Integrating OWL Ontologies for Smart Services into AutomationML and OPC UA

Andreas Bunte[1]
[1]Institute Industrial IT
OWL University of Applied Science, Germany
Email: andreas.bunte@hs-owl.de

Oliver Niggemann[1,2]
[2]Fraunhofer IOSB-INA, Germany
Email: oliver.niggemann
@iosb-ina.fraunhofer.de

Benno Stein[3]
[3]Digital Bauhaus Lab
Bauhaus University Weimar, Germany
Email: benno.stein@uni-weimar.de

***Abstract*—This work shows how OWL ontologies can be represented into the automation standards AutomationML and OPC UA. It is often asserted that an integration is possible, but no detailed review could be found. The integration of OWL into the standards is relevant, because it enables the collection and usage of data through the whole life cycle in OWL. We show that it is possible, but we identified some restriction regarding the representation in OPC UA.**

I. Introduction

Industrial automation is driven by trends such as Industrial Internet [1] or the German counterpart Industrie 4.0 [2]. Their aim is to make production systems more flexible and thus enable small batch sizes. Additionally, they have to be more resource-efficient because of environmental aspects and costs. To achieve these goals, additional functionalities support the operators with services such as self-diagnosis capabilities to enable predictive maintenance, optimization of production parameters, or the autonomous planning of the logistic chain. These functions can be enabled by smart services [2], which are software components with functional interfaces which are executed during the run-time.

Today, smart services[1] are often specialized for a single system and an adaption has to be performed manually, which is not suitable for flexible production systems. The interoperability has to be increased to enable services that adapt themselves to the current situation. It is a field of active research, where different standards are combined to increase the interoperability, e.g. in OLE for Process Control Unified Architecture (OPC UA) [3] [4]. This is an improtant issue, since there are more than 25 industrial ethernet protocols on the market [3].

Additionally, the availability of knowledge is a crucial point that is addressed in this paper. By enabling a standardized representation of knowledge during the whole production plant's life cycle (see Fig. 2), multiple acquisitions of knowledge are prohibited, which leads to a larger, and consistent knowledge base. The term 'knowledge' is used in the sense of AI, where a formal language based on first-order logic is used to describe knowledge about a specific domain, e.g. by using ontologies. Based on the formal descriptions, it is possible to check the consistency and to infer new facts. An example for such knowledge is: 'the total power consumption is the sum of all single modules'. This enables an automatic adaption e.g. if a new production module is integrated. Furthermore, ontologies are still mentioned in current versions of Plattform Industrie 4.0 [5] as technology for defining the vocabulary and representing knowledge.

Since there are many kinds of knowledge, it has to be defined what kind of knowledge is required for which smart service. These different kinds of knowledge are represented in the lower line of Fig. 1 (marked with italic font), whereas the upper line contains some smart services. Two services in the figure are exemplarily linked to the required knowledge. The knowledge can be divided into type knowledge (white boxes) and instance knowledge (grey boxes). Type knowledge is independent of a specific production system and describes concepts such as characteristics of causality, energy and sensors, whereas the instance knowledge describes a specific scenario, e.g. *if sensor 12 detect a product, conveyor3 have to stop*.

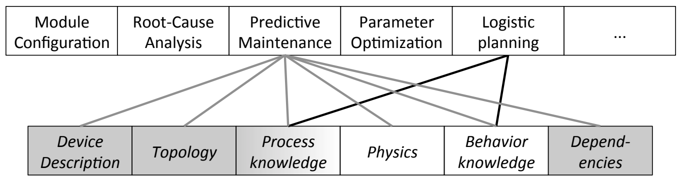

Fig. 1. Types of knowledge in the lower line, smart services in the upper line and exemplary the relation between them

Most of the instance knowledge can be acquired during the engineering phase. The Automation Mark-up Language (AutomationML/AML) standard is used for the data exchange between different engineering disciplines and can also be used to collect the knowledge during the engineering phase. The acquired knowledge can be provided via the OPC UA standard, which is used in the operational phase. Since knowledge is covered in the Web Ontology Language (OWL), a suitable representation of ontologies in AML and in OPC UA has to be identified. This raises up two research questions (RQ):

RQ 1: How can smart services be integrated in a future automation architecture?

[1]The term knowledge based service can be used synonymously, because we propose that services require knowledge about the system to perform an intelligent/smart behavior.

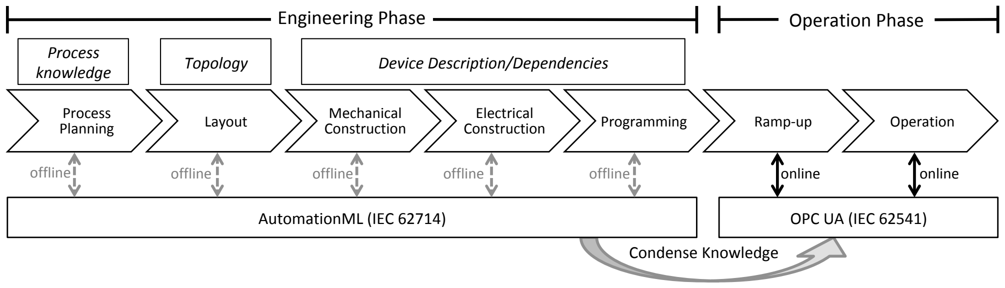

Fig. 2. Usage of AutomationML and OPC UA during different phases of the production plant's life-cycle

RQ 2: Can all types of required knowledge be integrated into the automation standards OPC UA and AML by using ontologies or are there any limitations?

