Ontology-Based OPC UA Data Access via Custom Property Functions

Gernot Steindl, Thomas Frühwirth, Wolfgang Kastner
Institute of Computer Engineering, Research Unit Automation Systems
TU Wien
Vienna, Austria
{gernot.steindl, thomas.fruehwirth, wolfgang.kastner}@tuwien.ac.at

***Abstract*—Cyber Physical Production Systems have a need of sharing and interlinking information and knowledge over different domains. In the area of industrial automation, OPC Unified Architecture (OPC UA) is a widely used and established standard for communication and information modeling. We propose an ontology-based OPC UA data access method utilizing custom property functions, which enables interlinking between OPC UA information and other factory data. To avoid duplicated data and to reduce the communication overhead in the proposed method, the OPC UA run-time data are loaded on-demand and are not persistently stored in the triplestore. To enable fast and easy ontology-based access and interlinking of the OPC UA information, the needed ontology is automatically generated from the OPC UA information model. A proof of concept demonstrates the application of our approach for a laboratory use-case of a Packed-Bed Regenerator.**

***Index Terms*—OPC Unified Architecture, Semantic Web, Ontology, Cyber Physical Production Systems**

I. Introduction

The process of digitalization in industry is an ongoing challenge for industry itself as well as for academia. This transformation is often called the fourth industrial revolution or Industry 4.0. The main goals of Industry 4.0 are optimization and customization of production as well as enhanced automation and adaption [1]. This claims benefits for increasing flexibility of the whole production system, which is a composition of human resources, production equipment, and aggregated products that interact over cyber-physical interfaces [2]. Such production systems are referred to as Cyber-Physical Production Systems (CPPSs).

CPPSs can be structured into five levels, which is called the 5C architecture [3] and depicted in Figure 1. This architecture consists of a connection, conversion, cyber, cognition, and configuration level. The Semantic Web Stack, including the Resource Description Framework (RDF), the Resource Description Framework Schema (RDFS) and the Web Ontology Language (OWL) as well as SPARQL Protocol and RDF Query Language (SPARQL) can be applied on the upper levels of this 5C architecture to implement and enhance its functionality. Semantic Web technologies can be used to semantically enrich industry data, integrate data from heterogeneous sources, and increase interoperability. With the help of Semantic Web technologies engineering and run-time information can be interlinked to form a so-called ontology-based Linked Factory Data [4]. A key aspect of Linked Factory Data is to keep the data in their natural format and storage, which can be achieved with the help of various types of ontology-based data integration methods [5].

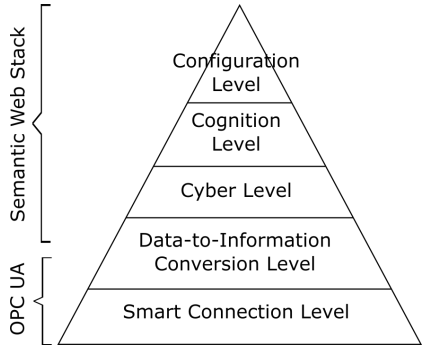


Fig. 1: Levels of the 5C architecture for CPPS [3] and the application of OPC Unified Architecture (OPC UA) and Semantic Web technologies

In the area of industrial automation, OPC UA is a well-established standard for industrial communication. It specifies the data exchange and facilitates semantic interoperability by use of an extensible information model. Thus, OPC UA is a framework for object-oriented data and information representation and exchange [6]. As it is widely used and also recommended by the Industry 4.0 initiative [7], the integration into Linked Factory Data is addressed in this paper. Therefore, the transformation of the OPC UA information model into an OWL ontology as well as an ontology-based data access method is presented, which both are prerequisite for OPC UA integration into Linked Factory Data.

Usually, the information in an ontology is stored in a graph-based triplestore and is not changing frequently over time. In industrial processes, run-time data, like sensor data or state conditions, are generated in high volume and velocity. Triplestores are not well suited for large time series data as it will reduce the performance of SPARQL queries [8], because the query engine has to perform a pattern matching over the whole graph in the triplestore. Therefore, we propose an ontology-based data access method, which uses custom

978-1-7281-0303-7/19/$31.00 ©2019 IEEE

property functions in SPARQL to retrieve run-time data and time series data on-demand and without a persistent storage of the data in the triplestore. To reduce the engineering effort and to enable an easy ontology-based OPC UA data access, an automatic transformation of the OPC UA information model into an OWL ontology is performed.

