OPC UA NodeSet Ontologies as a Pillar of Representing Semantic Digital Twins of Manufacturing Resources

Alexander Perzylo*, Stefan Profanter*, Markus Rickert*, Alois Knoll†

***Abstract*— The effectiveness of cognitive manufacturing systems in agile production environments heavily depends on the automatic assessment of various levels of interoperability between manufacturing resources. For taking informed decisions, a semantically rich representation of all resources in a workcell or production line is required. OPC UA provides means for communication and information exchange in such distributed settings.**

This paper proposes a semantic representation of a resource's properties, in which we use OWL ontologies to encode the information models that can be found in OPC UA NodeSet specifications. We further combine these models with an OWL-based description of the resource's geometry and – if applicable – its kinematic model. This leads to a comprehensive semantic representation of hardware and software features of a manufacturing resource, which we call *semantic* digital twin. Among other things, it reduces costs through virtual prototyping and enables the automatic deployment of manufacturing tasks in production lines. As a result, small-batch assemblies become financially viable.

In order to minimize the effort of creating OWL-based UA NodeSet descriptions, we provide a software tool for the automatic transformation of XML-based NodeSet specifications that adhere to the OPC Foundation's NodeSet2 XML schema.

I. Introduction

Many industrial manufacturing companies currently face changes in market demands regarding their products. As a result, they have to adjust their way of manufacturing to meet new requirements: Instead of high volume production of a few product variants, they often have to tailor their products to individual customers. This typically leads to a highly challenging situation, in which many different product variants have to be produced in only small batch sizes. In order to still maintain an economically viable production environment, the effort of programming and adjusting production processes needs to be reduced [1].

A promising approach to tackle this issue, is a modular systems engineering paradigm based on a formal representation of capabilities of manufacturing resources. These capability descriptions are part of a more generic device description and can be automatically interpreted by technical systems. They are used to identify compatible manufacturing resources for performing a given production step, and to assist a production engineer in reconfiguring a workcell or manufacturing line to match the requirements of a new or updated manufacturing task.

The OPC Foundation[1] and its members have been working on the specification of a platform-independent service-oriented architecture called *Unified Architecture (UA)* that addresses the challenges of such modular production environments. Apart from offering a communication protocol and discovery services [2], OPC UA features a flexible concept for describing and providing information models that may cover devices, particular functions, and internal system states. Such information models are hierarchically defined and build upon the base OPC UA data model and domain-specific extensions called companion specifications. This modular modeling approach allows third parties to develop their own vocabularies that are suitable for describing new devices and their capabilities.

The creation of OPC UA information models for the open source OPC UA stack *open62541*[2] and other implementations currently relies on manipulating XML files either through text editors or graphical tools. Various software tools, e.g., the OPC Foundation's UA Model Compiler, can be used to generate an OPC UA NodeSet description that complies with a corresponding XML Schema definition. The generated NodeSet description is a graph-based data structure that contains the content of information models, in which typed nodes are linked through typed references. The NodeSets can be imported and exported by OPC UA servers and might be used by OPC UA clients to browse a server's address space in an offline fashion.

However, a more semantic representation would be beneficial, in order to link and process these information models in a broader context. They must be interpreted with respect to manufacturing tasks and their requirements, as well as domain-specific and common sense knowledge.

In this paper, we propose a semantic description language for OPC UA NodeSets based on the *Web Ontology Language (OWL)*[3]. As the OWL formalism is based on description logics, OWL-based descriptions can be automatically interpreted by reasoning components, which are able to check the logical consistency of the models and derive implicit facts through logical inference. Additionally, OWL can be serialized into *Resource Description Framework (RDF)* statements. As a result, many existing software tools, such as graph data bases and SPARQL processors, can be used to persistently store and query OWL models. In order to minimize extra efforts in creating the OWL representation of UA NodeSets, we

*A. Perzylo, S. Profanter, and M. Rickert are with fortiss, An-Institut Technische Universität München, Munich, Germany.

†A. Knoll is with Technische Universität München, Munich, Germany. Correspondance should be directed to `perzylo@fortiss.org`

[1] `https://opcfoundation.org`
[2] `https://github.com/open62541`
[3] `https://www.w3.org/TR/owl2-primer/`

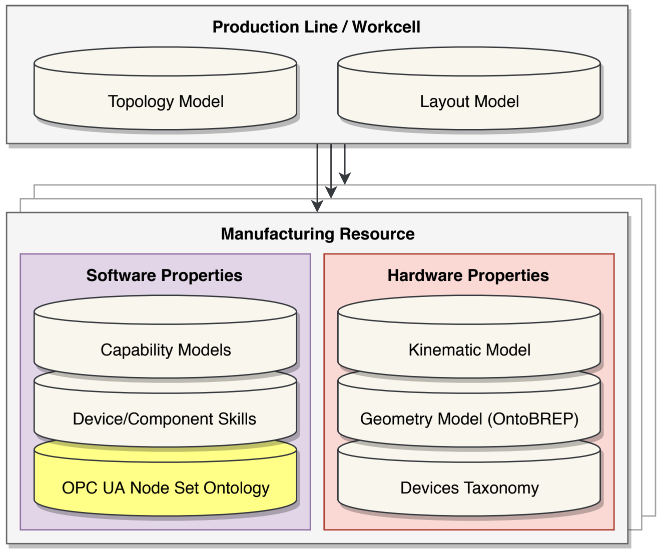


Fig. 1. Overview of proposed semantic digital twin architecture

developed a software tool for the automatic transformation of existing XML-based UA NodeSet definitions to the corresponding OWL models.

