Semantic Interoperability of Digital Twins: Ontology-based Capability Checking in AAS Modeling Framework

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

Semantic Interoperability of Digital Twins: Ontology-based Capability Checking in AAS Modeling Framework

Yining Huang
Université Paris-Saclay
CEA, List
F-91120, Palaiseau, France
yining.huang@cea.fr

Saadia Dhouib
Université Paris-Saclay
CEA, List
F-91120, Palaiseau, France
saadia.dhouib@cea.fr

Luis Palacios Medinacelli
Université Paris-Saclay
CEA, List
F-91120, Palaiseau, France
luis.palacios@cea.fr

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

***Abstract*—Industry 4.0 currently prepares a major shift towards extreme flexibility into production lines management. Digital Twins are one of the key enabling technologies for Industry 4.0. However, the interoperability gap among digital representation of Industry 4.0 assets is still one of the obstacles to the development and adoption of digital twins. If the Asset Administration Shell (AAS), the standard proposed to represent the I4.0 components, caters for syntactic interoperability, a more semantic kind of interoperability is deeply needed to develop flexible and adaptable production lines. In our work, we overcome the limitation of current syntactic-only resource matching algorithms by implementing semantic interoperability based on ontologies *i.e.*, by transforming AAS-based plant models into MaRCO (Manufacturing Resource Capability Ontology) instances and then query the expanded ontology to find the needed resources. This article presents this ontology-based approach as the first step towards the design and implementation of an automated I4.0 flexible plant supervision and control system based on model-driven engineering (MDE) within the "Papyrus for Manufacturing" toolset. We show how an MDE approach can aggregate around digital twin modeling tools from the Papyrus platform both I4.0 technologies and AI (Knowledge Representation and Reasoning) tools. Our platform aligns modeling and ontological elements to get the best of both worlds. This method has two main advantages: (1) to provide semantic descriptions for digital twin models, (2) to complement model-driven engineering tools with automated reasoning. This paper showcases this approach through a robotic cell use case.**

***Index Terms*—Digital Twins, Industry 4.0, Asset Administration Shell, Semantic Interoperability, Ontology, Model-Based Engineering**

I. Introduction

The future industry will be dominated by highly autonomous and adaptive intelligent manufacturing systems, with great flexibility to deal with the reconfiguration of production lines and fully automated processes. Safe and reliable production line management in the vision of Industry 4.0 will primarily rely on digital twin [1] technology. The automated process of digital twin production line 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.

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 equipment represents major challenges. 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 used at runtime, (2) interoperability standards, such as the Asset Administration Shell (AAS) [2] promoted by I4.0 as a standard interface to digitally represent all of I4.0 elements as well as (3) capability-based engineering (CBE) [3] which aims at representing and reasoning about production activities.

Although the AAS standard provides syntactic interoperability for cross-vendor assets, it leaves the major issue of semantic interoperability unresolved. Many researchers have realized this semantic gap as a major shortcoming of AAS and studied on to better describe assets, to propose a solution rather by referring to ontologies [4] or to conduct model transformations [5]. However, no solution towards enabling comprehensive semantic interoperability of asset administration shells has been shown yet. An ontology can define a semantic model of domain knowledge, and consist of inference rules [6]. This article follows up our previous work [7], where we mainly introduced how to annotate semantic meaning to AAS models, including a specific mapping between the MaRCO [8] ontology concepts and AAS meta-models. The focus of this article is to demonstrate our ontology-based implementation of the automatic AAS capability checking process in "Papyrus for Manufacturing" toolset [3]. The resulting process enables

the semantic interoperability of AAS models, hence a concrete realization of capability-based engineering. Our approach uses model transformations to align model-based and ontology-based industrial asset representations. By using this tool, we can obtain candidate resource combinations for production line reconfiguration from a well-defined AAS product model. A robotic cell use case will demonstrate the resulting methods and tools.

This article is organized as follows, Section II presents the backgrounds and related works. Section III introduces our capability checking design. Section IV focuses on the implementation of the capability matchmaking process using AAS models. Section V showcases our approach on a matchmaking example for a robotic cell use case. Finally, Section VI concludes with future works.

