DOI: 10.2507/34th.daaam.proceedings.xxx

AUTOMATIC TRANSFORMATION OF OPC UA SKILLS MODELS INTO KNOWLEDGE GRAPHS

Aleksandra Müller, Miray Özakkas, Oliver Petrovic, Werner Herfs, Christian Brecher

This Publication has to be referred as: Müller, A[leksandra]; Özakkas, M[iray]; Petrovic O[liver]; Werner, H[erfs] & Brecher C[hristian] (2023). Automatic Transformation Of OPC UA Skills Models Into Knowledge Graphs, Proceedings of the 34th DAAAM International Symposium, pp.xxxx-xxxx, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-xx-x, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/34th.daaam.proceedings.xxx

Abstract

As part of the Industry 4.0 initiative, OPC UA as a uniform standard for data and information exchange enables heterogeneous machine data to be transported and described so that other production participants can access it. In addition, control with the help of process-relevant capabilities, so-called OPC UA Skills, promises simple and flexible exchange and seamless integration of individual production participants and processes through uniform interfaces. In addition to interoperability, however, an Industry 4.0 network should be expandable by new information and provide possibilities for recognizing new connections of the existing data (reasoning). Information modelling based on the Semantic Web (e.g., by means of RDF) offers a solution for this. However, the already existing semantic OPC UA modelling on the shop floor is not directly interoperable with other semantic data models such as RDF/OWL. Furthermore, the OPC UA models lack query functionality, making them cumbersome to search. For these reasons, the focus of this thesis is on the transformation of flexible process control models based on OPC UA Skills into semantic RDF/OWL Knowledge Graphs. For this purpose, transformation rules are established, which are validated by means of application to a robot-based industrial application example.

Keywords: OPC UA Skills; semantic modelling; RDF; model transformation

1. Introduction

The goal of the Industry 4.0 research initiative is to enable globally networked, flexible and thus adaptable production [1]. In the vision of the Internet of Production (IoP) cluster of excellence, this requires the creation of a continuous network in which semantically adequate and context-dependent production data is available. One of the challenges here is the fact that production data is often heterogeneous and distributed. The OPC UA architecture as a uniform standard enables both a description of dynamic process data and a semantic modelling of the hardware and software structure of a production process with the help of the information models contained in the standard. In addition, the use of OPC UA enables cross-manufacturer data exchange between the production process participants [2]. In combination with the concept of a capability-based control with OPC UA Skills, a significant increase in the flexibility of the production process is made possible.

On the other hand, an IoP network should be expandable by new information and provide possibilities for recognising new connections of the existing data. Information modelling based on the Semantic Web is a suitable solution for this. However, the already existing semantic OPC UA modelling on the shop floor is not directly interoperable with other semantic data models such as RDF/OWL.

In this context, the integration of OPC UA with ontologies plays a decisive role. This allows the structures and information from the OPC UA information model to be transferred into an RDF model. In this way, with SPARQL as the query language, complex search queries can be made on RDF data in order to retrieve specific information easily. This facilitates the search and understanding of process data. By using OPC UA as the communication protocol and RDF as the data representation format data can be effectively managed, analysed and exchanged in the automation industry. There are already approaches that aim to transform OPC UA models into RDF, but these deal with the general methodology of OPC UA - RDF translation without considering the OPC UA Skills concepts. However, this is inevitable in order to map flexible control concepts in their fullness.

The focus of this thesis is on the transformation of OPC UA Skill information models into semantic RDF/OWL Knowledge Graphs. For this purpose, Section 2 presents existing approaches to the use of OPC UA Skills, semantic technologies in production and approaches to OPC UA - RDF translation. This is followed by a concept description of an OPC UA-RDF algorithm for the translation of control-relevant skills in Section 3. The algorithm is then prototypically implemented on an industrial robot-based demonstrator, described in Section 4. The thesis concludes with a summary and an outlook on further research possibilities.