The contribution of our paper is a concept that describes how ontologies can be integrated into existing automation standards to enable knowledge-based services such as self-configuration, self-optimization, or self-diagnosis. Our paper gives a review of the standards AML and OPC UA and makes suggestions how these standards can be extended to fit the needs of knowledge-based services.

II. Existing Standards and Use-Cases

The information flow of the production plant's life-cycle can be realized by just two standards which are introduced in this section. The focus is on promising standards which are already mentioned in the context of Industrie 4.0, so no detailed review on alternatives is given here. Furthermore, the OWL is introduced, which is well known for knowledge representation for smart services. There are some works that use OWL, or subsets of it, to provide helpful functions, such as answering natural language questions [6], representation of maintenance information [7], diagnosis [8], or configuration [9].

At the end of this section, three use-cases are introduced, which can be solved by smart services. These use-cases are chosen from the Plattform Industrie 4.0, so they act as reference for Industrie 4.0 applications [10] . They are decribed on an abstract level here, but they are used in a case-study in Section IV.

A. Overview

Fig. 2 shows the life-cycle of a module for a production plant. It can be divided into the engineering phase and the operation phase. The rectangles above the life-cycle steps indicate the knowledge that is generated during this step. Since instance knowledge is specific to one module, it has to be caught during the engineering phase, at the point where it is generated. Since the data usage is offline, there are no time constraints. AML can be used for the data exchange during the engineering phase and is seen as one standard for Industrie 4.0 [11]. To cover the knowledge in each process step, there must be the possibility to integrate ontologies into the AML standard. Since knowledge has to be transformed back into an ontology, the representation in AML must be unique.

During the operation phase, the knowledge has to be available at runtime, so an online access is required. OPC UA is a promising standard for it, because it provides an expressive information model and it is also suggested as important standard for Industrie 4.0 [11]. The knowledge has to be transformed from a representation in AML to the information model of OPC UA. A direct transformation is possible and we propose a bijective transformation from the ontology to each other format, because it will ease the integration of a new standard or the replacement of an old one.

B. Web Ontology Language (OWL)

A popular modeling language for ontologies is the OWL, which is also used in the following. OWL is a logic language with different profiles which can be chosen depending on the applications. Here, the Description Logic (DL) profile is used, because DL is a decidable but still expressive profile subset of the most expressive OWL full profile. Due to the good coverage of tools and the possibility to create expressive ontologies, it is also attractive for different domains. OWL ontologies can be represented through the Resource Description Framework (RDF), which is a triplestore. RDF triple stores use Extensible Markup Language (XML) to represent and store the information. Therefore, many standard reasoners are available and can be used without much effort, such as HermiT or Pellet.

To increase the expressiveness and thus the reasoning capability, OWL can be extended with the Semantic Web Rule Language (SWRL). SWRL rules are formulas that are applied to the instances, not to classes. SWRL does not only provide simple rules, but also built-ins. They can be used e.g. to handle strings, or mathematical calculations. [14]

C. OPC UA

The OPC UA is a platform-independent service-oriented architecture (IEC 62541), which is established in industrial environments to communicate and exchange data between machines. This protocol is located in the software-based upper layers (5 to 7) of the ISO/OSI Model and is therefore independent of the physical hardware that is connecting the participating devices. This flexibility is a major advantage for automation industries and makes OPC UA predestined for M2M communication, SCADA, HMI, and ERP applications. It enables a flexible communication network between devices of different vendors with authentication and encryption [15]. The communication of a OPC UA is based on a client-server-architechture, where OPC UA clients frequently send messages, or the server frequently sends messages to all subscribed clients. Servers hold an information model that describe their internal structure, which can be created by the use of object-oriented modeling techniques. The information model can be browsed and manipulated by clients, so it can be used for control purposes. The server also offers services in combination with this model. Vendors and interest groups can define their own specifications for the information model, which are known as OPC UA Companion Standards. Many groups work on these OPC UA Companion Standards, such as FDI (Field Device Integration), ADI (Analyzer Device Integration), and PLCOpen to name a few.

D. AutomationML

AML is a standardized data exchange format (IEC 62714) which can combine information, such as topology, geometry, kinematics, behavior, and network, for many engineering disciplines. Therefore, several existing standards, named CAEX, COLLADA and PLCopen XML, are used to represent the information. The XML-based data format is used to present this information.

CAEX is the top-level format for AML and contains roles, classes, instances, interfaces, and relations between them. The two other formats, COLLADA and PLCopen XML, are integrated through a reference from the CAEX file using the *COLLADAInterface* or *PLCopenXMLInterface*. Furthermore, AML provides an *ExternalDataConnector*, which can be used to refer to unspecified files such as documentations. The standard is not closed, AML can be extended with a new specific interface for another XML based data exchange format. There is an ongoing development, e.g. now it is possible to integrate eCl@ss information in the CAEX-File [16]. Another specification provides the possibility to transform an AML-File into a OPC UA information model [17], but this would only provide the information from the top-level format CAEX in a structured manner. Because it has to be transformed within two steps (from OWL to AML and from AML to OPC UA), the maintenance and access of knowledge with smart services would be more complicated and thus it seems not suitable to use this specification.