II. Background & Related Works

A. Digital Twins in Industry 4.0 & CBE

The current convergence of IT technologies, from the web of totally interconnected business information systems among purchasers and providers operating 24/24 to the Internet of things and sensor networks feeding in plant floor information in real time, enables a full automation of flexible production process management from order to delivery. More specifically, laying down the technical foundation for Industry 4.0, the current effort to organize this new wave of industry automation, the Digital twin technology brings new potential for managing and controlling increasingly complex systems. In parallel, the Model-Driven Engineering (MDE) paradigm [9] is now imposing itself as an essential direction in the field of digital twins. The unique characteristics of connectivity and extensibility between models bring dawn to the design and maintenance of digital twin systems.

However, when a unified description is lacking, digital twin models cannot achieve interoperability and scalability when designed and implemented independently. This problem becomes more prominent as the system scales. In this context, some widely accepted standards in the manufacturing domain were born, such as the Reference Architecture Model for Industry 4.0 (RAMI4.0) [10] and the Asset Administration Shell (AAS) [2]. RAMI4.0 is a reference model that can help manufacturing companies get through vertical integration, that is, the connection from Level 0 to Level 4 of the ISA-95 [11] manufacturing pyramid. It permits tracking data flow from production equipment, production execution, and production planning to enterprise business operation management. AAS provides a unified architectural framework and standardized interface (I4.0) system for Industry 4.0 [12] presented a method to map the AAS meta-model to RDF, but it does not provide domain specific semantics for the system being modeled.

The concept of a capability-based approach to flexible production line engineering was first published by Plattform Industrie 4.0, which describes the concept and its operational implementation guidelines. The term "capability" is defined in the article [13] as an abstract description of the capabilities of a productive resource. In contrast, the ability to achieve a particular effect depends on the "skill" of the asset. Capability-based engineering aims to deploy resources dynamically rather than directly specifying the actual production participants. By defining the capabilities required for the product's production process and letting the automated production line management system find the resources and implement the process to achieve the reliability of the digital twin production system. We have further refined the three critical steps of continuous capability-based engineering capability checking, feasibility checking & skill execution in [14].

B. Semantic Interoperability in Manufacturing

In the previous subsection, we introduced the significance of interoperability in digital twin production systems and some standards that provide syntactic interoperability for production systems. Semantic interoperability has long been recognized as a significant issue in manufacturing systems. This subsection introduces the problem and then elicits several related works that will be reused in our approach.

A white paper published by the Digital Twin Alliance on the system interoperability framework for digital twins [15] introduces seven interoperability concepts that form the design considerations needed to make systems interoperable at scale. deMeer [16] presents the definition of semantic interoperability in the context of Industry 4.0 and Smart Manufacturing. Meierhofer *et al.* [17] articulate new concepts of value creation through the use of digital twin decision support services in industrial service ecosystems, and discusses the implementation of hybrid semantic modeling and model-based systems engineering. In [18], Weser *et al.* present a manufacturing resources abilities ontology (C4I) formally describing capabilities using Semantic Web technologies.

Based on the above work and many other unmentioned articles, using ontologies to address semantic interoperability emerges as a common solution in the field. Our idea is to combine ontology-based knowledge representation with AAS digital twins to achieve semantic interoperability between digital twins, enabling capability-based engineering. To achieve this, we rely on two former works, MaRCO [19] providing capability-related ontologies for manufacturing systems and OML Adapter [20] providing the basis for transformations from OWL ontologies to OML and UML models. The OWL-based Manufacturing Resource Capability Ontology (MaRCO) describes the capabilities of manufacturing resources. MaRCO supports the representation of different resources and the capabilities in manufacturing, hence a good candidate for capability-based engineering. In addition, MaRCO is also provided as a complete capability matchmaking web service [21]. The OML (Ontological Modelling Language) is an ontology description language inspired by OWL and SWRL (Semantic Web Rule Language). It has a Java API and integration with useful tools such as OML Adapter provided by the openCAESAR project. In this context, Medinacelli *et al.* [22] focus on using and integrating ontologies and standardized vocabularies. Meanwhile, in our previous work [7], we have

shown how to assign semantics meanings to AAS models with MaRCO ontology concepts and define the mapping rules to realize the round way transformation. The remainder of this paper will focus on how an AAS model with ontology semantics can automate the capability checking process.

III. Ontology-Based Capability Checking

Figure 1 shows the whole process of the capability-based engineering approach that we propose. 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 or 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 ISA95 [11]. 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 "AASs" (or digital twins) 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 currently 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. The supervisor deploys the selected resource pool models according to the 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.

A concrete example to describe this capability checking process shows 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 the scope of this paper, we consider only design time models. Since runtime models will contain similar meta data to the ones of the design time models, the capability checking feature will have a similar behavior when interacting with the two types of models.