We further show how the semantic NodeSet descriptions can be extended to contain knowledge on the available executable skills of a manufacturing resource and their meta-level capability descriptions. Moreover, we combine the semantic UA NodeSet models with a description of the corresponding device's geometry based on our previously published OntoBREP CAD ontology[4]. By combining the OntoBREP geometry models with an optional OWL-encoded kinematic model, a full-fledged formal representation of hardware and software properties – a *semantic* digital twin – of a manufacturing resource can be created.

Fig. 1 depicts a visualization of the proposed ontology architecture, in which the introduced semantic representations of manufacturing resources are aggregated into layout and topology models of workcells or whole production lines.

II. Related Work

As motivated in the previous section, we aim at semantic representations of all relevant aspects of manufacturing resources, in order to automate the tasks of production systems engineering and process deployment. Parts of the proposed architecture and ecosystem have already been published. In [3], we present an ontology for describing CAD models in a semantic way based on the boundary representation (BREP) paradigm, in which the mathematical models of geometries are described instead of being approximated via polygons. One of the core applications of our semantic digital twins, is the automatic identification of suitable resources for a given semantic model of an industrial manufacturing process. The structure of our semantic process models and an intuitive way of teaching them to a robot system is introduced in [4]. The automatic mapping of a specific manufacturing task to a compatible resource is achieved by combining device descriptions with a semantic description of their provided functionalities. In this context, we distinguish between the representation of the executable elements of such functionalities, i.e., *skills* [5], and the meta-level representation of their effects, i.e., *capabilities* [6].

[4]https://github.com/OntoBREP

Next to ontology-based approaches to knowledge modeling in a manufacturing context, AutomationML is the most common language for representing various kinds of information regarding processes and manufacturing resources that need to be shared during engineering processes. Its data architecture is based on the Computer Aided Engineering Exchange (CAEX) XML format [7], which typically connects to various other industry data formats, e.g., Collada or PLCOpen XML. It supports the representation of topology, geometry, kinematics, sequencing, behaviour and control.

Some research effort was spent on the combination of AutomationML and ontologies, in order to generate automation solutions based on the robot operating system (ROS) [8]. In [9], the authors investigate the analogies between AutomationML and OPC UA information models, with the aim of AutomationML data assisting in the design of OPC UA information models.

A set of motivating examples of scenarios, in which OPC UA acts as an enabling technology to establish interoperability between Industrie 4.0 (I4.0) components or systems, is presented in [10]. A systematic approach to create OPC UA information models is presented in [11]. It is based on the automatic transformation of Unified Modeling Language (UML) models, and provides a model-based approach to OPC UA information model design. For conducting intelligent data analysis, the authors of [12] rely on an OPC UA-enabled semantic aggregation of process data of a smart factory. The integration of an ontology-based semantic engineering and data interpretation layer with an OPC UA-enabled loosely-coupled control system for astronomical instrumentation is explained in [13]. In [14] and [15], the authors present the combination of OPC UA methods with semantic service descriptions based on OWL for Web Services (OWL-S) and the Semantic Annotations for WSDL and XML Schema (SAWSDL) with the goal of automatically creating orchestration plans for manufacturing resources.

A similar approach to ours is described in [16]. The authors present an ontology-based concept for the semantic representation of an asset administration shell of I4.0 components. While this approach currently is based on a descriptive meta-level representation of such components, we aim at augmenting rather administrative information (e.g., device types, skill and skill parameter descriptions) with deep semantic models (e.g., capability, geometry, and kinematic models) – all in the same semantic language. We further relate the extended administration shell to relevant information from the manufacturing context, allowing us to refer to specific pieces of knowledge. Utilizing such a semantically rich representation of all relevant entities leads to many synergy effects. For instance, a grasp task can be described and parametrized based on individual faces of a gripper's and an object's geometry models instead of only coordinate frames and transformations. In another example, determining the compatibility of a manufacturing resource with a given task may not only involve the evaluation of a resource's capability model, but at the same time respect topological constraints of a factory.

The term digital twin originally only covers aspects of simulation. This meaning has changed in the past years to include more aspects of a manufacturing resource and a certain overlap with the I4.0 asset administration shell [17]. In our terminology, a *semantic* digital twin provides the functionality of the asset administration shell, while providing a deep semantic understanding of a resource's properties.

III. OPC UA Information Model

The communication between OPC UA-enabled devices follows a client-server paradigm. The description objects provided by an OPC UA server that are intended to be browsed by OPC UA clients are called the server's *address space*. Within such address spaces, OPC UA information models are encoded in order to inform clients about offered services and server states.