E. Use-Cases

Use-Case 1 - Adaptable Factories: Future production lines have to be adaptable to satisfy the demand of small batches and individual products. New production modules should be integrated with minimal or no manual effort to achieve that goal, e.g. a printer to label products can be changed to enable different materials. To keep the costs for such adaptions low, the whole reconfiguration has to be done automatically. This requires every module to contain a semantically unambiguous self-description of its properties and capabilities [11].

Use-Case 2 - Human-Machine-Interaction: In steadily changing environments, today's static interaction methods are not capable to fulfill the future needs. Besides, the need for adaptable interfaces and the amount of information is constantly increasing. Systems have to filter relevant information, e.g. according to the position of the operator. For this, new technologies such as smart watches, smart glasses, or advances in natural language processing provide new possibilities for the interaction. This aims to motivate the operator and enable him to cover more tasks efficiently. [10]

Use-Case 3 - Value-Based Service: Value-Based Services (VBS) are services to improve the usage of the machine with the usage of the machine's data. Typically, the vendor of a machine offers VBS but it is not limited to the vendor. Services which can be offered are manifold, and also the goals that can be achieved with it. For example, a parametrization of the machine for a given material to optimize the process, a condition monitoring system to reduce the downtime, or a rented machine is payed regarding the stress. [18]

All three use-cases have in common that their impact increases by using smart services. Therefore, the knowledge from all modules has to be online accessible for the services.

III. Proposed Approach

The main goal of this section is *(i)* to show how smart services can fit into the future automation structure (RQ1) and *(ii)* to propose a solution how knowledge, as OWL ontology, can be represented in AML and OPC UA (RQ2).

A. Positioning of Smart Services

At the present time, there are not many Industrie 4.0 standards, but a great effort is being made to develop them. Because of the large heterogeneity in industries, it is expected that there will be no single standard that fits to all domains [19]. However, some basic concepts are quasi fixed and introduced here.

A device which fulfills requirements regarding Industrie 4.0 is called Industrie 4.0 component. Therefore, it is not specified whether it is a single sensor, a whole factory, or something in between. It mainly depends on the use-cases that are addressed. By analyzing the application scenarios of the Plattform Industrie 4.0 [20], all scenarios can be reached with an asset administration shell (AAS) on module level, at least partly. So, for this work it is suggested that the AAS is implemented on module level. An AAS on sensor level would enable some benefits for a few use cases, but as such sensors would be much more expensive; this will play a minor role for the short- and medium-term.

Every Industrie 4.0 component has an asset and an AAS. The connection between the AAS and the asset is not specified, it can be proprietary. In comparison, the AAS provides a standardized communication interface which can communicate with other components without previous configuration. However, it is not communication as known today, the Industrie 4.0 components have to be more flexible. They must be able to have a common understanding of the exchanged information, e.g. for automated production planning or to negotiate contracts. To achieve this common understanding of information and knowledge, semanticsis an issue [19]. Since semantics can be represented by ontologies, it is a comprehensible need to exchange ontologies with standards used for Industrie 4.0.

An often used standard is the Reference Architecture Model Industrie 4.0 (RAMI4.0, DIN-SPEC 91345), which covers the life cycle, the hierarchy levels and the layers of a production system, see Fig. 3. The *Type* phase of the life cycle covers the engineering phase (*Development*) and the implementation of a prototype *Maintenance/Usage*. The *Instance* phase starts with the *Production* of the instance and migrate to the *Maintenance/Usage* phase during ramp-up. The hierarchy levels are presented according to the automation pyramid, with the extension of a product on the lower side and the connected world on the upper side.

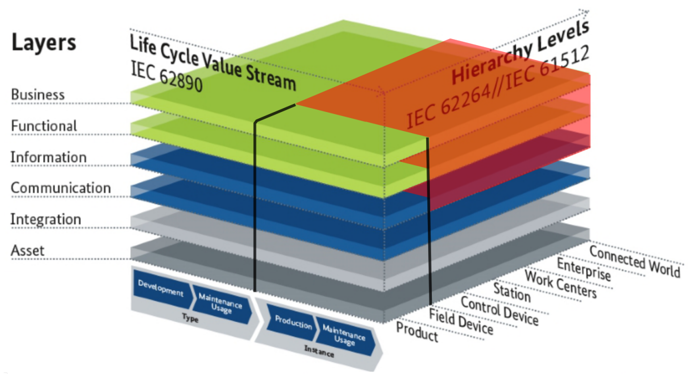

Fig. 3. Positioning of the mainly used smart services in RAMI4.0 [21]

This model will be used to locate the smart services in industry processes and derive possibilities for the integration into the automation architecture. Smart services, mentioned in this paper, are arranged in RAMI4.0 as marked red in Fig. 3. They should support the operator, for example by supporting the configuration during ramp-up or by diagnosing an anomaly, so they are located at the *Instance* phase (*Production* and *Maintenance/Usage*). On the hierarchical level, smart services cover the whole range between the field devices and the connected world. Mainly, the smart services are located in the functional layer, but the business and information layer can also be touched. The layers below are on a communication level and do not need explicit knowledge to perform their tasks; it is covered by the specific communication technology.