The remainder of this paper is structured as follows: Section II is giving a short introduction into the OPC UA information model and Semantic Web technologies. Also, a literature review of Semantic Web applications in combination with OPC UA is presented. Section III explains the proposed ontology-based OPC UA data access method as well as the automatic ontology generation process. Section IV presents a laboratory use case of a Packed-Bed Regenerator, which is used to demonstrate our approach. In Section V, a conclusion is drawn and ideas for future work are discussed.

II. State of the art

This section gives a short overview of OPC UA and its information modeling as well as some introduction to the Semantic Web technologies used in this work. Also previous work is reviewed which already applies Semantic Web technologies in combination with OPC UA.

A. OPC UA Information Model

A key feature of OPC UA is its information modelling capability. It allows to describe everything from simple devices (sensors, actuators, etc.) to very complex components (production machinery, energy storage devices, etc.) in an object-oriented and semantically meaningful way [9]. The information model is thereby built from *nodes*, representing objects, variables, etc. and *references*, representing relations between nodes. OPC UA defines eight different types of nodes, so-called *node classes*. They are briefly summerized in Table I.

Depending on its node class, each node has a set of *attributes*. Most importantly, each node has a unique *NodeId*, which consists of a *NamespaceIndex* and an *Identifier*, e.g. *"id=2;s=Heater"*. Other attributes are the *NodeClass* (one of the entries of Table I), *DisplayName* (a human-readable name for the node), *Description* (a human-readable description of the nodes's purpose), possibly a *Value*, and many more. The information model exposed by a specific OPC UA server is called the server's *address space*. OPC UA clients use *services* to interact with the server, e.g. the *Browse* service to navigate through the address space, the *Read/Write* service to read/write variable values, the *HistoryRead/HistoryUpdate* service to read/update historic values of variables, the *Call* service to invoke methods, etc. The historical values themselves are not visible in the OPC UA address space of the server.

TABLE I: OPC UA node classes

Node class	**Comment**	**Example**
Object Type	The ObjectType can be used to model complex objects. Typically, these objects expose some internal structure.	HeaterType
Object	An Object is an instance of the corresponding ObjectType, just like objects in programming are instances of their corresponding class.	Heater
DataType	DataTypes are typically simple types such as String, Boolean, Float, Int32, etc. However, they can also have a more complex internal structure if needed.	Float
Variable Type	VariableTypes are used to model the value and structure of data. Additional information, e.g. the EngineeringUnit, can be added. The value is of a specific DataType.	AnalogItemType
Variable	A Variable is an instance of the corresponding VariableType.	Temperature
Reference Type	The ReferenceType node class is used to define the references between nodes and their semantics.	HasComponent
Method	Methods can be called by an OPC UA client. Input arguments, as well as output arguments, are supported.	SetSetpoint()
View	Views can be used to structure and filter the information in a user-group-specific way.	OperatorView

B. Semantic Web Technologies

In the context of Semantic Web, various technologies are standardized under the lead of the World Wide Web Consortium (W3C)[1]. One of the base technologies is called RDF [10], which can be used to model information by creating statements about resources. These statements consist of a subject, a predicate and an object, and, therefore, are called triples. As the object of such a statement can be used as a subject in another statement, the information is represented as a graph with interlinked resources. Special databases, called triplestores, are used for this kind of data.

RDFS [11] and OWL [12] are both formal knowledge representation languages. RDFS is an extension of the basic RDF and provides vocabulary to create hierarchies of classes and properties. OWL extends RDFS with further language constructs, like cardinalities, value restrictions, characteristics of properties, and complex class construction and is based on description logic. Thus, RDFS and OWL enables reasoning for the Semantic Web.

With the help of these technologies and standards, so-called ontologies can be created. In computer science, ontologies are an explicit specification of a conceptualization, where a conceptualization is a simplified abstract model of a certain domain [13]. In addition, ontologies in computer science should be machine readable. Thus, an ontology can be defined as *"... a formal, explicit specification of a shared conceptualization"* [14].

To retrieve information from such an ontology, SPARQL [15] was designed to query RDF data including RDFS and OWL constructs. It supports a *SELECT* statement to perform a graph-based pattern matching over the RDF graph and to retrieve data in a table-based fashion. It also supports the *CONSTRUCT* clause, which allows to extract

[1]https://www.w3.org

information from the ontology and to create new RDF graphs based on these data.

