriptions as input in order to infer the matchmaking results. These inference results will be generated in the same folder as *AASs.owl* under the name of *matches.ttl*. Figure 11 shows the changing status of the required capability *Hammering* in the LocalSEA production process at different stages of capability checking. In the modeling environment, the corresponding stereotypes are applied to "AASProcess1" model. As shown at the upper part of the figure, the attribute of *matchmakingRequired* is set to true to trigger the matchmaking inference. This information is written to AASs.owl intact, as shown in the middle screenshot. The lower part of the figure shows the inferred information stored in *matches.ttl* after inference by the capability matchmaker. The figure shows the hammering process can be implemented with the human operator typed devices.

The capability checking results of the "AASProduct1" are grouped in a pop-up window shown in Figure 10. According to these results, *PickAndPlaceFlexible* can be implemented by NiryoNeds, *Transporting* can be done by TurtleBot or conveyor belt, and human operators can realize all the capabilities required in this process, which just matches our previous definition of LocalSEA devices. Through this result list, the user can select the production line combination to be further checked in the feasibility checking module.

6 CONCLUSIONS AND FUTURE WORKS

The work described in this paper takes part in a larger project aiming at designing and implementing an automated Industry 4.0 flexible plant supervision and control system based on MDE concepts within the "Papyrus for Manufacturing" toolset. Within such a system, capability checking is the phase that, given a client order for some produce, select plant resources able to fulfill the needs

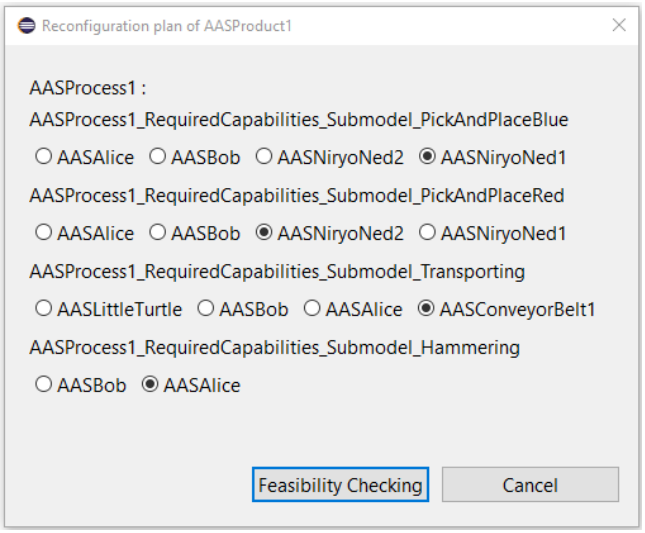

Figure 10: Capability checking result window

of the produce production plan. A matchmaking process between requirements of the plan and the plant resources automatically performs this selection. The main contribution of the paper is a new matchmaking algorithm that extends current syntactic-only matching with semantic matching based on information represented in the MaRCO manufacturing ontology.

This new algorithm has been fully implemented within the recently released "Papyrus for Manufacturing" platform, using a large set of tools from I4.0 standards (RAMI 4.0, Asset Administration Shells, etc.) to ontologies (MaRCO) and ontological query languages (SPARQL and SPIN), orchestrated through a set of models, UML profiles and model transformations (OML adapter), targeted to flexible production line management, that we have either developed or integrated in our platform. MDE concepts and tools have been used to integrate ontological tools in order to provide semantics and semantic interoperability among I4.0 concepts and assets, hence bringing an effective implementation of a semantic-based capability checking to flexible plant management.

The paper thoroughly explains the entire capability checking design flow that we have implemented. This flow uses model alignments to keep synchronized MDE-based models and an ontological representation of produces, production plans and plant resources in order to select the plant resources able to fulfill the production plan requirements through semantic-based ontological queries, and then get the results back into the MDE-based representation for further processing. We have demonstrated the effectiveness of this capability checking algorithm in Papyrus4Manufacturing on a robotic cell use case (LocalSEA).