2. State of the Art

2.1. OPC UA Skills

In industrial automation, OPC UA (Open Platform Communications Unified Architecture) has become the established standard for industrial communications, approved by RAMI 4.0 as uniform recommendation on the implementation of the communication layer. OPC UA specifies data exchange and enables semantic interoperability through more extensible information models. It provides a framework for object-oriented representation and exchange of data and information [3]. In addition, OPC UA introduces the concept of OPC UA Skills to enable flexible process control [4]. According to the Industry 4.0 platform, skills should be modelled as state machines, as this enables targeted monitoring and control of them [5]. The concept of OPC UA Skills allows complex processes to be decomposed into smaller units that can be executed in a specific sequence based on specific production requirements. This enables the adaptation of production systems for different processes, utilizing only the necessary functionalities for each process [6]. The implementation of OPC UA Skills allows precise control of the process flow, considering criteria such as production demand or energy efficiency. In this way, process control can be adapted to meet specific demands, leading to optimal performance. This requires an efficient analysis and processing of skills and process-related properties. Queries can be used to collect the relevant data to perform essential evaluations. However, there are currently no query tools available for extracting information from OPC UA documents [7].

2.2. Semantic Technologies in production

To model a knowledge base in a production environment, the available data needs to be extended by means of semantics [8]. Following the rules of Semantic Web, the data can be interpreted by a machine [9], establishing explicit relationships and creating an accessible knowledge repository for both machines and humans. For semantic enrichment of data, the Resource Description Framework (RDF) is used as a standardized syntax and core technology. To provide a consistent and declarative structure, the RDF Schema (RDFS) has been introduced to enable the semantic modelling of contexts. When merging many so-called RDF and RDFS statements, the resulting graph must be made readable to the machine. This is done through various serializations that specify the data format and thus the syntax [10]. The respective syntax can be used for the Ontology Web Language (OWL) as a data format and represents the essential OWL ontology. OWL enables the complex modelling of relations with additional features such as cardinalities or inverse relations [8]. Thus, ontologies using OWL create a suitable data modelling for complex relations and enable the intelligent derivation of knowledge [11]. This flexibility allows for adaptation and transferability to other data models. Additionally, SPARQL queries can be used to perform complex searches on RDF models to extract valuable insights and enable sophisticated data analysis.

Despite the potential benefits of semantic integration, the lack of standardization of Semantic Web Technologies presents challenges. There are some academic and industrial approaches to implement them in production, e.g. [12], [13], [14] and many more, but the vast majority of them lack standardization of the models. In this regard, the combination of OPC UA and Semantic Web Technologies can be promising to overcome the limitations of OPC UA for semantic descriptions in production environments. However, a conceivable parallel semantic modelling in the form of an OPC UA information model and an RDF graph, which are created independently of each other but model the same concept, proves to be unfavorable unless the additional effort of synchronizing both models can be automated.

2.3. Related work: OPC UA RDF Transformation

Previous studies have already investigated the integration of OPC UA with semantic modelling technologies. In [15] an approach for the integration of OPC UA into a Linked Data Environment is described. A new ontology is created specifically for the OPC UA information model and the built-in concepts of OPC UA are not used to their full extent. In [16], a mapping from OWL to OPC UA is proposed. However, this mapping does not consider every single concept of OPC UA, such as ReferenceType. In [17] it is pointed out that Schiekofer et al. in [7] presented the first complete formal mapping between OPC UA and RDF. Their work shows that a trivial mapping between OPC UA and OWL DL is not possible because the expressiveness of OWL DL is not sufficient to represent the entire OPC UA information model. Moreover, in [18] Steindl et al. proposed an alternative mapping from OPC UA to OWL by defining mapping rules to extract the required information from the OPC UA information model and use it in a domain-specific target ontology.

These previous studies lay the groundwork for developing an interoperable connection between OPC UA and Semantic Web Technologies. They emphasize the significance of a comprehensive mapping of OPC UA into RDF to ensure a better understanding and effective processing of OPC UA data. This integration of OPC UA with ontologies and knowledge graphs opens up great potential for automation technology.

While several approaches and works have explored combining OPC UA information models with Semantic Web Technologies, the predominant focus has been on converting entire OPC UA servers into RDF. Notably, there has been no concerted effort to specifically transform OPC UA Skill information into RDF. The challenge arises due to the specific structures and semantics of the OPC UA Skill concept. To address this, this thesis focuses on meeting the specific requirements associated with the OPC UA Skills. This involves a comprehensive mapping of the way the skills are organized, encompassing their hierarchical structures and properties, as well as their relationships into RDF. The objective is to establish a formal mapping between OPC UA and RDF that not only includes the data structures but also takes into account the functionalities and properties inherent in OPC UA Skills. This semantic representation facilitates efficient analysis and enables improved information retrieval.