As shown in Figure 2, the capability checking module interacts with AAS models to set/get their semantics and then to trigger the capability matchmaking reasoner in order to compute the best resources matching the requirements of each production process. The four stages depicted in Figure 2 are:

1) The designer annotate the AAS models with semantic definitions (semanticIds) from the ontology.
2) The designer triggers the automatically transformation of the AAS models (Product, Process, Resources) into ontology compliant individuals.

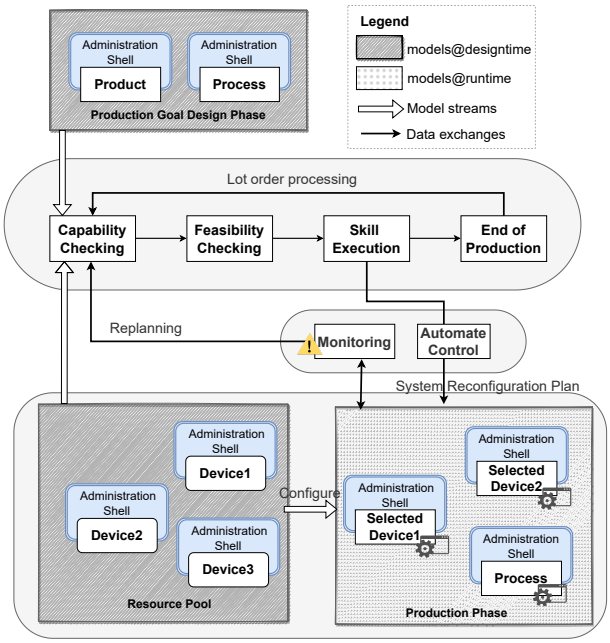

Fig. 1. Capability-based reconfiguration approach

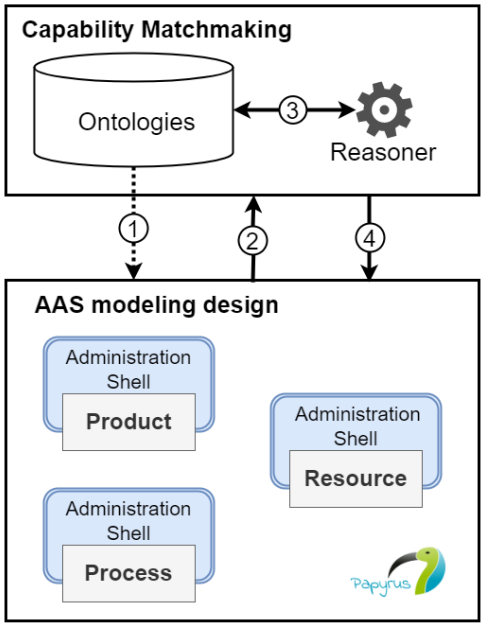

Fig. 2. Capability checking architecture

3) With the input individuals, the automated reasoning engine matches the capabilities required by the process with the capabilities provided by the resources.
4) Finally the capability checking module returns the matchmaking result to the designer.

Since there was a comprehensive expert investment in its design, the ontology will not frequently change over time but to add new resources and drop decommissioned ones. Consequently, the first stage (ontology to UML profile conversion

part) only needs to be performed once, as long as the ontology concepts do not change. 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 2 are automated, system architects only need to define and select the required production models.

IV. Implementation

When we implement the capability checking architecture mentioned in Section III, the numbers used in Figure 3 represent the implementation process of their tagged stage in Figure 2. This entire capability checking feature is developed as an Eclipse plug-in bundled with Papyrus4Manufacturing [23].

Our capability checking implementation involves three different modules: (A) the model transformation module for the ontology concept conversions between different file natures, (B) the capability matchmaker module for inferences, and (C) user interface module for launching capability checking requests and displaying the reasoning results in Papyrus4Manufacturing. To implement the above functions, we have selected two well-established jobs. One is OML Adapter, which is used to convert OWL to UML profile. The other is MaRCO ontology, on the one hand, because the description of capabilities in manufacturing perfectly suits our needs. On the other hand, it also provides complete inference rules for capability matchmaking.