B. Automation Structure

As we can see in the RAMI4.0, the services are used during the *maintenance/usage* phase. This requires a standard which can provide the required knowledge online. Additionally, to the online capability, the standard must be compatible to the AAS. The AAS can be mapped bidirectionally to the OPC UA Device Interface (IEC 62541-100). [11] This makes it possible to transform IEC 62541-100 models to AAS models and thus to increase the compatibility for an existing plant. As an alternative, the Message Queuing Telemetry Transport (MQTT/HTTP) is discussed as a possible technology to implement the AAS. However, MQTT has a lot of limitations and thus cannot provide as much functionalities as OPC UA. Because it needs less resources, it could be interesting for smaller devices [11], but not for our use-cases on module level.

To sum it up, Fig. 4 presents a proposal of how smart services can be integrated into future automation systems. OPC UA is used to communicate with the IT-Infrastructure. The AAS is implemented as OPC UA information model. Knowledge models of modules are located in the AAS and can be accessed through the standardized Industrie 4.0 communication. Communication within the module is not restricted and can be realized by traditional field buses. Smart services are performed on the IT-Infrastructure, because the performance is needed to handle complex algorithms for large production systems. The processing of ontologies is expensive, due to the exponentially increasing reasoning time [22]. It does not seem appropriate to reason on controllers with limited resources.

In Fig. 4, a diagnosis service is randomly chosen, which fits to use-case 3 (VBS). The service contains machine learning algorithms to detect anomalies along with a reasoning engine to perform the root-cause analysis. The reasoning engine requires a plant ontology which is combined from module ontologies and provide a standardized output for a user interface.

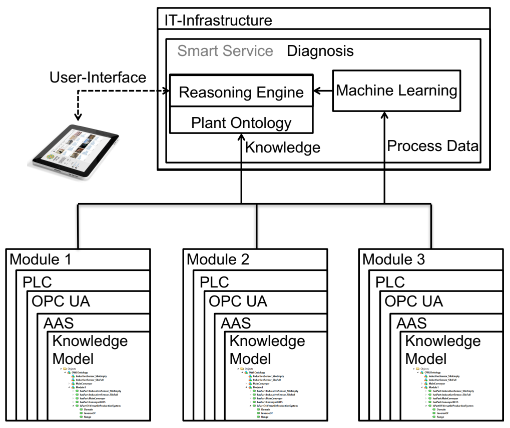

Fig. 4. Possible integration of smart services into future automation systems

C. Required knowledge and ontology modeling

The aim of the plant ontology is to cover relevant knowledge of the plant and to provide it to smart services. We identified six categories in which the required knowledge could be divided into, as shown in Figure 1. The following list gives some details about the knowledge.

- **Device Description:** Types, Signal Names, Capabilities
- **Topology:** Order of Modules and Devices, Network
- **Process Knowledge:** Actions, Change of Products, KPIs
- **Physics:** Time, Energy, Mass, Chaining, Units
- **Behavior:** Production, Ramp-up, Maintenance, Idle
- **Dependencies:** Causalities, Relations

Obviously, there are many other schemas of how knowledge can be classified. Another important classification is the differentiation between type knowledge and instance knowledge. Type knowledge, also known as T-Box (terminological component) holds true for all automation systems, such as physics, behavior, and partly process knowledge. Instance knowledge, also known as A-Box (assertion component), holds true only for a single system, such as topology, device description, dependencies and partly process knowledge. If the system changes, e.g. by integrating a new module, only the instance knowledge has to be adapted.

Together the present paragraphs answer RQ 1 by showing a concept of how smart services can be integrated. Instance knowledge can be covered during the engineering phase in AML. Afterwards, it is transformed to an OPC UA model to provide an access for other services, see Fig. 2. Smart services are located at the RAMI4.0, see Fig. 3. Because of performance reasons, the smart services are performed on the IT-Infrastructure, whereas the knowledge can be part of the AAS, see Fig. 4.

D. *Ontology integration to OPC UA*

This section presents an approach for the representation of OWL DL ontologies in an OPC UA information model to answer RQ 2. Since no suitable companion specification could be identified, the initial OPC UA specification is used.

The general approach of the OPC UA information model and ontologies are similar. Both can be represented based on nodes, or triples. But, there are three main differences in the modeling: *(i)* OPC UA can represent hierarchies on type and object level whereas OWL represents them only on instance level (please note: OPC UA types are a synonym for OWL classes or concepts and OPC UA objects are known as instance or individual.); *(ii)* class description in OWL cannot be represented in OPC UA information model directly; *(iii)* not all characteristics and none of the object properties descriptions of OWL can be represented in an OPC UA information model natively. However, since OPC UA provides freedom in modeling, it is possible to bridge differences and to provide a transformation from OWL to OPC UA. The transformation has to be bijective, so that it can be used to create an ontology out of the information model and vice versa.