This section does not intend to explain all available concepts provided by the OPC UA specifications, but to introduce the structure of information models. The OPC UA address space specification defines the base data model of OPC UA. It contains eight different classes of nodes: *Variable*, *VariableType*, *Object*, *ObjectType*, *ReferenceType*, *DataType*, *Method*, and *View*. Every type of node provides a set of mandatory and optional attributes that can be used to further describe the node's properties. Most importantly, each node has a *nodeId* consisting of an identifier and a namespace, which are used together to uniquely refer to specific nodes. Nodes are connected through binary relations called *references*, which span a directed graph. References can be defined to be symmetric, so that some edges of the graph can be bidirectional. Various types of references have been hierarchically specified to represent different relations between nodes. As an example, a *hasTypeDefinition* reference can be used to link a *Variable* to a particular *VariableType*.

OPC UA information models use the address space concepts to describe information about a particular domain or device. They are modular and hierarchically extensible. These features are used by a multitude of UA companion specifications, which rely on the base UA NodeSet or other companion specifications and extend the contained vocabulary, e.g., with respect to object or reference types. For instance, the OPC UA companion specification for robotics is based on the companion specification for devices, which itself is defined on top of the base UA NodeSet.

For a complete description of the OPC UA address space and information models, please consult part 3 and 5 of the IEC 62541 specification [18], [19].

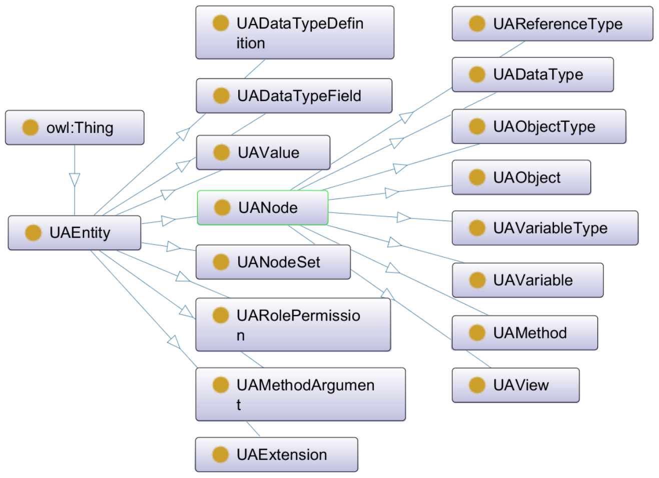


Fig. 2. Upper taxonomy of the OPC UA core ontology.

IV. OPC UA NodeSet Ontologies

For representing OPC UA information models in a semantic description language, we implemented a core OPC UA ontology using OWL. OWL can be used to define *class* taxonomies and instances of classes called *individuals*. Properties may be defined for classes and individuals. There are different types of properties, which can be hierarchically specified: *object properties*, *data properties*, and *annotation properties*. While object properties are used to link two individuals to each other, data properties link individuals to literals, e.g., strings or numbers. Annotation properties can be used to add meta-information to various ontology entities. In this work, they are used to link generated object properties to their associated UAReferenceType individual.

A. UA Core Ontology

The OPC UA core ontology describes the base classes of the OPC UA data model as introduced in Section III. Additional classes and properties have been added to properly represent certain parts of UA NodeSets in an OWL-compatible way, e.g., value arrays are converted to singly linked lists in OWL using individuals of type *UAValue* and the object property *nextValue*. In a similar fashion, *UAMethodArguments* are added to an *UAMethod* individual. The first argument is asserted to the method through object property *firstMethodArgument*, while the following arguments are linked from the previous argument through object property *nextMethodArgument*. Fig. 2 depicts a visualization of the main OWL classes used to represent NodeSet descriptions. Explaining all entities of the OPC UA core ontology is out of the scope of this paper, but the associated OWL file is available for further inspection[5].

B. Generation of UA NodeSet Ontologies

To avoid the tedious task of manually modeling OWL ontologies for OPC UA base concepts and additional companion specifications, we developed a software tool for the

[5]`https://github.com/OntoUA`

TABLE I

ONTOLOGY METRICS FOR THE OPC UA CORE ONTOLOGY AND THE INTRODUCED UA NODESET ONTOLOGIES REGARDING THE NUMBER OF AXIOMS
PLEASE NOTE: TOTAL NUMBER OF AXIOMS INCLUDES ADDITIONAL TYPES OF AXIOMS; NUMBERS ALSO INCLUDE IMPORTED DEPENDENCIES.