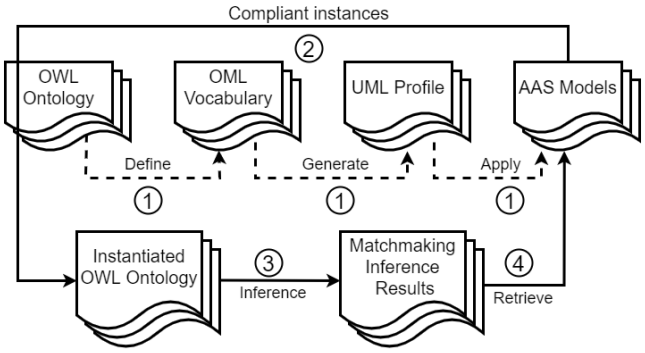

Fig. 3. Capability checking implementation workflow

A. Model Transformation Module

The model transformation module provides a round-way transformation between OWL ontologies and UML models. The three dotted steps in Figure 3 correspond to the first stage introduced in Section III, which enriches AAS models with semantic annotations in the manufacturing capability domain. As mentioned earlier, once generated from the ontology, this UML profile can be reused for all forthcoming actions.

The OML adapter takes care of the transformation from OML vocabularies to UML profiles. However, the MaRCO ontology was initially described in OWL format. To better use the existing works, we need to first define OML vocabularies referencing the original OWL ontology.

After confirming the definition of OML vocabularies and the corresponding relationship between these concepts and UML meta-model, we can obtain the MaRCO UML Profile through OML adapter. 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 [19]. By using the OML Adapter, a subset of MaRCO vocabularies is transformed into a UML profile that can be applied to AAS models as stereotypes, including different sub-classes of the concepts. 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*) as well as 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.

The MaRCO concepts in the generated UML profile are applied as stereotypes to the AAS models. The concrete mapping rules are described in [7]. The designer improves the AAS model based on the device properties and capabilities provided in the configuration file. The system designer should refine the stereotyped AAS models based on the properties of the equipments and capabilities. There is a semanticID concept for the submodel and submodel element of an AAS model. It is used to refer to the semantic meaning of this element. So when we assign a MaRCO concept to an AAS element, the semanticID should refer to the IRI of this concept in the ontology.

The second step refers to the second stage in the capability checking architecture (Figure 2), that generates the MaRCO concept instances from the AAS system model for further inferences. Based on the APIs provided by the org.eclipse.uml2.uml and org.semanticweb.owlapi packages, we developed a convertor from stereotyped UML models to Owl individuals. All AAS models and the information stored in the stereotypes that come from the MaRCO profile will be converted as OWL individuals that conform to the MaRCO ontology.

B. 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 [8]. The functionalities of ontology read and write are provided by Jena semantic web framework and the SPARQL queries can be

executed by Openllet reasoner. As for the reasoning process of SPIN rules, it is realized by SpinAPI (provided by TopBraid), 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 are 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 *can be implemented with* are computed. These reasoned relationships and inferred elements are saved in a separate file.

C. User Interface Module

This user interface 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 4. First, the user defines the production process in Papyrus4Manufacturing, 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).

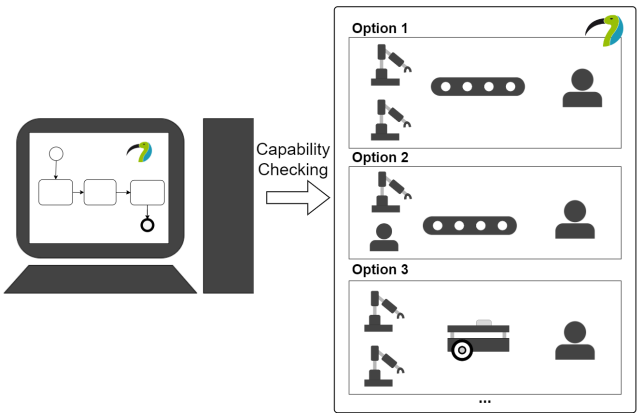

Fig. 4. User interaction scenario

The result retrieval aims to integrate and extract the results of ontology inferences, return them to the user, and save them for later use. We defined a SPARQL query (Figure 5) to automatically extract information from newly reasoned relationships. We want to select the equipment (either an individual or a combined device) that can provide the capabilities required for the production process through the query (Line 1). Via Lines 2-4, it is possible to select all processes participating in the capability checking. Lines 6 and 7 select the capabilities required for the aforementioned production processes. The eighth line finds the devices capable of implementing the required capabilities. The capability matchmaking results show the *DeviceBlueprints* or *DeviceCombinations* the that can realize the capability. However, in our application, the production process is realized by the device instances (*IndividualDevices*). So when the result is a *DeviceBlueprint*, we will find all available *IndividualDevices* belonging to it (Line 9-11). The results are sorted out via a popup window for the users to choose. And the inferenced information is again connected to the AAS digital twin models. The selected information can then be included as input for a feasibility checking or device deployment step coming next.