Ontologies can be used in different fields of application, like communication, interoperability, as well as information sharing and reuse [16]. In the context of CPPS, data and information integration is one of the key applications, where ontologies are applied. Ontology-Based Data Access (OBDA), as the foundation of Ontology-Based Data Integration (ODBI), enables semantic enrichment of available data from heterogeneous sources to create a semantic integration layer, which is able to provide a higher level of abstraction [17].

Since in the manufacturing domain, data is hardly available in the RDF format, mappings and transformations from existing data sources have to be performed [18]. For relational databases, frameworks for OBDA, like Ontop exist [19] and have already been applied for industrial use cases [20]. Similar concepts have to be investigated for OPC UA, which is a common data source in an industrial environment [6].

C. Combining Semantic Web Technology and OPC UA

Semantic Web technology in combination with OPC UA can be used in CPPSs for various applications, like creating flexible orchestration plans in manufacturing [21] or semantic data integration and analysis like conceptually shown in [22].

Semantic Web technologies are also used to harmonize the access and utilization of industrial devices. Therefore, a conceptional architecture with an OPC UA adaption layer is presented in [23], to perform a transformation of data exchange structures into RDF. The transformation itself is not described in detail in this work, because it primarily focuses on the creation of a plant model ontology.

A more concrete implementation strategy of combining OPC UA and Semantic Web for an ontology-based OPC UA data access can be found in [24]. Thereby, Semantic Web technology is used in combination with OPC UA for sensor discovery. A semantic access layer is implemented to tap the full potential of ontologies while not affecting the existing OPC UA standard. The data are retrieved by a subscription handler if available. Otherwise, a sensor data request is triggered frequently to map the OPC UA data to a sensor ontology. In [25], the OPC UA ontology-based data access is enabled by mirroring the OPC UA data into a relational database and to map this data into an ontology. This architecture is chosen to enable real-time processing of a large amount of industrial data. The stored data in this database are used in combination with the ontology to perform reasoning periodically to generate new knowledge dynamically. Both of these approaches to ontology-based OPC UA data access have the disadvantages of duplicating data or communication overhead, if the information is frequently polled.

To directly access the OPC UA data, a so-called linked data adapter is implemented in [26]. This adapter provides data from an OPC UA server as RDF via a REST interface. The base functionality of OPC UA is mapped to HTTP GET, POST, PUT and DELETE requests. As the adapter returns RDF data, URI dereferencing can be used in SPARQL to access these data on-demand.

Our proposed approach of ontology-based OPC UA data access is a direct extension of the SPARQL query engine, by implementing so-called custom property functions. This extension implements an efficient data extraction mechanism, as it only retrieves the necessary data from an OPC UA server, if a SPARQL query is invoked, which needs these specific data. This enables data access on-demand, but waives an additional REST mapping.

III. Proposed Ontology-based OPC UA Data Access Method

To enable ontology-based OPC UA data access, an OWL model has to be created and mapped to the OPC UA information model. To reduce engineering effort, the OPC UA information model, which is stored in the OPC UA server, is extracted by a software module and automatically transformed into OWL. This needs only to be performed once or if the OPC UA information model of the server has changed.

To implement the proposed ontology-based OPC UA data access method, a Semantic Web framework has to be used which supports custom property functions. Inside these functions, the OPC UA server is accessed and its run-time data is retrieved. For our proof of concept, the Apache Jena Framework[2] is used. It is able to handle the OWL data and its SPARQL query engine ARQ can be extended with custom property functions. For the OPC UA communication the Eclipse Milo framework[3] is chosen to implement an OPC UA client inside the ARQ extension. To test the implementation, an Apache Jena Fuseki server is configured to load the ARQ extension and to provide a SPARQL web interface for exploring the data in the triplestore. Figure 2 gives an overview of the used software modules of our proof of concept. The OPC UA ontology extraction as well as the custom property function implementation are described in more detail in the following subsections.

A. OPC UA Ontology Extraction

The components and steps involved in generating OWL models from the address space of an existing OPC UA server are illustrated in Figure 3. First, the OPC UA client connects to the OPC UA server and starts analyzing the address space by reading the OPC UA server's *NamespaceArray*, which associates each *NamespaceIndex* to its corresponding *NamespaceUri*. This is required because each *NodeId* only contains the *NamespaceIndex*, while in Semantic Web technologies each resource is identified by ⟨*Namespace*⟩#⟨*Identifier*⟩. Next, the OPC UA client recursively browses the address space of the OPC UA server and creates an RDF model for each *Namespace*. It thereby maps OPC UA nodes to RDF resources and OPC UA references to RDF properties. Additionally, for each reference between two OPC UA nodes, an RDF statement is created in the model. If such a statement involves resources