	C [i]	*OP* [j]	*DP* [k]	*AP* [l]	*I* [m]	*CA* [n]	*OPA* [o]	*DPA* [p]	*AA* [q]	*Total*
OPC UA core ontology	17	15	39	2	0	0	0	0	0	160
Base UA NodeSet	17	86	39	2	6067	6067	23166	41565	74	77296
Companion Spec. Devices	17	90	39	2	6271	6271	23967	43045	80	80097
Companion Spec. Robotics	17	94	39	2	6585	6585	25326	45622	84	84758
fortiss Devices	17	90	39	2	6537	6537	25399	45334	80	84421
fortiss Robotics	17	94	39	2	7007	7007	27619	49313	84	91720
fortiss Kuka iiwa robot	17	94	39	2	7240	7240	28613	51335	84	95277
fortiss GEP1402 gripper	17	90	39	2	6657	6657	25855	46207	80	86044

[i]Class [j]Object property [k]Data property [l]Annotation property [m]Individual [n]Class assertion [o]Object property assertion [p]Data property assertion [q]Annotation assertion

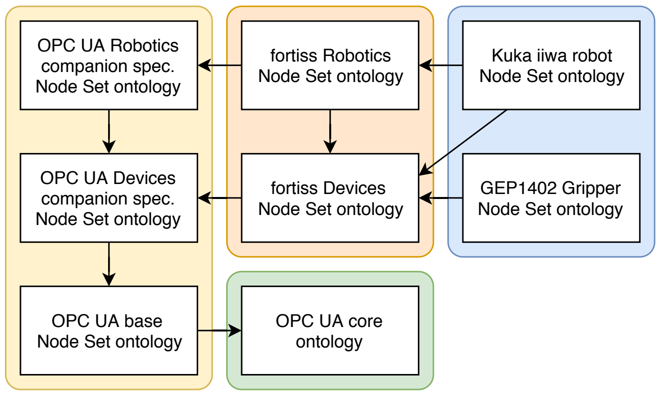

OPC UA Robotics companion spec. Node Set ontology
fortiss Robotics Node Set ontology
Kuka iiwa robot Node Set ontology
OPC UA Devices companion spec. Node Set ontology
fortiss Devices Node Set ontology
GEP1402 Gripper Node Set ontology
OPC UA base Node Set ontology
OPC UA core ontology

Fig. 3. Example of hierarchical dependencies of UA NodeSet ontologies including the OPC UA core ontology, base UA NodeSet and official OPC UA companion specifications for Devices and Robotics, fortiss extensions to these two companion specifications, and device-specific NodeSet ontologies for the Kuka iiwa robot and the GEP-1402 gripper. The arrows denote an *owl:imports* relation.

automatic generation of these ontologies in OWL. Given an UA NodeSet2 description in the official XML dialect[6], the corresponding XML Schema definition contained in *UANodeSet.xsd*, and the OPC UA core ontology, which has been described in the previous section, the transformation to OWL can be carried out automatically. The transformation tool uses XMLBeans[7] to parse the NodeSet2 description and to check it for conformity to its XML Schema. The parsed data is then encoded in OWL using the OWL API library[8] and the OWL concepts that have been defined in the OPC UA core ontology. For every node in the NodeSet description, an OWL individual is created. The node's attributes are converted to OWL data properties that are asserted to the newly created individuals. As part of the transformation process, new OWL object properties are generated based on the references that are contained in the NodeSet description. These object properties are then asserted to the OWL individuals according to the list of references for each node.

For the base OPC UA NodeSet, each companion specification, and each manufacturing resource's NodeSet description, a separate OWL ontology is generated. Hierarchical dependencies are automatically analyzed and resolved through topological sorting based on the NodeSets' model URIs (Uniform Resource Identifiers) given in the *Model* and *RequiredModel* tags. As an example, Fig. 3 shows the hierarchical dependencies of UA NodeSet ontologies for a Kuka iiwa robot and a Sommer Automatic GEP1402 parallel gripper.

Table I provides ontology metrics for the OPC UA core ontology and the introduced NodeSet ontologies, showing the number of specific OWL axioms per ontology. The Kuka iiwa robot's NodeSet ontology roughly takes 4 s to realize using the HermiT reasoner (1.3.8.413) in the Protégé ontology editor[9] (5.5.0) on an Intel i5-8600K CPU running at a base clock of 3.6 GHz. As most of the corresponding OWL axioms belong to its generic dependency on the base UA NodeSet ontology, adding additional resource descriptions to a workcell increases the total number of axioms only to a relatively small extent.

V. TOWARD A SEMANTIC DIGITAL TWIN

In this section, we introduce various elements of our proposed semantic digital twin architecture. We describe how the introduced NodeSet ontologies can be extended with models of device or component skills. Our concept is further augmented with capability models, in order to enable flexible production systems engineering. In our nomenclature, we distinguish between the terms *skill* and *capability*: The former representing the executable part of a device's functionality, and the latter being a meta-description of potential effects that the invocation of a skill may cause.

Additionally, we propose deep semantic device models, which include a mathematical representation of their geometries and – if applicable – their kinematic structure. Individual devices or components can be used as building blocks of

[6]https://github.com/OPCFoundation/UA-Nodeset
[7]https://xmlbeans.apache.org/
[8]https://github.com/owlcs/owlapi

[9]https://protege.stanford.edu/

workcell or factory models, in which device instances are placed in an environment. Their poses as well as topological connections are semantically encoded and can be queried to analyze the flow of materials.