The next steps and future work that we have undertaken for the implementation of our flexible plant management system concerns (1) the feasibility checking phase and then (2) the plant reconfiguration prior to (3) the supervision and control of the production plan execution. The feasibility checking extends the capability checking by taking into account the current contextual constraints and the actual time-dependent aspects of plan execution to generate

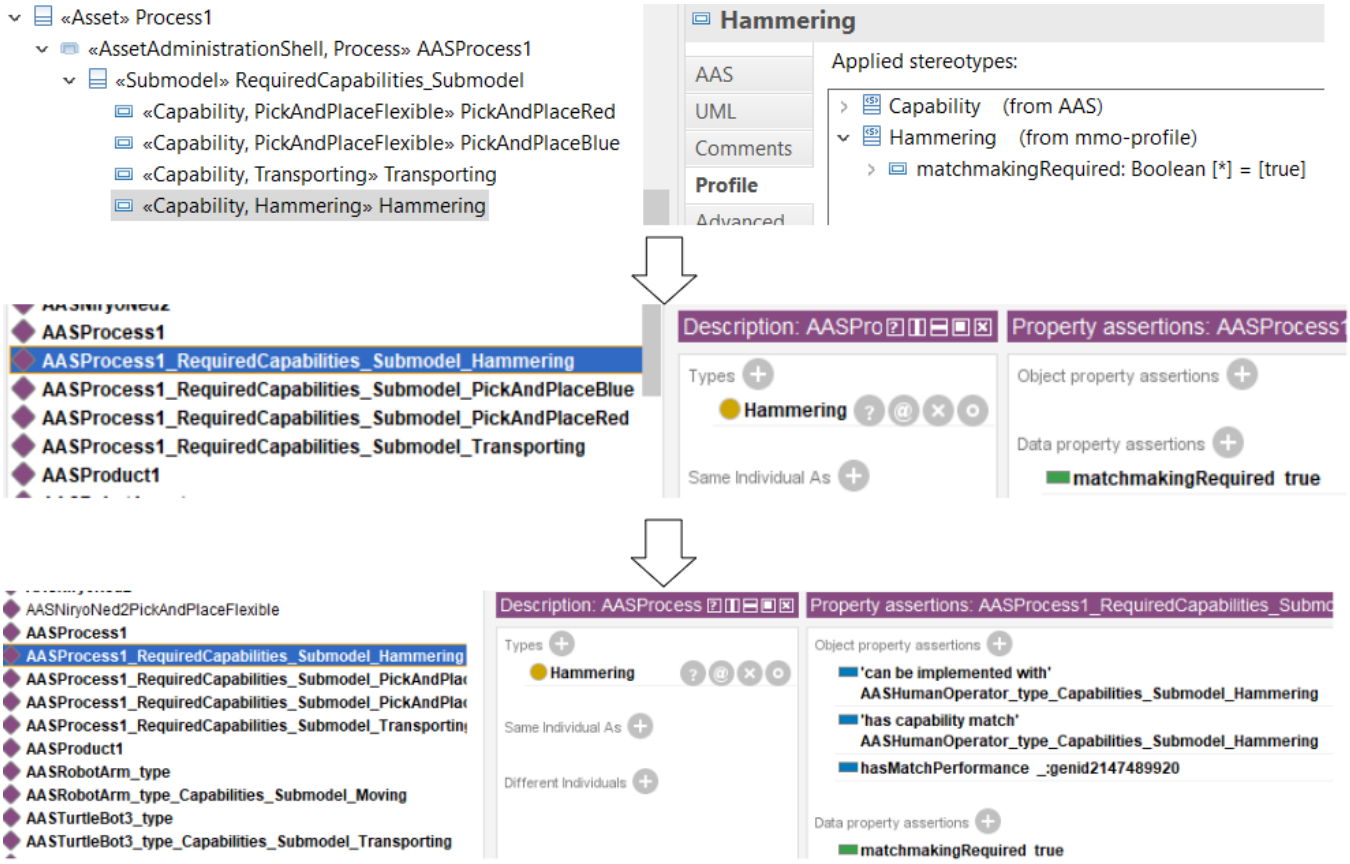

Figure 11: An AAS2MaRCO generation of a required capability

feasible plans for production. Feasibility checking will leverage the previously developed knowledge-based representation, but it will also need a time-related reasoning capability that may go as far as simulating the plan to ensure its actual feasibility. From the feasible plan and its selected resources, the system will then have to reconfigure the plant to prepare for the production *per se*, which will also need to be supervised to react to failures or abnormal events. When such events happen, the system may have to stop the production, revise the plan, adjust the plant configuration and restart the production. To implement this supervisory control phase as well as the simulation part of feasibility checking, our system will require a comprehensive usage of models@runtime and digital twins representation of the plant to enable effective plan execution and runtime adaptations.