SPARQL query:

```
SELECT distinct ?process ?requirement ?required ?match ?deviceBlueprint (1)
?deviceCombination ?individualDevice
WHERE {
   ?activityCls rdfs:subClassOf+ pm:Activity .                          (2)
   ?process rdf:type ?activityCls .                                     (3)
   filter not exists { ?process mmo:ignoreProcess true } .              (4)
   ?process mmo:hasMatchPerformance ?performance .                      (5)
   ?process pm:requiresProcessCapability ?requirement .                 (6)
   ?requirement pm:matchmakingRequired ?required .                      (7)
   optional {
      ?requirement mmo:canBeImplementedWith ?match .                    (8)
      optional {
            ?deviceBlueprint rm:hasCapability ?match .                  (9)
            ?individualDevice rm:hasDeviceBlueprint ?deviceBlueprint .  (10)
         }
         optional {
            ?deviceCombination rm:hasCalculatedCapability ?match .      (11)
         }
   }
}
```

Fig. 5. SPARQL query for result extraction

V. 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 [3]. In this scenario, a new product has been designed and the system architect wants 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 is defined, including the information on the product and the manufacturing capabilities required by the process. The product defined in this example is the assembly of two objects of different colors and shapes. Therefore, the corresponding production process (Figure 6) is as follows:

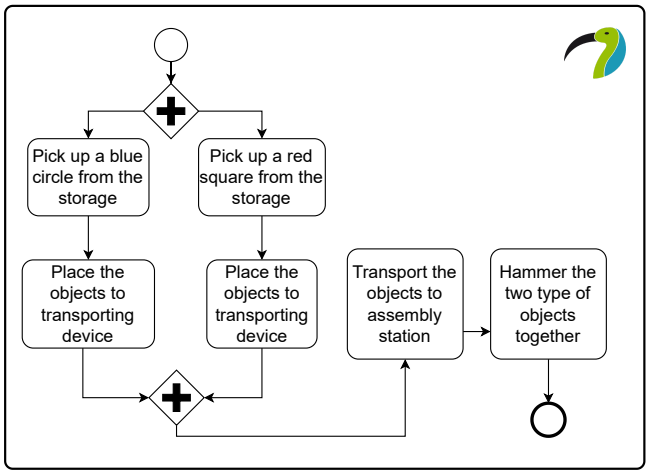

Fig. 6. LocalSEA BPMN production process

- 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 [24] process diagram. The capabilities required by this manufacturing process are: *PickAndPlaceFlexibles*, *Transporting* and *Hammering*.