3. Transformation concept

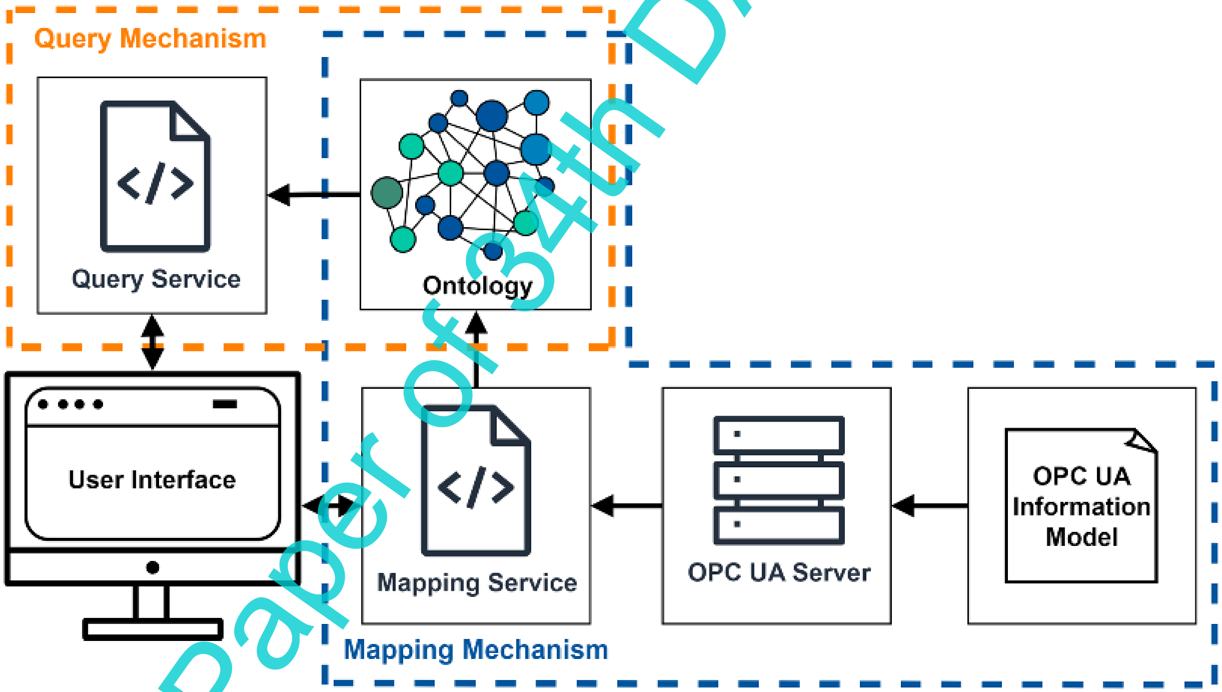


Figure 1. Transformation service architecture

The proposed concept for transforming OPC UA information models into ontologies and its effective utilization introduces a system comprising two key mechanisms: the mapping mechanism and the query mechanism. These mechanisms are accessible through a user interface, offering a user-friendly and interactive approach. Within the user interface, the user can select the OPC UA server to be translated. The proposed concept allows the transformation of one or more OPC UA servers, each operating as a process participant. For this purpose, the corresponding information models must already be implemented and running on the OPC UA Servers, with the address space of each server being accessible through the OPC UA client. The mapping service processes the user's selection and transforms the selected OPC UA information model into an ontology. This involves establishing a connection with the OPC UA server hosting the desired information model and executing the transformation process based on the defined mapping rules, as shown in

Table 1 (s. below). Once the mapping process is complete, and the selected OPC UA information model has been successfully transformed into an ontology, the query mechanism comes into play. This allows users to interact with the created ontology and perform relevant queries. Here, a query service is employed that effectively searches the ontology using user-defined queries and returns the corresponding results. These results, obtained through the query mechanism, can be used for further analysis and applications. The overall system structure is shown in Figure 1.

3.2. Mapping Service

The Mapping Service enables the transformation of OPC UA information models into RDF models and ensures a consistent transfer of OPC UA skills into ontologies. By analyzing the structure and semantics of both models, equivalent concepts are identified, leading to the development of mapping rules. These rules govern the transformation process, ensuring accurate and meaningful representation of OPC UA data into RDF format.