1) Class Hierarchy: OPC UA can represent the class hierarchy of OWL on its type level. An OWL instance can be modeled as OPC UA object. Since it is not possible to order OWL instances hierarchically, this OPC UA function is not used. Instances of both formalisms have a type property that refers to the class or object type definition, but OWL instances can have multiple class assignments, whereas OPC UA only supports single class assignments. This is the first identified modeling restriction. Table I shows the properties to describe a class in OWL and maps them to relations of OPC UA. By using these relations, the model can be transferred one-to-one, if all OWL individuals have just one type definition, because OPC UA supports only one relation of *HasTypeDefinition* for each object, which is a suitable restriction. Additionally, it is only suitable for non OWL full, because the OWL full profile does not distinguish between classes and instances, which cannot be represented in OPC UA.

Some general information about the ontology, such as version information or the ontology's Internationalized Resource Identifier (IRI), are modeled at the top node of the ontology. This enables the usage of short prefixes instead of long IRIs.

TABLE I
SOME REFERENCES CAN BE TRANSLATED DIRECTLY

OWL Property	OPC UA Reference
subClassOf	HasSubtype
Type	HasTypeDefinition
DatatypeProperty	HasProperty

2) Class Description: In OPC UA, OWL classes are defined as *Object Types*, which can be complex constructs. Three properties describe classes, namely *equivalent to*, *subclass of* and *disjoint with*.[2] The arguments of these properties are classes, property restrictions using quantifiers and logical expressions combined with classes and/or quantifier.

To provide a uniform model, the representation of class restriction and quantifier restriction should be the same, as well as the representation of combined expressions. This is achieved by adding a node class variable to the object type, which is a two dimensional array with four columns and n-rows. Four columns are needed if a negated quantifier expression has to be modeled. It is oriented at the OWL notation, which would be: *not(isRequiring some AirPressure)*. The model uses the variable type *SubClassOf*. The first column contains a *not*, the second column represent the relation, in this case *isRequiring*, the third column contains the quantifier's name *some*, and the fourth column contains the class *AirPressure*, as represented in Table II. For simple class representation, the same type of array but only necessary columns are used; all others remain empty.

TABLE II
ARRAY TO REPRESENT CLASS DESCRIPTIONS IN OPC UA

Negation	Relation	Quatifier	Class
not	isRequiering	some	AirPressure
	isPerforming	some	Heating

Combined expressions are multiple statements connected with logical *and* and *or*, such as $Corn \sqcap (Sugar \sqcup Salt)$. Therefore, a convention that all statements in one array are connected via logical *or* is made. A logical *and* is modeled by using the relation once again. This enables the expression of all formulas, but they have to be in the conjunctive normal form. It should be noted, there are two options to model the logical *and* in OWL, a relation can be used more times or a logical *and* can be used in a single statement. That difference gets lost using the transformation, but the expressed statement remains unchanged.

3) Property Characteristics: Two types of properties are available in OWL, named data properties and object properties. Data properties are used to bind data, e.g. a string, date or a float to an individual, whereas object properties are relations between individuals, e.g. hasInput, isPartOf, or hasUnit.

Data properties can be modeled as children of an individual, using an OPC UA variable, similar to OWL. Therefore, data

[2]The variable type for disjoint classes were named *disjointWithClass*, because the name *disjointWith* is used for the object property description.

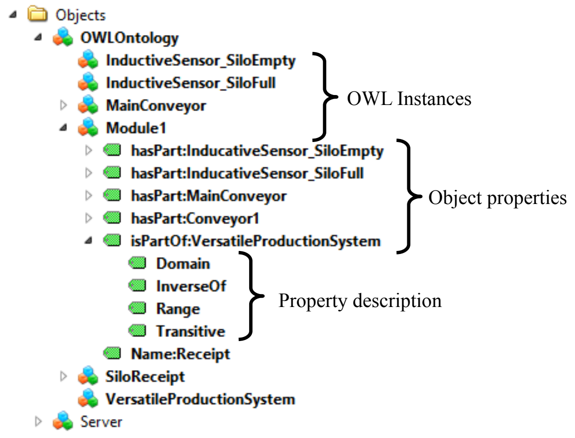

Fig. 5. Transformed OPC UA information model from an ontology

properties of the ontology have to be modeled as a *Variable Type* in OPC UA, with the same data type as in OWL.

Object properties have some characteristics, namely *functional*, *inverse functional*, *symmetric*, *asymmetric*, *reflexive* and *irreflexive*, which have to be presented in the model. Additionally, object properties have a description, namely *equivalent to*, *subProperty of*, *inverse of*, *domain*, *range*, *disjoint with* and *superProperty of*, which have to be represented. Typically, such properties would be represented by references, which would match OWL exactly. But it is not possible to express the property description or characteristics of a *Reference Type* in OPC UA, except the *symmetry* and the *inverse of*. Since this is not sufficient, these OPC UA functions are not used to get a homogeneous presentation of characteristics. Instead properties are modeled as *Variable Types*. The characteristics of the property are presented as a variable of the type *boolean*, which is true if the property has the characteristic. If the property is false or not modeled, the property does not have that characteristic. Descriptions of properties are modeled with the data type *string*, which contains the content of the description, e.g. the range. The disadvantage is that it is not linked in the model, it is just a string which has to match an instance to be consistent. Nevertheless, the information model is easy to understand for humans, because all descriptions and restrictions of an instance are presented directly at the node.