ACKNOWLEDGMENTS

REFERENCES

[1] 2010. ISA95, Enterprise-Control System integration- ISA. https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95

[2] Doug Migliori (CloudBlue) Anto Budiardjo (Padi). 2021. Digital Twin System Interoperability Framework. *Digital twin consortium whitepaper* (2021). https://www.digitaltwinconsortium.org/pdf/Digital-Twin-System-Interoperability-Framework-12072021.pdf

[3] Barbara Rita Barricelli, Elena Casiraghi, and Daniela Fogli. 2019. A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications. *IEEE Access* 7 (2019), 167653–167671. https://doi.org/10.1109/ACCESS.2019.2953499

[4] Jean Bézivin. 2005. Model Driven Engineering: An Emerging Technical Space, Vol. 4143. 36–64. https://doi.org/10.1007/11877028_2

[5] International Electrotechnical Commission. [n.d.]. Common data dictionary (CDD). https://cdd.iec.ch/cdd/iec61360/iec61360.nsf/TreeFrameset?OpenFrameSet

[6] Jan deMeer. 2021. Semantics for I4.0 Smart Manufacturing. In *INFORMATIK 2020*, Ralf H. Reussner, Anne Koziolek, and Robert Heinrich (Eds.). Gesellschaft für Informatik, Bonn, 289–298. https://doi.org/10.18420/inf2020_27

[7] ECLASS. [n.d.]. Le standard ECLASS. https://www.eclass.eu/fr/standard.html

[8] Maged Elaasar. 2018. Definition of Modeling vs. Programming Languages. In *Leveraging Applications of Formal Methods, Verification and Validation. Modeling*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer International Publishing, Cham, 35–51.

[9] Dominique Ernadote. 2017. Ontology-Based Pattern for System Engineering. In *Proceedings of the ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems* (Austin, Texas) *(MODELS '17)*. IEEE Press, 248–258. https://doi.org/10.1109/MODELS.2017.4

[10] Shannon Flumerfelt, Katherine G. Schwartz, Dimitri Mavris, and Simon Briceno. [n.d.]. *Complex Systems Engineering: Theory and Practice.* American Institute of Aeronautics and Astronautics, Inc.

[11] Eclipse Foundation. 2022. *BaSyx: the open source Industry 4.0 middleware.* Retrieved May 18, 2022 from https://www.eclipse.org/basyx/

[12] Roland Heidel, Michael Hoffmeister, Martin Hankel, and Udo Döbrich. [n.d.]. *The Reference Architecture Model RAMI 4.0 and the Industrie 4.0 component.*

[13] Yining Huang, Saadia Dhouib, and Jacques Malenfant. 2021. AAS Capability-Based Operation and Engineering of Flexible Production Lines. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 01–04. https://doi.org/10.1109/ETFA45728.2021.9613435

[14] Yining Huang, Saadia Dhouib, and Jacques Malenfant. 2021. An AAS Modeling Tool for Capability-Based Engineering of Flexible Production Lines. In *IECON 2021 - 47th Annual Conference of the IEEE Industrial Electronics Society*. IEEE,

Toronto, Canada, 1–6. https://doi.org/10.1109/IECON48115.2021.9589329

[15] Plattform I4.0. [n.d.]. Describing capabilities of Industrie 4.0 components. https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Capabilities_Industrie40_Components.html.

[16] Plattform I4.0. [n.d.]. Details of the Asset Administration Shell - Part 1. https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html.