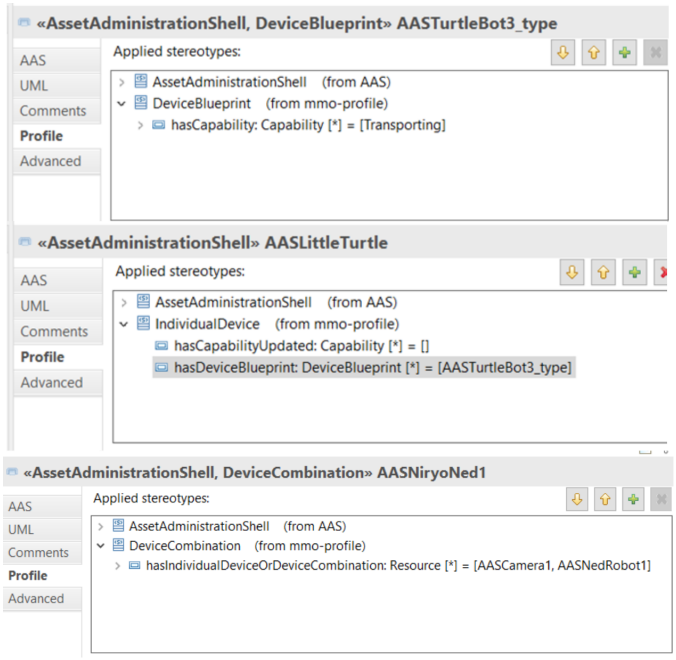

Fig. 7. Different types of LocalSEA Resources

During the design phase, the MaRCO Ontology profile is applied to the LocalSEA models. Also, the AASs have applied stereotypes corresponding to the different types of *Resources* existing in MaRCO. Figure 7 is an example of different types of devices existing in LocalSEA and their attributes. The stereotype *DeviceBlueprint* is applied to "AAS-TurtleBot3_type" contains the information about a Turtlebot3 robot in LocalSEA. The capabilitiy *Transporting* mentioned above are attached to AAS capabilities owned by the "AAS-TurtleBot3_type" as stereotypes. "LittleTurtle", an instance of Turtlebot3, is defined as an *IndividualDevice*, so the attribute *hasDeviceBlueprint* is set to "AASTurtleBot3_type". And the last model shown in Figure 7 "AASNiryoNed1" is a *Device-Combination*, which is a composite resource of a camera and a robot arm.

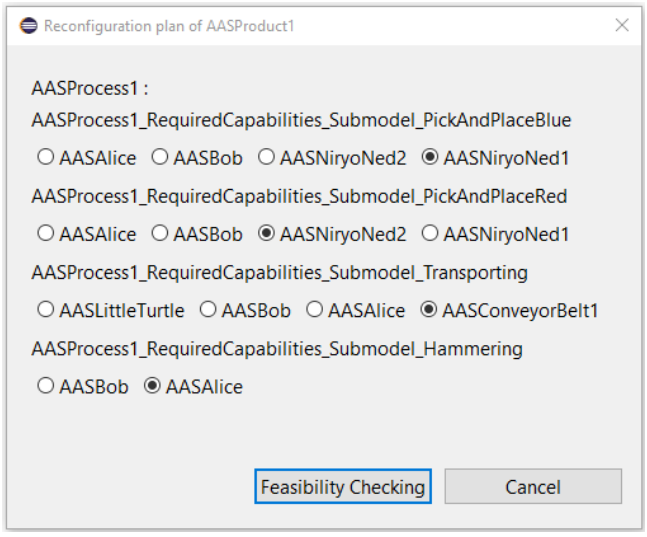

Fig. 8. Capability checking result window

Once the user selects the product to produce, the rest of capability checking process is fully automated and triggered by a right-click command. Firstly, the OML Adapter is called to transform the AAS models into MaRCO instances. The resulting *AASs.owl* file contains all the AAS model capability-related information. Then the capability matchmaker takes the resources and product descriptions as input to infer the matchmaking results. These inference results are generated in the same folder as *AASs.owl* under the name of *matches.ttl*. Figure 9 shows the changing status of the required capability *Transporting* 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. All the information defined in the process model is written to AASs.owl, 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 transporting process has capability match with the human operator typed devices, ConveyorBelt typed devices or a human operator. But it can only be implemented with TurtleBot3 or a human operator, because the parameters of the conveyor belt in this case doesn't fit the requirement.

The capability checking results of the "AASProduct1" are grouped in a pop-up window shown in Figure 8. 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.

VI. Conclusion

Our work takes part in a larger project aiming at designing and implementing an automated Industry 4.0 flexible plant digital twin within the "Papyrus4Manufacturing" toolset. A matchmaking process between requirements of the plan and the plant resources automatically performs these latters selection for production. The main contribution of the paper is a new matchmaking functionality that extends current syntactic-only matching with semantic matching based on information represented in the MaRCO manufacturing ontology.

This new feature has been fully implemented within the recently released "Papyrus4Manufacturing" platform, using a large set of tools from I4.0 standards (RAMI 4.0, AAS, etc.) to ontologies (MaRCO) and their query languages (SPARQL and SPIN), orchestrated through a set of models, UML profiles and model transformations (OML adapter) that we have either developed or integrated in our platform. We have demonstrated the effectiveness of this capability checking functionality in Papyrus4Manufacturing on a robotic cell use case (LocalSEA).

The future work, that we have already undertaken for 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 feasible plans for production. 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 firstly to react to failures or abnormal events but also to perform online machine learning and dynamic process optimization. When such events happen or optimized steps must be put in place, the system may have to stop the production, revise the plan, adjust the plant configuration and resume the production. To implement this supervisory control phase as well as the simulation part of feasibility checking. Our short-term goal concerns about generating data streams and dynamic model reasoning to enable runtime supervision. The system will require a comprehensive usage of models@runtime and digital twins representation of the plant and real-time data collection for effective plan execution.

Acknowledgment

This work has been partially funded by the European Union's Horizon 2020 program via Project GA Nr. 952003 AI REGIO.