An example is shown in Fig. 5, where the ontology of a small module is represented as OPC UA information model. Figure 5 shows instances, where module 1 is enlarged, so that all relations are shown. Only property characteristics, which are true, are modeled. The *isPartOf* relation is enlarged, so it can be seen which characteristics and descriptions are defined.

4) SWRL: The variability of SWRL rules is limited, which is a great advantage for the integration into OPC UA, because OPC UA has no initial construct which is able to represent rules. The general syntax of SWRL is: *antecedent* $\rightarrow$ *consequent*. Both, the antecedent and the consequent consist of conjunctive atoms. Typically an atom consists of a class assignment, a property assignment or built-ins. Class assignments have the form *C(x)*, which means that individual *x* has the type *C*. Property assignments have the form *P(x,y)*, which means that individual *x* is connected to individual *y* by the property *P*. Built-ins have the form *<type>(x,y,...)*, where *<type>* indicates the type of the built-in and *(x,y,...)* the involved instances. A problem is that the list of arguments is between one and infinity. For example *swrlb:add* sum the arguments from the second to the last argument.

One way to represent SWRL in OPC UA is to take each rule as a separate string. Even if there are no limitations in the standard, it can lead to problems in practical applications, because every device has an implementation-specific limit of characters of a string. The identified limitations are between 254 and 32,767 characters. Since the worst case should be taken into account, rules can be larger than 254 characters, even if the variables get meaningful names instead of single characters. Instead, a string array can be used, where every atom has its own string. The distinction between antecedent and the consequent could be made by a separate line, which only includes the arrow =>. An example rule is presented in Table III.

TABLE III
ONE DIMENSIONAL ARRAY TO REPRESENT AN SWRL RULE

$provideOutputTo(?x, ?y)$
$hasOutput(?x, ?z)$
$hasInput(?y, ?z)$
$\Rightarrow$
$validGoodsFlow(?x, ?y)$

To integrate rules to OPC UA, an *ObjectType* is created, which has at least one *DataType* SWRLRule, which is a one dimensional string array. More SWRLRules can be added, to represent the rules. Because the rules are an extension, they are modeled as a separate object. Fig. 6 shows the information model with OWLOntology and SWRL rules, the AAS is not shown in that picture.

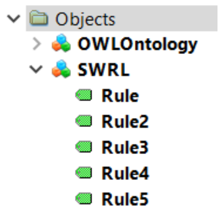

Fig. 6. Modelling of SWRL rules in OPC UA

E. Representation in AML

OPC UA is suitable to represent ontologies without many limitations in its information model, but it is just used for online purposes. AML should represent the instance knowledge during the engineering phase. Therefore, two possibilities to represent ontologies in AML are presented. First, by modeling the ontology in a CAEX-File and second, by setting a reference with a data connector. Both are explained and rated shortly. A CAEX-File consists of a instance hierarchy and three libraries, viz *SystemUnitClassLib*, *RoleClassLib* and *InterfaceClassLib* [23]. OWL classes can be defined in the *SystemUnitClassLib*. Every class needs an *InternalElement* to represent class descriptions through *Internal Link*. Every *Internal Link* has a class, which is defined in the *InterfaceClassLib*, similar to object properties in OWL. The *Internal Link* can have attributes and constraints, which allows a modeling of the characteristics. OWL individuals are represented in the *InstanceHierarchy* and have to be linked to a *SystemUnitClass*. Therefore, a representation of ontologies in CAEX is possible.

However, AML provides a much simpler way for the integration by linking a file with a *ExternalDataConnector*. This enables it to link OWL files without any transformation. It is suitable, because AML is used during the engineering phase and thus does not share the need of accessing with limited resources. In addition to that, the AML standard explicitly allows a further integration of XML standard formats, as for COLLADA and PLCopen XML [23]. The integration by linking the OWL-File is the smartest solution since no additional effort is required. Additionally, it can be standardized in the AML standard, if the usage of ontologies increases.

IV. Case-Study

The versatile production system (VPS) is a demonstrator in the SmartFactoryOWL, which was created as an adaptable production system. Due to its separate modules with similar electrical and mechanical interfaces, modules can be exchanged as needed for the actual production process. The VPS uses bulk good (corn) as input and produces popcorn. In the actual plant configuration it has four modules: delivery of goods (corn), storage, dosing and popcorn production. Additionally, there is a quality control module available, which sorts out bad corn. Each module is equipped with a separate PLC, so that a module just needs a trigger signal and input material; it acts independently of the modules nearby. Every PLC has its own OPC UA Server with an information model providing an interface to other modules and user control. The use-cases above realized with this demonstrator.

Case-Study 1 - Adaptable Factory: The configuration of production systems has to be performed quickly and reliably. For example, if the quality of the delivered corn is bad, the operator has to integrate the quality control module and reconfigure the modules according to the lower throughput of the new module. With traditional automation technology, the operator himself has to configure the parameters of all modules separately. If he is supported by smart services, the service will check whether the chosen configuration is valid and adapt the parameter automatically.

To enable this scenario, the modules have to provide knowledge about themselves. Smart services access the knowledge and process it with rules. The rule

$$provideOutputTo(x, y) \wedge hasOutput(x, z) \wedge \\ hasInput(y, z) \Rightarrow validGoodsFlow(x, y)$$

checks whether the input and output material of modules is valid or not. Likewise, such rules have to be defined for the mechanical and electrical interfaces of the modules, as well as for characteristics such as throughput. If all rules can be applied successfully, the operator gets a message that confirms the configuration and the operation can be started.