OPC UA and RDF are two different technologies, each serving different purposes and applications. However, they have relevant similarities in the modelling methodology. The primary goal of OPC UA is to provide a standardized data exchange between different devices and systems in industrial environments. It provides information models to represent data in a structured way. On the other hand, RDF is designed for data representation and exchange on the Web. It aims to visualize the data in a graph-based, machine-readable format. Although OPC UA and RDF are used in different areas, they have in common that they deal with structured data representations and exchange based on a graph-based structure composed of nodes and their connections. In both models, the connection between a *Source Node (Subject)* and a *Target Node (Object)* is established by a *Reference* (*Predicate*), as depicted in Figure 2. However, the specific XML format used by OPC UA for serialization is not directly compatible with RDF, making it difficult to directly map OPC UA information models to RDF [17]. Therefore, mapping rules must be developed to map the structure and semantics of OPC UA to RDF. The main mapping rule entails mapping OPC UA nodes as RDF resources and OPC UA references as RDF properties.

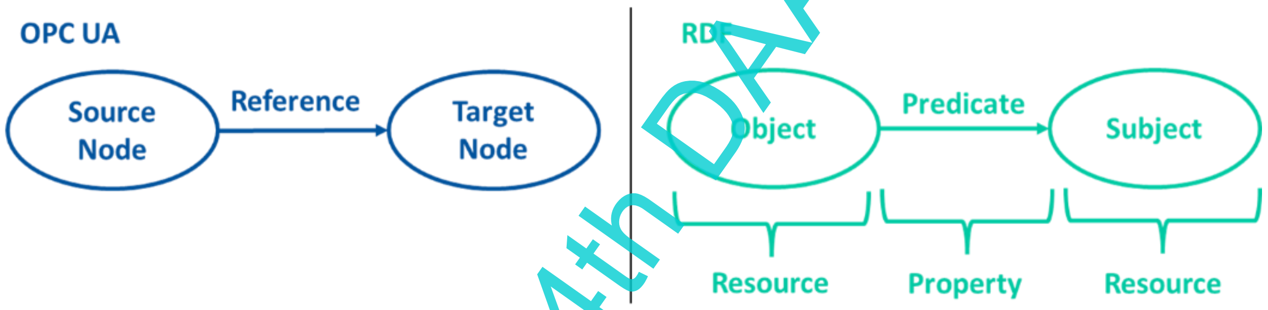


Figure 2. *Target Node-Source Node* in OPC UA, *Object-Subject* in RDF

The exact transformation of OPC UA information models into RDF models depends on the specific requirements of the system. It should be noted that OPC UA and RDF use different semantics to define concepts. In the context of this work the goal is to map OPC UA Skills into ontologies while preserving the semantics and meaning of the skill data. However, the complexity of the OPC UA information models is a challenge. The complexity increases with the number of classes, instances of those classes and their relationships and attributes. The OPC UA Base Information Model is already extensive and contains numerous classes to create instances for system modeling [19]. Yet not all classes are used in every context. Therefore, the concept of mapping the relevant classes is developed. Relevant classes are those for which at least one instance has been created in the OPC UA information model. By adopting this approach, only essential information is transformed from OPC UA to the ontology and the complexity of the resulting graph is minimized. This facilitates fast data processing and streamlines information retrieval.

Table 1 presents the mapping rules in more detail, specifying how different OPC UA nodes are mapped to specific types of RDF resources and properties. The focus is primarily on the OPC UA nodes of *Object*, *ObjectType*, *Variable*, *VariableType*, *Method* and *ReferenceType* node classes, as they play the central role in representing OPC UA Skills. In the mapping process, a crucial decision is made to represent OPC UA references as RDF properties, while all other OPC UA nodes are mapped to resources, with one exception. While inheritance exists in OPC UA, the explicit relationship between inherited features and their source classes is not explicitly defined, as shown in Figure 3. This lack of explicit relationship makes it difficult to identify and establish the connection between the inherited features and their respective source classes. To address this issue, an exception is made for the nodes belonging to the *Variable* node class which are instances of the *PropertyType* class. Because these nodes semantically represent properties, they are mapped as datatype properties in the RDF model. This mapping solves the problem by providing a clearer and more explicit representation of the relationship between nodes, their properties and their parent classes.