A key aim of this approach is to make knowledge explicit that typically is hidden either in the heads of employees, software implementations, or natural language specifications. By doing so, maintaining and reusing this knowledge can be more easily accomplished. Moreover, technical systems are enabled to process the knowledge and make informed decisions, resulting in a higher degree of automation and system resilience.

In the following sections, we describe the introduced aspects of our semantic digital twin architecture in more detail, and whenever possible provide references to our previous publications on individual submodels.

A. Device and Component Skills

Every device and component in a production system should provide a hardware-agnostic interface. This interface should be modeled in a generic way, independent of the component's functionality, i.e., its skills.

In [5], we present a corresponding skill model. It is completely modeled in an OPC UA NodeSet description and uses separate state machines for each offered skill, in order to represent their internal states. State transitions can be controlled through dedicated UA methods, which are the same for every skill. Before the invocation of a skill, its skill parameters need to be set. The referenced paper shows that such a skill model can be used to create a generic interface to the functionality of hardware as well as software components. Component skills can be hierarchically arranged in order to combine them into higher-level functionalities, while still relying on the same skill interface.

With this skill model, it is possible to simply exchange system components, as long as they provide the same subset of required skill types. The system's higher-level functionalities and its task control do not have to be changed. For instance, [5] demonstrates that an industrial robot can be replaced by a robot from a different manufacturer without requiring any changes to the task implementation.

Since our skill model is encoded in the NodeSet2 description of a device or component, it is part of the automatically generated UA NodeSet ontologies, as described in Section IV-B. The available skill and parameter types are also modeled in the NodeSet, therefore, a mapping from ontology parameters to the corresponding skill parameters can be created.

In order to support a more flexible compatibility evaluation of manufacturing resources, skill models can be augmented with a meta-level capability description.

B. Capability Models

Capability models are required to determine the compatibility of a manufacturing resource with a requested task. Many approaches rely on using standardized property sets to

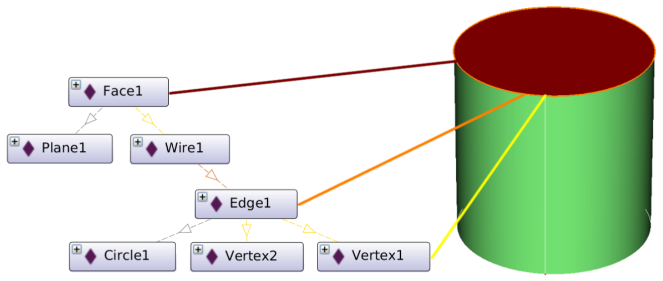


Fig. 4. Visualization of an excerpt of a deep semantic model of a cylinder's geometry based on the OntoBREP ontology.

describe the effects of a resource. These properties are provided as part of a resource model, and aggregated in groups that relate to specific skills. An ontology-based approach to capability modeling enables a cognitive production system to not only match requirements and offered capabilities, but to also automatically derive orchestrated capabilities from basic ones [6].

The semantic depth of these models directly influences the level of confidence of such an evaluation. As a purely symbolic investigation cannot guarantee a successful execution of a task, more (subsymbolic) evaluation techniques are needed, e.g., using analytic means or simulations. A symbolic capability matching process can be seen as a prefilter for potential solutions, thus reducing the search space of computationally expensive simulations. An engineered solution provided by a single party will most likely not be able to tackle this issue completely. As a result, we suggest an open ontology-based concept for capability matching, in which different parties may extend the core concepts according to their own needs.

C. Geometry Models

For describing geometric properties of products and manufacturing resources alike, we rely on our OntoBREP ontology [3]. As the name implies, the ontology follows a BREP paradigm, in which faces, edges, and vertices are typically specified by defining an infinite geometry and corresponding bounds that make it finite. For instance, the top face of the cylinder depicted in Fig. 4 (represented by OWL individual *Face1*) has been specified by an infinite plane (*Plane1*) that is bounded by an edge (*Edge1*) of type circle (*Circle1*). The individual *Wire1* represents a topological BREP structure, which may hold multiple connected edges. The given edge itself is bounded by two vertices (*Vertex1* and *Vertex2*), which happen to be at the exact same position, resulting in *Edge1* being a complete circle.