[2]https://jena.apache.org

[3]https://projects.eclipse.org/proposals/milo

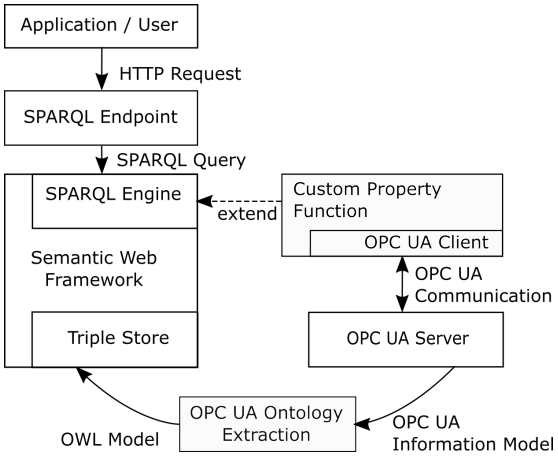


Fig. 2: Software Modules of the proof of concept

of different namespaces, and thus different models, it is added to the model corresponding to the namespace with the higher index, according to the *NamespaceArray*. This way, the model corresponding to namespace with *NamespaceIndex* 0 only contains statements about namespace 0, namespace 1 contains statements about namespace 1 and 0, etc. Finally, attributes in OPC UA (such as *NodeId*, *BrowseName*, etc.) are added to the corresponding resources by means of literals.

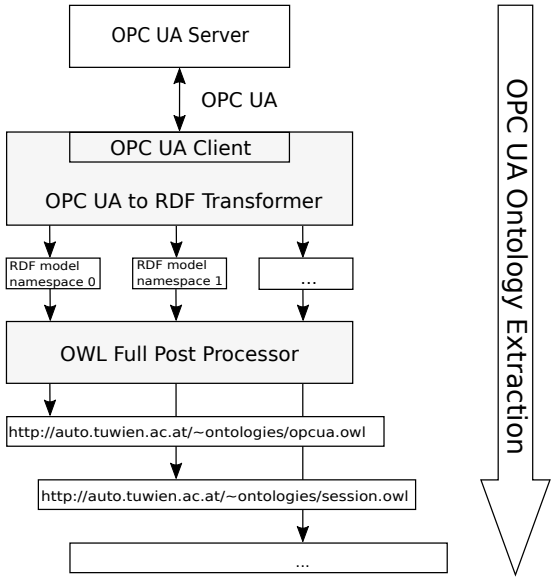


Fig. 3: OPC UA ontology extraction process

The result of this transformation step is a set of models, which only use the vocabulary provided by RDF. Additional vocabulary of a more specific ontology language, e.g. RDFS, OWL DL or OWL Full, is then added in the post processing step. As different ontology languages provide different modelling concepts, the resulting RDFS/OWL models may deviate quite substantially from the initial OPC UA address space. A very prominent challenge in this regards is the capability of OPC UA to arbitrarily create references between objects (corresponding to individuals in most ontology languages) and types (corresponding to classes in most ontology languages). Only OWL Full allows to use object properties for relating individuals and classes with similar flexibility. This short discussion already indicates that the post processing step requires many design decisions and compromises to be made. As a more expressive ontology language enables the creation of models that better represent the OPC UA address space, a post processor for OWL Full has been developed in a first attempt. The post processor currently implements the following transformation rules and can easily be extended:

- An rdfs:label is created for each resource from the OPC UA node's *DisplayName*.
- All resources corresponding to an OPC UA node of node class *ObjectType*, *VariableType*, or *DataType* are declared as owl:Class.
- All resources corresponding to an OPC UA node of node class *ReferenceType* are declared as owl:ObjectProperty.
- All resources corresponding to an OPC UA node being the source of a *HasTypeDefinition* reference are declared as individual of the corresponding class.
- All resources corresponding to an OPC UA node being the source of a *HasSubtype* reference are declared as superclass of the corresponding class.
- All object properties, for which the corresponding OPC UA *ReferenceType*'s attribute *Symmetric* is set, are declared as symmetric property.
- For all object properties, for which the corresponding OPC UA *ReferenceType*'s attribute *InverseName* is set, an inverse object property is created.
- All properties relating OPC UA nodes to their OPC UA attributes are declared as annotation properties.