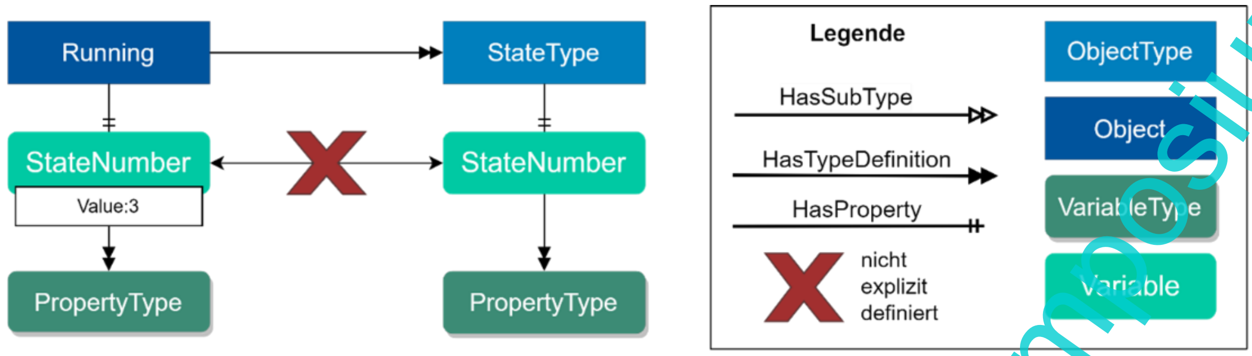


Figure 3. Example of an object node and its property

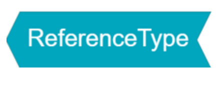
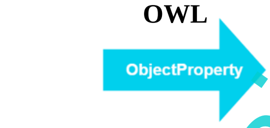
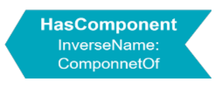
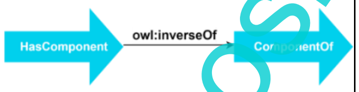
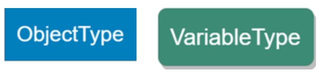
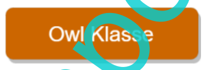
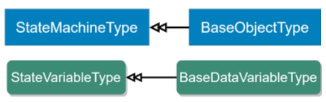
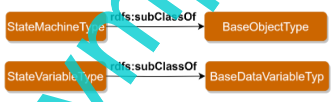
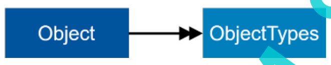
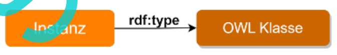
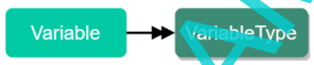
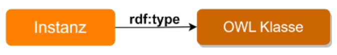
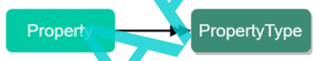
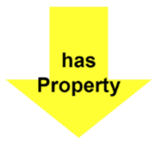
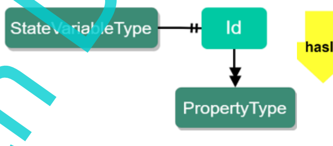
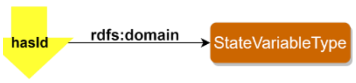
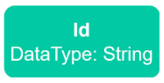
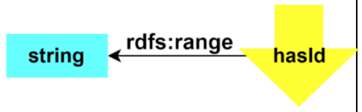
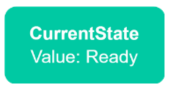
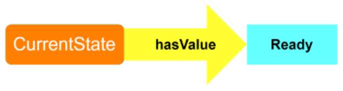
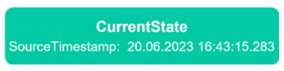
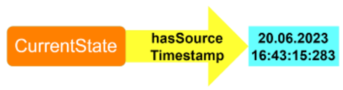
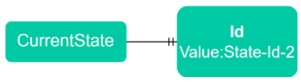
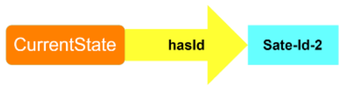
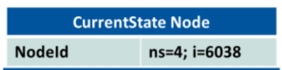
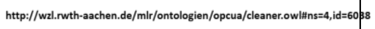
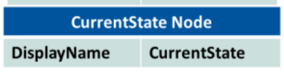
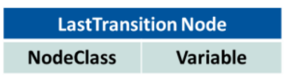
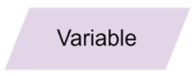
3.3. Query Service

The query service is designed to provide a user-friendly interface without requiring the user to have in-depth knowledge of the ontology, OPC UA server or SPARQL. Users can select a skill from the OPC UA server, and the query service processes this selection and searches the ontology for the variables, objects and methods associated with the selected skill. The retrieved results are then presented to the user for further refinement of the selection. The query service dynamically adapts its search operations based on the user's selection and displays the relevant results.