References

[1] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," *IEEE Access*, vol. 7, pp. 167 653–167 671, 2019.

[2] P. I4.0, "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.

[3] Y. Huang, S. Dhouib, and J. Malenfant, "An AAS Modeling Tool for Capability-Based Engineering of Flexible Production Lines," in *IECON 2021 - 47th Annual Conference of the IEEE Industrial Electronics Society*. Toronto, Canada: IEEE, Oct. 2021, pp. 1–6. [Online]. Available: https://hal.sorbonne-universite.fr/hal-03476685

[4] B. Vogel-Heuser, F. Ocker, I. Weiß, R. Mieth, and F. Mann, "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*, vol. 379, no. 2207, p. 20200368, 2021. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2020.0368

[5] M. Platenius-Mohr, S. Malakuti, S. Grüner, and T. Goldschmidt, "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*, ser. IoT 2019. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3365871.3365873

[6] K. Munir and M. Sheraz Anjum, "The use of ontologies for effective knowledge modelling and information retrieval," *Applied Computing and Informatics*, vol. 14, no. 2, pp. 116–126, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210832717300649

[7] Y. Huang, L. Palacios, S. Dhouib, and J. Malenfant, "Enabling semantic interoperability of asset administration shells through an ontology-based modeling method," in *2022 25th IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS '22 Companion)*, vol. 1, 2022.

[8] E. Järvenpää, O. Hylli, N. Siltala, and M. Lanz, "Utilizing spin rules to infer the parameters for combined capabilities of aggregated manufacturing resources," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 84–89, 2018, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318313636

[9] J. Bézivin, "Model driven engineering: An emerging technical space," vol. 4143, 01 2005, pp. 36–64.

[10] R. Heidel, M. Hoffmeister, M. Hankel, and U. Döbrich, *The Reference Architecture Model RAMI 4.0 and the Industrie 4.0 component*.

[11] "Isa95, enterprise-control system integration part 1: Models and terminology," 2010. [Online]. Available: https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95

[12] S. R. Bader and M. Maleshkova, "The semantic asset administration shell," in *Semantic Systems. The Power of AI and Knowledge Graphs*, M. Acosta, P. Cudré-Mauroux, M. Maleshkova, T. Pellegrini, H. Sack, and Y. Sure-Vetter, Eds. Cham: Springer International Publishing, 2019, pp. 159–174.

[13] P. I4.0, "Describing capabilities of industrie 4.0 components." [Online]. Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Capabilities_Industrie40_Components.html

[14] Y. Huang, S. Dhouib, and J. Malenfant, "Aas capability-based operation and engineering of flexible production lines," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 01–04.

[15] D. M. C. Anto Budiardjo (Padi), "Digital twin system interoperability framework," *Digital twin consortium whitepaper*, 2021. [Online]. Available: https://www.digitaltwinconsortium.org/pdf/Digital-Twin-System-Interoperability-Framework-12072021.pdf

[16] J. deMeer, "Semantics for i4.0 smart manufacturing," in *INFORMATIK 2020*, R. H. Reussner, A. Koziolek, and R. Heinrich, Eds. Gesellschaft für Informatik, Bonn, 2021, pp. 289–298.

[17] J. Meierhofer, L. Schweiger, J. Lu, S. Züst, S. West, O. Stoll, and D. Kiritsis, "Digital twin-enabled decision support services in industrial ecosystems," *Applied Sciences*, vol. 11, no. 23, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/23/11418