[17] Eeva Järvenpää, Otto Hylli, Niko Siltala, and Minna Lanz. 2018. Utilizing SPIN Rules to Infer the Parameters for Combined Capabilities of Aggregated Manufacturing Resources. *IFAC-PapersOnLine* 51, 11 (2018), 84–89. https://doi.org/10.1016/j.ifacol.2018.08.239 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[18] Eeva Järvenpää, Niko Siltala, Otto Hylli, and Minna Lanz. 2021. Capability matchmaking software for rapid production system design and reconfiguration planning. *Procedia CIRP* 97 (2021), 435–440. https://doi.org/10.1016/j.procir.2020.05.264 8th CIRP Conference of Assembly Technology and Systems.

[19] CEA List. 2022. *Papyrus for Manufacturing Model Driven Workbench.* Retrieved May 18, 2022 from https://www.eclipse.org/papyrus/components/manufacturing/

[20] Gaëlle Lortal, Saadia Dhouib, and Sébastien Gérard. 2010. Integrating Ontological Domain Knowledge into a Robotic DSL. In *Proceedings of the 2010 International Conference on Models in Software Engineering* (Oslo, Norway) *(MODELS'10)*. Springer-Verlag, Berlin, Heidelberg, 401–414.

[21] Azad M. Madni, Carla C. Madni, and Scott D. Lucero. 2019. Leveraging Digital Twin Technology in Model-Based Systems Engineering. *Systems* 7, 1 (2019). https://doi.org/10.3390/systems7010007

[22] Jürg Meierhofer, Lukas Schweiger, Jinzhi Lu, Simon Züst, Shaun West, Oliver Stoll, and Dimitris Kiritsis. 2021. Digital Twin-Enabled Decision Support Services in Industrial Ecosystems. *Applied Sciences* 11, 23 (2021). https://doi.org/10.3390/app112311418

[23] Anant Mital, Niko Siltala, Eeva Järvenpää, and Minna Lanz. 2019. Web-based solution to automate capability matchmaking for rapid system design and reconfiguration. *Procedia CIRP* 81 (2019), 288–293. https://doi.org/10.1016/j.procir.2019.03.050 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.

[24] Jeff Morgan, Mark Halton, Yuansong Qiao, and John G. Breslin. 2021. Industry 4.0 smart reconfigurable manufacturing machines. *Journal of Manufacturing Systems* 59 (2021), 481–506. https://doi.org/10.1016/j.jmsy.2021.03.001

[25] Kamran Munir and M. Sheraz Anjum. 2018. The use of ontologies for effective knowledge modelling and information retrieval. *Applied Computing and Informatics* 14, 2 (2018), 116–126. https://doi.org/10.1016/j.aci.2017.07.003

[26] Quang-Duy Nguyen, Fadwa TMAR, Yining Huang, and Saadia Dhouib. 2022. Early lessons learned from the development of a local OPC UA-based robotic testbed for research. In *The 31st IEEE International Symposium on Industrial Electronics (2022 IEEE International Symposium on Industrial Electronics (ISIE))*. Anchorage, Alaska, United States, 1–4. https://hal-cea.archives-ouvertes.fr/cea-03610950

[27] OMG. 2010. Business Process Model And Notation v2.0. https://www.omg.org/spec/BPMN/2.0/

[28] Marie Platenius-Mohr, Somayeh Malakuti, Sten Grüner, and Thomas Goldschmidt. 2019. Interoperable Digital Twins in IIoT Systems by Transformation of Information Models: A Case Study with Asset Administration Shell. In *Proceedings of the 9th International Conference on the Internet of Things* (Bilbao, Spain) *(IoT 2019)*. Association for Computing Machinery, New York, NY, USA, Article 2, 8 pages. https://doi.org/10.1145/3365871.3365873

[29] Eric Prud'hommeaux and Andy Seaborne. 2008. SPARQL Query Language for RDF. https://www.w3.org/TR/rdf-sparql-query/

[30] Thomas H. J. Uhlemann, Christian Lehmann, and Rolf Steinhilper. [n.d.]. The Digital Twin: Realizing the Cyber-Physical Production System for Industry 4.0. ([n. d.]).

[31] Birgit Vogel-Heuser, Felix Ocker, Iris Weiß, Robert Mieth, and Frederik Mann. 2021. Potential for combining semantics and data analysis in the context of digital twins. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379, 2207 (2021), 20200368. https://doi.org/10.1098/rsta.2020.0368 arXiv:https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2020.0368