2. Limitations

It is important to note that the proposed mapping approach focuses only on transforming OPC UA Skills into RDF. This selective approach is intended to ensure that the resulting knowledge graph contains only the essential data required for an efficient OPC UA Skill analysis, since the OPC UA information models are often complex and consist of numerous nodes and references. Consequently, it does not cover the entire OPC UA information model.

Furthermore, the viability of the proposed transformation concept depends on the OPC UA server being built according to the OPC UA Skill structure. This conformance check requires manual verification, as the mapping process itself does not include server conformance assessment. In cases where the development of the server is incomplete or where the structure of skills deviates from the expected standard, the implementation of the transformation concept will require adjustments to align with the actual server structure.

Rules	OPC UA	OWL
1. OPC UA nodes of the node class *ReferenceType* are defined as owl:ObjectProperty.		
2. The references that have an *InverseName* are linked to their inverse *ObjectProperty* via owl:inverseOf.		
3. Nodes of node classes *ObjectType* and *VariableType* are defined as owl:Class.		
4. Nodes of node classes *ObjectType* and *VariableType* are defined as subclasses of their parent nodes using rdfs:subClassOf, to which they are connected through the *SubTypeOf* reference.	 StateMachineType; BaseObjectType; StateVariableType; BaseDataVariableType	 StateMachineType; rdfs:subClassOf; BaseObjectType; StateVariableType; rdfs:subClassOf; BaseDataVariableTyp
5. Nodes belonging to the node class *Object* are defined as instances of the type node to which they are linked by the *HasTypeDefiniton* reference.		
6. Nodes belonging to the node class *Variable*, which are not instantiated from the *PropertyType* node, are defined as instances of the type node to which they are connected via *HasTypeDefinition* reference.		
7. Nodes belonging to the *Variable* node class, which are connected to the *PropertyType* node via the reference *HasTypeDefinition*, are mapped as owl:DatatypeProperty.		
8. The range of *DatatypeProperties* mapped from the nodes instantiated from *PropertyType* are set on the class they are associated with *HasProperty* reference.		
9. The range of DatatypeProperties mapped from *PropertyType* nodes is set to the corresponding XSD data format depending on their OPC UA Datatype.		
10. The value attributes of nodes belonging to *Variable* node class are mapped as owl:DatatypeProperty named hasValue.		
11. The SourceTimestamp attributes of nodes that are not typed by *PropertyType* are mapped as owl:DatatypeProperty named hasSourceTimestamp.		
12. The SourceTimestamp attributes of nodes typed by *PropertyType* are mapped as owl:DataTypeProperty named has[Name of Property]SourceTimestamp.		
13. Node ids are used to determine URIs of Resources.		 http://wzl.rwth-aachen.de/mlr/ontologien/opcua/cleaner.owl#ns=4,id=6038
14. The *DisplayName* attribute is mapped as rdfs:label.		CurrentState
15. Attributes, except *DisplayName*, *Value* and *SourceTimestamp*, are mapped as owl:AnnotationProperty.		