Case-Study 2 - Human-Machine-Interaction: Since the VPS is versatile, it is hard to develop a user interface that is suitable for all possible plant configurations. New human-machine-interaction technologies, such as a natural language interface [24], can be used to interact with the VPS. This can be realized with a smart service that processes the language and uses a knowledge base to determine answers: i.e. every natural language interfaces should have an underlying knowledge base to determine meaningful answers [25]. The service maps the data of the machine to a conceptual representation, which is needed for communication with humans. Therefore, the knowledge is used to derive the meaning of a user request, without an explicit mapping between the user request and the proposed signal name (a unique name which represents a device on the field bus).

The advantage for operators is that they can communicate naturally with the machine, independently of the actual configuration. Moreover, it is not important how familiar the operator is with the machine: The system can answer questions in colloquial language as well as specified questions of a service engineer. Typically, users request sensor values, such as the power consumption or key performance indicators, which are a combination of different values, such as overall equipment efficiency. Also, they can ask for the health state of the machine or information regarding the configuration.

Case-Study 3 - Value-Based Service: A downtime of a production system is always costly and has to be avoided. Therefore, VBS, such as predictive maintenance, are getting more and more important for the reduction of unplanned downtimes and the handling unplanned situations.

The diagnosis has to identify an anomaly and to derive the root cause of it, which can be performed by a smart service. The anomaly detection is done by common data-driven anomaly detection algorithms such as clustering [26], deep neural networks [27][28], or learned automata [29]. Knowledge-based services can be used to obtain a diagnosis [8]. If, for example, the scale of a module is a possible anomaly, it can be caused by the blow pipe which puts corn into the scale. Most users would identify the scale as root-cause, but this assumption is wrong. Since it is difficult to check the scale for the operator, it is time consuming to identify the real root-cause. The smart service uses the knowledge which is provided by the modules to derive possible root-causes and provide them to the operator; in our example three root-causes are possible. The operator can decide the order of checking them, since two of them are easy to check and thus save valuable time.

All case-studies require knowledge to perform their tasks in a high quality. To check whether or not the concept is suitable, the use cases were implemented in the SmartFactoryOWL respectively a simulation tool, to ease the reconfiguration. Ontologies from different modules could be accessed, merged and used from the smart service.

V. Conclusion

In this paper, the positioning of smart services in future automation systems and the compatibility of ontologies with existing standards used for Industrie 4.0 has been analyzed. Following answers could be made regarding the RQ:

RQ 1: RAMI 4.0 was introduced and smart services were located into it. OPC UA was chosen as technology, among others because it is feasible to represent the AAS by using the IEC 62541-100 standard. A concrete example is used to show how different elements are combined into the automation technology. Because large knowledge bases are expected, smart services are executed on an external IT infrastructure.

RQ 2: The commen OWL ontologies were analyzed and they can be integrated into the OPC UA standard. The only identified limitation is that OWL full ontologies cannot be represented in the OPC UA information model and the individuals can have just one type in the OPC UA representation, in this approach. For many practical applications, these limitations will be not a problem, because they do not use these modeling constructs. SWRL rules cannot be represented in OPC UA via a similar construct, but by representing them in a string array they are accessible. The modeling in AML is possible. Therefore, no limitations could be identified.

Both standards are able to integrate a representation of ontologies. AML provides the possibility to refer to ontologies with a universal data connector, which can be specified and integrated into the standard. The integration of ontologies into OPC UA can be represented in a companion specification.

References

[1] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, Nov 2014.

[2] H. Kagermann, W. Wahlster, and J. Helbig, Eds., *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry*. Frankfurt/Main: Secretariat of the Platform Industrie 4.0, 2013.

[3] O. Givehchi, K. Landsdorf, P. Simoens, and A. W. Colombo, "Interoperability for industrial cyber-physical systems: An approach for legacy systems," *IEEE Transaction in Industrial Informatics*, vol. 13, no. 6, pp. 3370 – 3378, 2017.

[4] S. Banerjee and D. Großmann, "Aggregation of information modelsan opc ua based approach to a holistic model of models," in *4th International Conference on Industrial Engineering and Application*, 2017.

[5] P. Adolphs, S. Auer, H. Bedenbender, M. Billmann, M. Hankel, R. Heidel, M. Hoffmeister, H. Huhle, M. Jochem, M. Kiele-Dunsche, G. Koschnick, H. Koziolek, L. Linke, R. Pichler, F. Schewe, K. Schneider, and B. Waser, "Structure of the administration shell: Continuation of the development of the reference model for the industrie 4.0 component," Federal Ministry for Economic Affairs and Energy (BMWi), Tech. Rep., April 2016.

[6] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao, "Answering natural language questions by subgraph matching over knowledge graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, 2017.

[7] V. Ebrahimipour and S. Yacout, "Ontology-based schema to support maintenance knowledge representation with a case study of a pneumatic valve," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 4, pp. 702 – 712, 2015.