Using deep geometry models, it is possible to annotate any subelement of an encoded resource or object with additional information. For instance, this can be used to geometrically describe regions of interest on a device, e.g., where to put a workpiece or which areas to avoid due to moving parts. In assembly specifications, the OntoBREP-based representation can be combined with semantically specified geometric interrelational constraints [3], e.g., stating that two faces must be coincident after a particular assembly step.

```
Individual: cell:KUKA_iiwa_7_R800_1_body2
  Types:
    rob:KUKA_iiwa_7_R800_body2
  Facts:
    swdl:succeedingJoint  cell:KUKA_iiwa_7_R800_1_joint2,
    brep:shape  link2:Solid1,

Individual: cell:KUKA_iiwa_7_R800_1_joint2
  Types:
    swdl:RevoluteJoint
  Facts:
    swdl:succeedingLink  cell:KUKA_iiwa_7_R800_1_frame3,
    swdl:transformation  cell:RigidTransformationMatrix_12

Individual: cell:RigidTransformationMatrix_12
  Types:
    brep:RigidTransformationMatrix
  Facts:
    brep:a11  "1.0"^^xsd:double,
    brep:a12  "0.0"^^xsd:double,
    ...
    brep:a44  "1.0"^^xsd:double
```

Fig. 5. Excerpt of the kinematic model of a Kuka iiwa robot in Manchester OWL syntax. It shows a body of the robot, a succeeding revolute joint, and the corresponding transformation matrix.

D. Kinematic Models

The kinematic and dynamic behavior of robot manipulators, grippers, and other manufacturing resources can be described via multibody systems [20]. Such a definition includes references to rigid bodies that are connected via a number of either fixed or actuated transformation elements. A rigid body is described via its mass, center of gravity, and inertia properties. It directly relates to an element of its geometric model via its calculated 3D position and orientation in world coordinates. A transformation between individual rigid bodies describes the relative position and orientation to each other, leading to a tree structure or in case of closed loops even a graph structure. Different joint types such as revolute, prismatic, spherical, or even 6-DOF joints can be used to describe the system's state based on its current state vector. This includes the individual joint's position, velocity, and acceleration values. Each joint includes additional properties such as maximum and minimum joint angles, or maximum allowed speed.

The kinematics model in our ontology is based on the one used in the Robotics Library [21], which supports all properties described above and includes a matching C++ implementation of kinematics and dynamics algorithms. Other existing specifications on how to describe multibody systems include Gazebo's SDF[10] and ROS' URDF[11] formats.

Fig. 5 provides an insight into the kinematic model of a Kuka iiwa robot. It shows a particular body *cell:KUKA_iiwa_7_R800_1_body2* being linked to its geometry model *link2:Solid1* via the *brep:shape* object property. Furthermore, property *swdl:succeedingJoint* links to OWL individual *cell:KUKA_iiwa_7_R800_1_joint2*, which is of type *swdl:RevoluteJoint*. The corresponding transformation is given through matrix *cell:RigidTransformationMatrix_12*.

[10]http://sdformat.org/
[11]http://wiki.ros.org/urdf

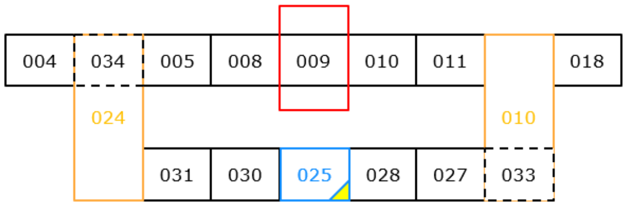
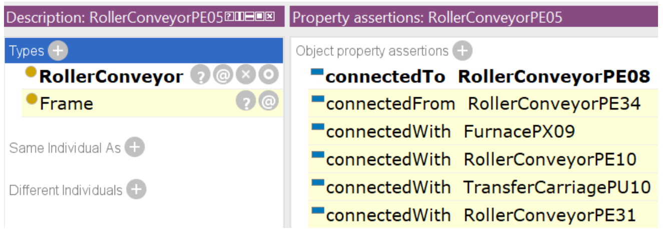


Fig. 6. Excerpt of the topology of a pallet conveyor system inside of a cold rolling mill consisting of multiple roller conveyors, two transfer carriages (024, 010), a rotary roller conveyor (025), and a furnace (009).


Description: RollerConveyorPE05

Types
- RollerConveyor
- Frame

Same Individual As

Different Individuals

Property assertions: RollerConveyorPE05

Object property assertions
- connectedTo RollerConveyorPE08
- connectedFrom RollerConveyorPE34
- connectedWith FurnacePX09
- connectedWith RollerConveyorPE10
- connectedWith TransferCarriagePU10
- connectedWith RollerConveyorPE31

Fig. 7. Partial screenshot of the Protégé ontology editor showing explicitly modeled object property assertions (white background) and a subset of inferred property assertion (yellow background) for a selected manufacturing resource *RollerConveyorPE05*, which is an instance of OWL classes *RollerConveyor* and *Frame*.

The matrix values are asserted by 16 data properties, *brep:a11* to *brep:a44*.

E. Factory and Workcell Models

The semantic representations of manufacturing resources can be further combined into workcell or even factory models. The presented approach consists of an OWL-based layout and topology model. While the layout model specifies where resources are located with respect to each other and within a given environment, the topology model contains the logical connectivity information regarding the flow of materials between resources.

In a layout model, a device's location is given as a rigid transformation matrix that encodes the transformation between an environment's reference frame and the device's coordinate frame. The transformation matrix itself is encoded in OWL, which also enables the calculation of absolute positions of devices along a kinematic chain of relative transformations via SPARQL-based matrix multiplications.