While not always being respected, it is best practice in ontology engineering to use namespaces that actually correspond to ontology documents accessible via the web. This idea is not present in OPC UA, and, thus, the namespaces extracted from the OPC UA server's *NamespaceArray* have no meaning except for being unique. For this reason, the post processor allows to substitute namespaces with user-defined replacements, e.g. `http://opcfoundation.org/UA/` is substituted with `http://auto.tuwien.ac.at/~ontologies/opcua.owl`.[4] This completes the OPC UA ontology extraction process.

B. Ontology-based OPC UA data access method

As most of the run-time data that are accessible through the *Read* or *HistoryRead* service are changing over time, they should not be statically stored in the ontology. An ontology-based data access method has to be provided which loads the OPC UA data on-demand to avoid a periodical or event-based update of the ontology and to keep the triplestore as small as possible.

[4]The resulting ontology documents are available in RDF/XML format at http://auto.tuwien.ac.at/~ontologies/opcua.owl, http://auto.tuwien.ac.at/~ontologies/session.owl, and http://auto.tuwien.ac.at/~ontologies/packed-bed-regenerator.owl.

We have implemented such an ontology-based data access for the OPC UA *Read* and *HistoryRead* service as an extension to a SPARQL query engine. SPARQL is a very flexible language which allows to implement custom property functions to add functionality. Two custom property functions were implemented, namely *value* and *histValue*. These functions use the information stored in the ontology to retrieve the OPC UA endpoint url and the OPC UA node ID to connect to the OPC UA server and request the required data. These two custom property functions are registered within the `http://auto.tuwien.ac.at/~ontologies/opcua.owl` namespace.

The *value* property function can be applied on every OPC UA node in the ontology. If for that node the *Read* service is available, the data are retrieved during the SPARQL query and the value is assigned to the SPARQL variable. Listing 1 shows a SPARQL query which uses the custom property function *value*.

The *histValues* property function returns the timestamp and its related value and binds them to the specified SPARQL variables. As additional parameter, the start and end time of the *HistoryRead* service can be defined. If no end time is defined, the current time is used as default value. Table II illustrates the implemented function overloadings of *histValues*. The usage of the custom property function *histValues* is shown in Listing 2

TABLE II: Overloading of the Custom Property Function *histVlaues*

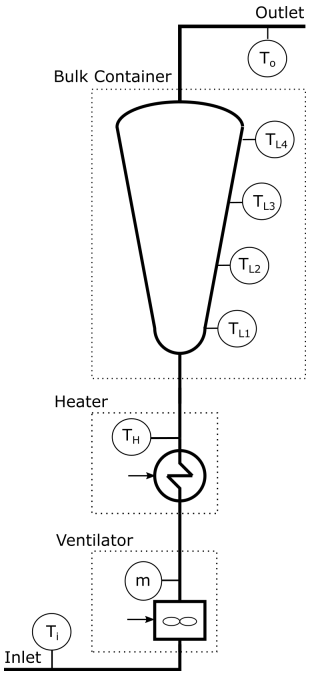
Function Signature	**Behaviour**
histRead(?timestamp ?value)	Retrieves all available data
histRead(?timestamp ?value int n)	Retrieves the n last values till now
histRead(?timestamp ?value ”YYYY-MM-DD hh:mm:ss”)	Retrieves all data since the specified date
histRead(?timestamp ?value ”YYYY-MM-DD hh:mm:ss” ”YYYY-MM-DD hh:mm:ss”)	Retrieves all data between the first and second date-string

SPARQL can not only be used for querying, but also for updating or inserting new data. For example, if certain OPC UA run-time data should be stored persistently in the ontology, the SPARQL *INSERT* command can be utilized.

IV. Use Case - Packed-Bed Regenerator

As use-case for the ontology-based OPC UA data access, a model of a Packed-Bed Regenerator (BPR) test rig is chosen, which is located at the laboratory of the Institute for Energy Systems and Thermodynamics (IET) at TU Wien (Figure 4b). The BPR is a thermal energy storage, which has a conic steel container filled with gravel as storage medium and surrounded by an insulation. Ambient air is used as a heat transfer fluid. It gets heated by a electric heater and transported through the tank during the charging phase. The hot air heats up the gravel inside the tank which stores the energy. For discharging, cold air is ventilated through the hot gravel. The schematic of the BPR and its components are depicted in (Figure 4a)

The BPR consists of a ventilator, a heater and the bulk container. It has various temperature sensors installed as well



(a) Schematic diagram of the Packed-bed Regenerator

(b) Test rig at the laboratory of TU Wien [27]

Fig. 4: Packed-bed Regenerator

as a mass flow sensor. The bulk container has four temperature sensors at different levels to track its state of charge. An OPC UA information model has been created which is partly shown in Figure 5. This model is automatically transformed into OWL, as explained in previous Section. With this OWL model and the implemented custom property function *value* and *histValues*, ontology-based OPC UA data access is performed, by sending SPARQL queries to the Apache Jena Fuseki Server.