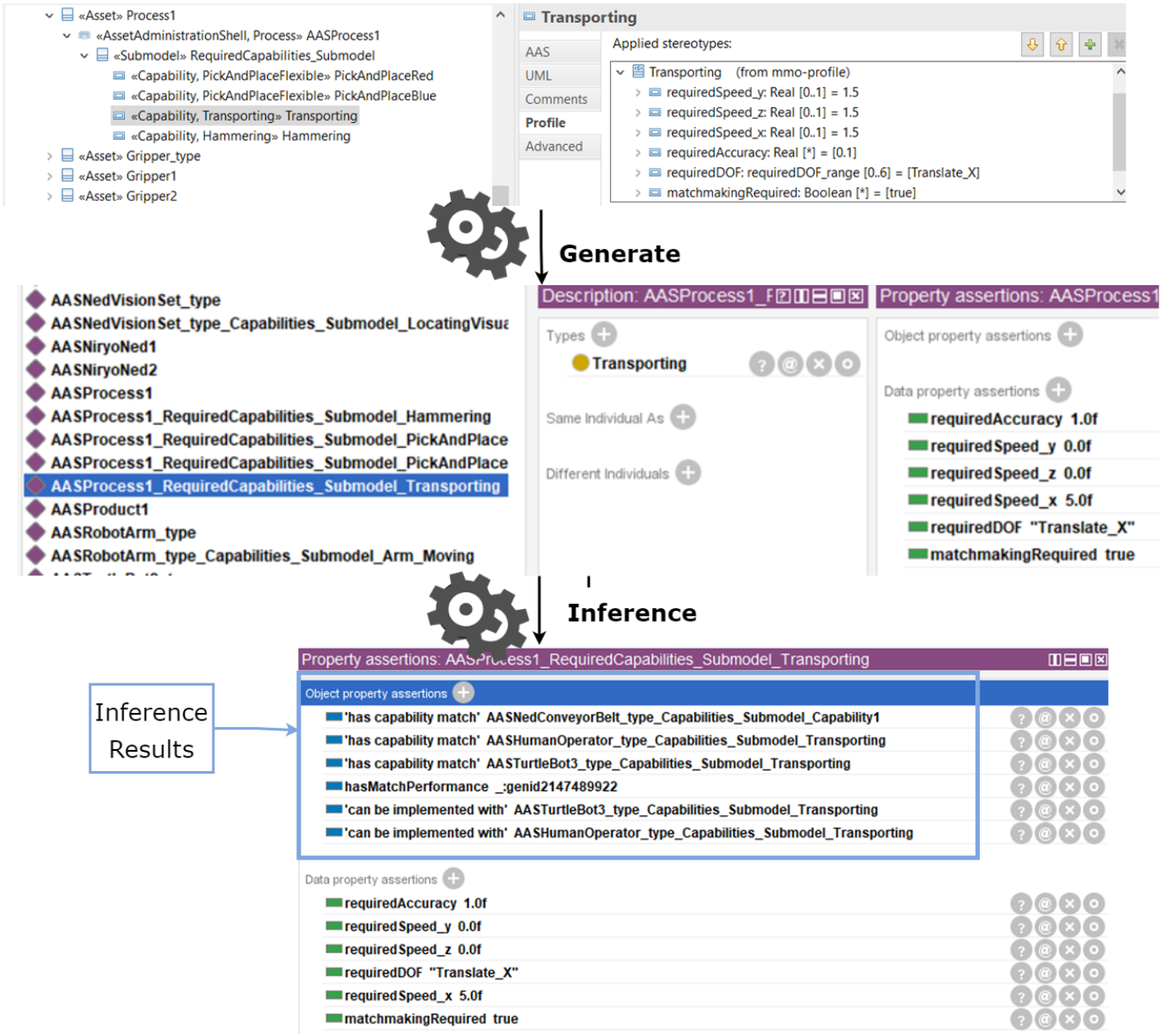

Fig. 9. An AAS2MaRCO generation of a required capability

[18] M. Weser, J. Bock, S. Schmitt, A. Perzylo, and K. Evers, "An ontology-based metamodel for capability descriptions," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1679–1686.

[19] E. Järvenpää, N. Siltala, O. Hylli, and M. Lanz, "Capability matchmaking software for rapid production system design and reconfiguration planning," *Procedia CIRP*, vol. 97, pp. 435–440, 2021, 8th CIRP Conference of Assembly Technology and Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827120314864

[20] M. Elaasar, "Definition of modeling vs. programming languages," in *Leveraging Applications of Formal Methods, Verification and Validation. Modeling*, T. Margaria and B. Steffen, Eds. Cham: Springer International Publishing, 2018, pp. 35–51.

[21] A. Mital, N. Siltala, E. Järvenpää, and M. Lanz, "Web-based solution to automate capability matchmaking for rapid system design and reconfiguration," *Procedia CIRP*, vol. 81, pp. 288–293, 2019, 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827119303555

[22] L. P. Medinacelli, C. Mraidha, and F. Noyrit, "Augmenting model-based systems engineering with knowledge," ser. 4th Workshop on Artificial Intelligence and Model-driven Engineering (MDEIntelligence), co-located with MODELS 2022.

[23] C. List. (2022) Papyrus for manufacturing model driven workbench. [Online]. Available: https://www.eclipse.org/papyrus/components/manufacturing/

[24] OMG, "Business process model and notation v2.0," 2010. [Online]. Available: https://www.omg.org/spec/BPMN/2.0/