Topology models use dedicated OWL object properties, e.g., *connectedWith*, *connectedTo*, and *connectedFrom*, which encode bidirectional and the two possible unidirectional logical connections between two manufacturing resources. The *connectedWith* property has been defined to be transitive and symmetric, the *connectedTo* and *connectedFrom* properties are asymmetric and subproperties of *connectedWith*. *connectedTo* is the inverse property of *connectedFrom*. Only the *connectedTo* property is used for the explicit modeling of pairwise connected resources. The other property types and the associated transitive hull can be automatically inferred.

Fig. 6 depicts an overview of the topology of various resources of a pallet conveyor system inside of a cold rolling

```
PREFIX mil: <http://www.fortiss.org/basys/rollingmill.owl>
ASK {
  mil:FurnacePX09 mil:connectedWith mil:RollerConveyorPE11
}
```

Fig. 8. SPARQL query to check whether two resources are topologically connected or not. The ASK query will be evaluated to *yes* for the introduced topology example.

```
PREFIX mil: <http://www.fortiss.org/basys/rollingmill.owl>
SELECT ?r WHERE {
  mil:RollerConveyorPE05 mil:connectedTo* ?r .
  ?r mil:connectedTo* mil:FurnacePX09 .
}
```

Fig. 9. SPARQL query to retrieve the route from resource *mil:RollerConveyorPE05* to target resource *mil:FurnacePX09*. The returned bindings for variable *?r* are *mil:RollerConveyorPE05*, *mil:RollerConveyorPE08*, and *mil:FurnacePX09*.

mill. Black, orange, blue, and red boxes represent roller conveyors, transfer carriages, rotary roller conveyors, and a furnace, respectively. A transfer carriage can move the associated roller conveyor so that it can be part of different routes. Rotary roller conveyors cannot only transport material coils, but they can also change the coils' orientation by rotating them in steps of 180 degrees. A furnace is used to control the temperature of transported material coils prior to processing them further.

Exploiting the logical formalism of OWL, the topology model can be efficiently queried using OWL reasoners and the SPARQL query language. A subset of inferable object property assertions for a specific instance of a roller conveyor resource is shown in Fig. 7. When inferred statements are materialized, SPARQL processors can consider them when they evaluate queries. For instance, it can be queried whether two resources are topologically connected or not (see Fig. 8), or which resources are part of a connecting route (see Fig. 9). In this experiment, GraphDB[12] was used for persistent storage of the topology model and as materialization and querying engine.

In order to achieve the same functionality with AutomationML, more complex XPath/XQuery expressions would need to be combined with a dedicated software component that knows about the implications of topological relations [22].

VI. Conclusion and Future Work

This paper introduces a novel way of representing OPC UA information models in a semantic way based on UA NodeSet ontologies. We presented our automatic transformation tool that is able to generate ontologies for the base UA NodeSet, OPC UA companion specifications, and arbitrary information models. By describing executable skills within NodeSet descriptions, they are automatically present in the generated OWL representation as well. With the help of capability models, the effects of skills are semantically described, enabling manufacturing systems to reason about the compatibility of resources with the requirements of manufacturing tasks.

We further explained our envisioned concept of a semantic digital twin that combines the already mentioned types of information with a semantic description of hardware features, such as a BREP representation of a resource's geometry and a kinematic model for its moving parts. Additionally, we presented an example of how individual resources can be arranged in a production environment. This includes the physical location of resources and their topological connectivity.

While many submodels of our semantic digital twin architecture are currently interpreted individually, we work toward a highly integrated setup that is able to better showcase the additional benefits of a common language for representing diverse aspects of manufacturing environments.

In summary, the semantic digital twin aims at providing access to all relevant information of a manufacturing resource in a formal language that can be easily maintained and flexibly interpreted. Instead of restricting the use of ontologies to provide an upper-level semantic integration layer, we combine such a layer with deep semantic models of relevant entities of the production systems engineering domain. This approach enables the flexible representation of semantic relations between any high-level or low-level aspect of the overall manufacturing system.

Using the logical formalism behind the semantic description languages, the consistency of encoded models can be automatically checked and implicit facts derived. The ontology ecosystem is open and can be easily extended, allowing represented knowledge to be seamlessly related to external sources of information.

[12]https://www.ontotext.com/products/graphdb/

Acknowledgment

The research leading to these results has been partly funded by the German Federal Ministry of Education and Research (BMBF) under grant agreement no. 01IS16022 in the project BaSys 4.0 and by the Bavarian Ministry of Economic Affairs, Regional Development and Energy (StMWi) under grant agreement no. IUK-1711-0033 in the project Data Backbone with project support from the Zentrum Digitalisierung Bayern (ZD.B).

References

[1] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. B. Beck, M. Haage, M. R. Hansen, M. T. Nibe, M. A. Roa, O. Sörnmo, S. G. Robertz, U. Thomas, G. Veiga, E. A. Topp, I. Kessler, and M. Danzer, "SMErobotics: Smart robots for flexible manufacturing," *IEEE Robotics & Automation Magazine*, vol. 26, no. 1, pp. 78–90, Mar. 2019, https://doi.org/10.1109/MRA.2018.2879747.