Listing 1 shows a SPARQL query which retrieves the current value of the temperature sensor T_{L1} in the bulk container of the BPR. The SPARQL query engine connects to the OPC UA server in the background, requests the data, and assigns it to the SPARQL variable *?temp*. The answer to this query is also shown at the end of Listing 1.

```
1  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2  PREFIX uaBase: <http://auto.tuwien.ac.at/opcua.owl#>
3  PREFIX j.0: <http://auto.tuwien.ac.at/packed-bed-
       regenerator.owl#>
4
5  SELECT ?displayName ?temp
6  WHERE {
7      ?sensor rdfs:label "T_L1".
8      ?sensor rdfs:label ?displayName.
9      ?sensor uaBase:HasComponent/uaBase:value ?temp
10 }
```

```
|?displayName|?temp |
|""""""""""""""""""""""""""|
|"T_L1"      | 81.00|
```

Listing 1: SPARQL query using custom property function *value* and its corresponding answer

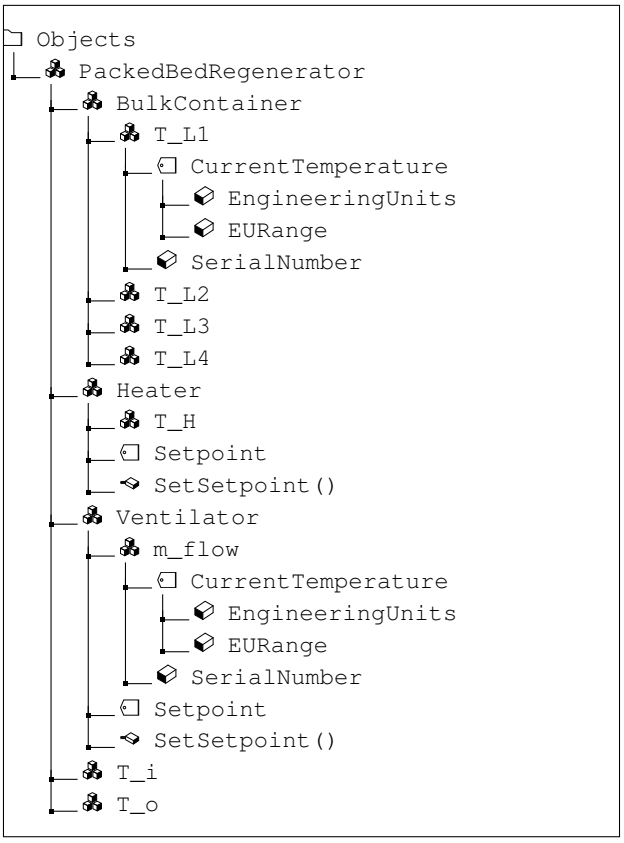


Fig. 5: OPC UA Information Model of the Packed-Bed Regenerator

Listing 2 shows a SPARQL query which retrieves the historical data (timestamp and values) of the temperature sensor T_{L1}, starting at 2019-03-21 10:00:00. As the retrieved data is assigned to a SPARQL variable, the values can be processed by the SPARQL engine. In the case of this query, a filtering of the values greater than "41.0" is applied. The corresponding answer to this query is also partly shown in Listing 2.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uaBase: <http://auto.tuwien.ac.at/opcua.owl#>
PREFIX j.0: <http://auto.tuwien.ac.at/packed-bed-
    regenerator.owl#>