[8] A. Mallak, C. Weber, M. Fathi, and A. Holland, "Active diagnosis automotive ontology for distributed embedded systems," in *IEEE European Technology and Engineering Management Summit (E-TEMS)*, 2017.

[9] M. Stenmark, J. Malec, K. Nilsson, and A. Robertsson, "On distributed knowledge bases for robotized small-batch assembly," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 519–528, April 2015.

[10] R. Anderl, K. Bauer, B. Diegner, J. Diemer, A. Fay, J. Firtz, D. Goericke, J. Grotepass, C. Hilge, J. Jasperneite, J. Kalhoff, U. Kubach, I. Lwen, G. Menges, M. J. Stefan, W. Mnch, H. Prei, O. Reus, F. Schmidt, E.-J. Steffens, T. Stiedl, M. ten Hompel, and C. Zeidler, "Aspects of research roadmap in application scenarios," Plattform Industrie 4.0, Tech. Rep., 2016.

[11] H. Bedenbender, A. Bentkus, U. Epple, T. Hadlich, R. Heidel, O. Hillermeier, M. Hoffmeister, H. Huhle, M. Kiele-Dunsche, H. Koziolek, S. Lohmann, M. Mendes, J. Neidig, F. Palm, S. Pollmeier, B. Rauscher, F. Schewe, B. Waser, I. Weber, and M. Wollschlaeger, "Industrie 4.0 plug-and-produce for adaptable factories: Example use case definition, models, and implementation," Federal Ministry for Economic Affairs and Energy (BMWi), Tech. Rep., June 2017.

[12] T. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal Human-Computer Studies*, vol. 43, no. 5-6, pp. 907–928, 1995.

[13] R. Chen, Z. Zhou, Q. Liu, D. T. Pham, Y. Zhao, J. Yan, and Q. Wei, "Knowledge modeling of fault diagnosis for rotating machinery based on ontology," in *13th International Conference on Industrial Informatics (INDIN)*, 2015.

[14] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. (2004, may) Swrl: A semantic web rule language combining owl and ruleml. Website. W3C. [Online]. Available: http://www.w3.org/Submission/SWRL/

[15] *OPC unified architecture - Part 2: Security Model*, IEC Std. IEC TR 62 541-2:2016, 2016.

[16] O. Gräser, L. Hundt, M. John, G. Lobermeier, A. Lüder, S. Mülhens, N. Ondracek, M. Thron, and J. Schmelter, "White paper - automationml and ecl@ss integration," Common Working Group of AutomationML e.V and eCl@ss e.V., Tech. Rep., November 2015.

[17] *OPC Unified Architecture Information Model for AutomationML*, AutomationML consortium, OPC Foundation Std., Rev. V 1.0.0, March 2016.

[18] K. Bauer, J. Diemer, C. Hilger, U. Löwen, and J. S. Michels, "Benefits of application scenario value-based service," Federal Ministry for Economic Affairs and Energy (BMWi), Tech. Rep., April 2017.

[19] H. Kagermann, R. Anderl, J. Gausemeier, G. Schuh, and W. Wahlster, "Industrie 4.0 in a global context," acatech National Academy of Science and Engineering, Tech. Rep., 2016.

[20] K. Bauer, T. Bauernhansl, R. Henkel, J. Jasperneite, U. Löwen, G. Menges, J.-P. Meyer-Kahlen, J. S. Michels, R. Rauh, O. Stiefenhofer, M. ten Hompel, C. Zeidler, H. Szigeti, J. Sreng, and E. Truffet, "Plattform industrie 4.0 & industrie du futur: Common list of scenarios," Plattform Industrie 4.0 and Alliance Industrie du Futur, Tech. Rep., 2016.

[21] 2017. [Online]. Available: https://fdtgroup.org/rami-4-0-identifies-fdt-standard-for-the-connected-world/

[22] W. Dai, V. Dubinin, J. Christensen, V. Vyatkin, and X. Guan, "Towards self-manageable and adaptive industrial cyber-physical systems with knowledge-driven autonomic service management," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2016.

[23] AutomationML, "AutomationML Part 1 - Architecture and general requirements, V2.0.0," www.automationml.org, April 2016.

[24] A. Bunte, A. Diedrich, and O. Niggemann, "Integrating semantics for diagnosis of manufacturing systems," in *21th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Berlin, Sep 2016.

[25] B. J. Huber Jr., "A knowledge-based approach to understanding natural language," Ph.D. dissertation, Rochester Institute of Technology, 1991.

[26] A. J. Torabi, M. J. Er, X. Li, B. S. Lim, and G. O. Peen, "Application of clustering methods for online tool condition monitoring and fault diagnosis in high-speed milling processes," *IEEE Systems Journal*, vol. 10, no. 2, pp. 721–732, June 2016.

[27] N. Hranisavljevic, O. Niggemann, and A. Maier, "A novel anomaly detection algorithm for hybrid production systems based on deep learning and timed automata," in *International Workshop on the Principles of Diagnosis (DX)*, Denver, Oct 2016.

[28] H. Hu, B. Tang, X. j. Gong, W. Wei, and H. Wang, "Intelligent fault diagnosis of the high-speed train with big data based on deep neural networks," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2017.

[29] A. Maier, "Identification of timed behavior models for diagnosis in production systems," Ph.D. dissertation, University of Paderborn, 2015.