[2] S. Profanter, K. Dorofeev, A. Zoitl, and A. Knoll, "OPC UA for plug & produce: Automatic device discovery using LDS-ME," in *Proceedings of the IEEE International Conference on Emerging Technologies And Factory Automation (ETFA)*, Limassol, Cyprus, Sep. 2017.

[3] A. Perzylo, N. Somani, M. Rickert, and A. Knoll, "An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, Sep. 2015, pp. 4197–4203.

[4] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll, “Intuitive instruction of industrial robots: Semantic process descriptions for small lot production,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Republic of Korea, Oct. 2016, pp. 2293–2300.

[5] S. Profanter, A. Breitkreuz, M. Rickert, and A. Knoll, “A hardware-agnostic OPC UA skill model for robot manipulators and tools,” in *Proceedings of the IEEE International Conference on Emerging Technologies And Factory Automation (ETFA)*, Zaragoza, Spain, Sep. 2019.

[6] A. Perzylo, J. Grothoff, L. Lucio, M. Weser, S. Malakuti, P. Venet, V. Aravantinos, and T. Deppe, “Capability-based semantic interoperability of manufacturing resources: A BaSys 4.0 perspective,” in *Proceedings of the IFAC Conference on Manufacturing Modelling, Management, and Control (MIM)*, Berlin, Germany, Aug. 2019.

[7] IEC 62424, “Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools,” 2016.

[8] Y. Hua, S. Zander, M. Bordignon, and B. Hein, “From AutomationML to ROS: A model-driven approach for software engineering of industrial robotics using ontological reasoning,” in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Berlin, Germany, Sep. 2016.

[9] R. Henßen and M. Schleipen, “Interoperability between OPC UA and AutomationML,” *Procedia CIRP*, vol. 25, pp. 297–304, 2014, International Conference on Digital Enterprise Technology (DET). https://doi.org/10.1016/j.procir.2014.10.042.

[10] M. Schleipen, S.-S. Gilani, T. Bischoff, and J. Pfrommer, “OPC UA & Industrie 4.0 - enabling technology with high diversity and variability,” in *Proceedings of the CIRP Conference on Manufacturing Systems (CMS)*, Stuttgart, Germany, May 2016, pp. 315–320, https://doi.org/10.1016/j.procir.2016.11.055.

[11] F. Pauker, T. Frühwirth, B. Kittl, and W. Kastner, “A systematic approach to OPC UA information model design,” in *Proceedings of the CIRP Conference on Manufacturing Systems (CMS)*, Stuttgart, Germany, May 2016, pp. 321–326, https://doi.org/10.1016/j.procir.2016.11.056.

[12] S. Wang, J. Wan, D. Li, and C. Liu, “Knowledge reasoning with semantic data for real-time data processing in smart factory,” *Sensors*, vol. 18, no. 2, 2018, https://dx.doi.org/10.3390%2Fs18020471.

[13] W. Pessemier, G. Raskin, H. V. Winckel, G. Deconinck, and P. Saey, “A practical approach to ontology-enabled control systems for astronomical instrumentation,” *CoRR*, vol. abs/1310.5488, 2013, https://arxiv.org/abs/1310.5488.

[14] B. Katti, C. Plociennik, and M. Schweitzer, “SemOPC-UA: Introducing semantics to OPC-UA application specific methods,” *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1230–1236, 2018, IFAC Symposium on Information Control Problems in Manufacturing (INCOM) 2018.

[15] B. Katti, C. Plociennik, M. Ruskowski, and M. Schweitzer, “SA-OPC-UA: Introducing semantics to OPC-UA application methods,” in *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE)*, Munich, Germany, Aug. 2018, pp. 1189–1196.

[16] I. Grangel-González, L. Halilaj, G. Coskun, S. Auer, D. Collarana, and M. Hoffmeister, “Towards a semantic administrative shell for industry 4.0 components,” in *IEEE International Conference on Semantic Computing (ICSC)*, 2016, pp. 230–237.

[17] C. Wagner, J. Grothoff, U. Epple, R. Drath, S. Malakuti, S. Grüner, M. Hoffmeister, and P. Zimermann, “The role of the Industry 4.0 asset administration shell and the digital twin during the life cycle of a plant,” in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Limassol, Cyprus, Sep. 2017.

[18] IEC 62541-3, “OPC Unified Architecture – Part 3: Address Space Model,” 2015.

[19] IEC 62541-5, “OPC Unified Architecture – Part 5: Information Model,” 2015.

[20] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer, 2008.

[21] M. Rickert and A. Gaschler, “Robotics Library: An object-oriented approach to robot applications,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, Sep. 2017, pp. 733–740.

[22] M. Wenger, A. Zoitl, and T. Müller, “Connecting PLCs with their asset administration shell for automatic device configuration,” in *Proceedings of the IEEE International Conference on Industrial Informatics (INDIN)*, Jul. 2018, pp. 74–79.