SELECT ?displayName ?time ?temp
WHERE {
    ?sensor rdfs:label "T_L1".
    ?sensor rdfs:label ?displayName.
    ?sensor uaBase:HasComponent/uaBase:histValues (?time
     ?temp "2019-03-21 10:00:00").
    FILTER(?value > "41.0")
}
```

?displayName	?time	?temp
"T_L1"	"Thu Mar 21 10:00:00 CET 2019"	81.0
"T_L1"	"Thu Mar 21 10:00:01 CET 2019"	80.0
...	...	...
"T_L1"	"Thu Mar 21 10:25:52 CET 2019"	104.0

Listing 2: SPARQL query using custom property function *histValue* and its corresponding answer

In Listing 3 a SPARQL query is shown, which retrieves all temperature sensor values from the bulk container and calculates the average temperature. As the OPC UA type definitions are present in the *Objects* and the *Types* folder of the OPC UA information model, the query has to state that the *BulkContainer* is a component of the *PackedBedRegenerator*. Afterwards, all components of the *BulkContainer* which are from type *TemperatureSensorTypes* are retrieved and their current values are accessed by invoking the custom property function *uaBase:value*. The temperatures in the bulk container are $T_{L1} = 180\,°\text{C}$, $T_{L2} = 190\,°\text{C}$, $T_{L3} = 200\,°\text{C}$ and $T_{L4} = 210\,°\text{C}$. The answer to this query is an average temperature of $T_A = 195\,°\text{C}$, which is also shown as a result at the end of Listing 3. The calculation of the average temperature is performed by the SPARQL engine, as the retrieved data are assigned to the SPARQL variable *values*.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uaBase: <http://auto.tuwien.ac.at/opcua.owl#>
PREFIX j.0: <http://auto.tuwien.ac.at/packed-bed-
    regenerator.owl#>

SELECT (avg(?values) as ?averageTemp)
WHERE {
    ?regenerator rdfs:label "PackedBedRegenerator".
    ?regenerator uaBase:HasComponent+ ?bulkCont.
    ?bulkCont uaBase:HasTypeDefinition j.0:
     BulkContainerType.
    ?bulkCont uaBase:HasComponent+ ?tempSensors.
    ?tempSensors uaBase:HasTypeDefinition j.0:
     TemperatureSensorType.
    ?tempSensors uaBase:HasComponent/uaBase:value ?values
    }
```

?averageTemp
195.0

Listing 3: SPARQL query to retrieve the average temperature of the bulk container and its corresponding answer

V. Conclusion and Future Work

We have shown how ontology-based OPC UA data access can be implemented with the help of custom property functions and an automatic transformation of the OPC UA information model into OWL. The combination of Semantic Web technologies and OPC UA enables new possibilities in the context of CPPS, which could lead to higher flexibility in the production system. We have shown the application of the SPARQL query language to retrieve OPC UA data from a specified information model.

In this paper, we did not explore the reasoning capabilities of OWL, which can be beneficial for certain applications in the context of CPPS. Further research will be carried out to enable reasoning for the OPC UA ontology and combining it with logical rules. Also, a performance evaluation of our approach, regarding to query execution time and memory space, is planned as future work.

The presented proof of concept showed only a simple use case in an isolated environment. We believe that the full potential of the ontology-based OPC UA data access is only achieved, if other information, like environmental conditions,

production plans, device information, etc. are interlinked to build up Linked Factory Data. This will lead to new insights into the data and the production system itself.

ACKNOWLEDGEMENTS

This work has been partially supported and funded by the Austrian Research Promotion Agency (FFG) for the "PoSyCo - Power System Cognification" project under the contract number 867276, as well as the "SIC! - Smart Industrial Concept!" doctoral program.

REFERENCES

[1] V. Roblek, M. Meško, and A. Krapež, "A Complex View of Industry 4.0," *SAGE Open*, vol. 6, no. 2, apr 2016. [Online]. Available: http://journals.sagepub.com/doi/10.1177/2158244016653987

[2] L. Ribeiro and M. Bjorkman, "Transitioning from Standard Automation Solutions to Cyber-Physical Production Systems: An Assessment of Critical Conceptual and Technical Challenges," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3816–3827, 2018.

[3] J. Lee, B. Bagheri, and H. A. Kao, "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.mfglet.2014.12.001

[4] G. Steindl, B. Heinzl, and W. Kastner, "A Novel Ontology-Based Smart Service Architecture for Data-Driven Model Development," in *eKNOW 2019 - The Eleventh International Conference on Information, Process, and Knowledge Management*, J. L. Mauri, Ed., Athens, Greece, 2019, pp. 26–27.

[5] F. J. Ekaputra, M. Sabou, E. Serral, E. Kiesling, and S. Biffl, "Ontology-Based Data Integration in Multi-Disciplinary Engineering Environments: A Review," *Open Journal of Information Systms*, vol. 4, no. 1, pp. 1–26, 2017.

[6] P. Drahos, E. Kucera, O. Haffner, and I. Klimo, "Trends in industrial communication and OPC UA," *Proceedings of the 29th International Conference on Cybernetics and Informatics, K and I 2018*, vol. 2018-Janua, pp. 1–5, 2018.

[7] S. Baier, H.-D. Flick, T. Gast, A. Huger, H. Schleinkofer, H. Schmidtke, H.-J. Schneider, K.-H. Stoffels, F. Völker, and O. Zbikowski, "Guidlines. Industry Services - Technical services in the lifecycle of machines and plants," German Electrical and Electronic Manufacturers' Association, Frankfurt am Main, Germany, Tech. Rep., 2015. [Online]. Available: http://www.industry.siemens.com/services/global/en/Pages/home.aspx

[8] M. Martin, J. Unbehauen, and S. Auer, "Improving the performance of semantic web applications with SPARQL query caching," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6089 LNCS, no. PART 2, pp. 304–318, 2010.

[9] W. Mahnke and S.-H. Leitner, "OPC Unified Architecture - The future standard for communication and information modeling in automation," *ABB Review*, vol. 3, p. 2009, 2009.

[10] G. Klyne and J. J. Carroll, "Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation," World Wide Web Consortium (W3C), Tech. Rep., 2004. [Online]. Available: https://www.w3.org/TR/rdf-concepts/

[11] D. Brickley and R. Guha, "RDF Schema 1.1. W3C Recommendation," World Wide Web Consortium (W3C), Tech. Rep., 2014. [Online]. Available: https://www.w3.org/TR/rdf-schema/

[12] W3C OWL Working Group, "OWL 2 Web Ontology Language. Document Overview," World Wide Web Consortium (W3C), Tech. Rep., 2012. [Online]. Available: https://www.w3.org/TR/owl2-overview

[13] T. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *International Journal Human-Computer Studies*, vol. 43, pp. 907–928, 1995.

[14] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge Engineering : Principles and methods," *Data & Knowledge Engineering*, vol. 25, pp. 161–197, 1998.

[15] S. Harris and A. Seaborne, "SPARQL 1.1 Query Language. W3C Recommendation," World Wide Web Consortium (W3C), Tech. Rep., 2013. [Online]. Available: https://www.w3.org/TR/sparql11-query/

[16] M. Uschold and M. Gruninger, "Ontologies : Principles , Methods and Applications," *The Knowledge Engineering Revie*, vol. 11, pp. 93–136, 1996.

[17] D. Schachinger, W. Kastner, and S. Gaida, "Ontology-based abstraction layer for smart grid interaction in building energy management systems," *2016 IEEE International Energy Conference (ENERGYCON)*, pp. 1–6, 2016.

[18] B. Mörzinger, T. Weiler, T. Trautner, I. Ayatollahi, B. Angerer, and B. Kittl, "A large-scale framework for storage, access and analysis of time series data in the manufacturing domain," *Procedia CIRP*, vol. 67, no. January, pp. 595–600, 2018. [Online]. Available: http://dx.doi.org/10.1016/j.procir.2017.12.267

[19] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao, "Ontop: Answering SPARQL Queries over Relational Databases," *Semantic Web Journal*, 2017. [Online]. Available: https://github.com/ontop/ontop-examples/

[20] B. Mörzinger, M. Sabou, F. Ekaputra, and N. Sindelar, "Improving industrial optimizationwith Semantic Web technologies," in *SEMANTiCS 2018*, vol. 2198, 2018.

[21] 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.

[22] M. Obitko and V. Jirkovský, "Big Data Semantics in Industry 4.0," in *HoloMAS*, 2015, vol. 1, pp. 217–229. [Online]. Available: http://www.springer.com/series/1244http://link.springer.com/10.1007/978-3-319-22867-9{_}19

[23] D. Hastbacka and A. Zoitl, "Towards semantic self-description of industrial devices and control system interfaces," in *2016 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, mar 2016, pp. 879–884. [Online]. Available: http://ieeexplore.ieee.org/document/7474867/

[24] C. Legat, C. Seitz, and B. Vogel-heuser, "Unified Sensor Data Provisioning with Semantic Technologies," *ETFA2011*, pp. 1–8, 2011.

[25] S. Wang, J. Wan, D. Li, and C. Liu, "Knowledge reasoning with semantic data for real-time data processing in smart factory," *Sensors (Switzerland)*, vol. 18, no. 2, pp. 1–10, 2018.

[26] M. Graube, L. Urbas, and J. Hladik, "Integrating industrial middleware in linked data collaboration networks," *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 21, 2016.

[27] P. Drochter, "Auslegung, konstruktion und errichtung eines festbettregenerators," Master's thesis, TU Wien, Faculty of Mechanical and Industrial Engineering, Institute for Energy Systems and Thermodynamics, 2016.