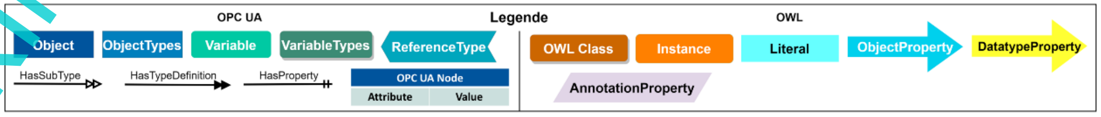


Table 1. Mapping Rules

3. Implementation

3.2. Use Case

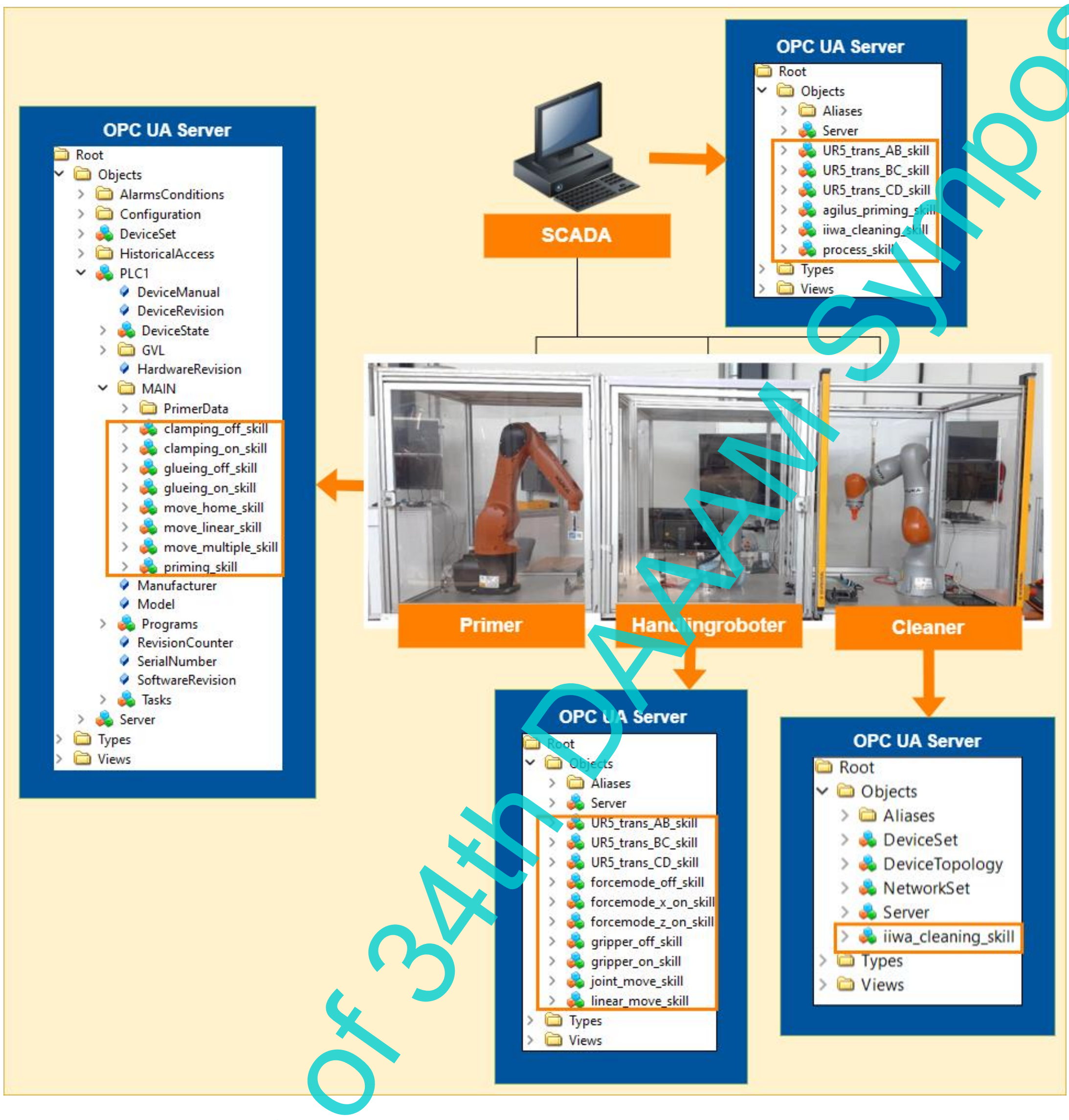


Figure 4. Process units and their OPC UA servers

The proposed concept is practically applied in a robotic sub-process of glass pane assembly. The sub-process consists of Cleaning, Handling, and Priming stations. In the first station, a KUKA iiwa robot is used to clean the working trajectory of a car window. In the Priming station a KUKA Agilus (KR 6) is used to apply glue to the cleaned shape of the glass. Each station is equipped with workpiece carriers for securely holding the glass pane and proximity sensors for precise positioning. The handling robot (UR 5) facilitates the transfer of the panes between the stations. Each station is assigned different skills to perform its specific tasks. Additionally, there is a SCADA (Supervisory Control and Data Acquisition) system that includes the compilation of the skills of all the individual units that are involved in the assembly process. Figure 4 illustrates an overview of the different systems and their corresponding OPC UA servers, showing the skills of each robot at its respective station.

Cleaner, Handling Robot and SCADA systems share the same OPC UA server structure, while there is a slight deviation in the Priming unit due to the limited import functionality of the individual information models in the TwinCAT software. Therefore, the implementation of the mapping algorithm for the standardized OPC UA Skills structure is covered first. This is followed by an explanation of the adjustments required to adapt the mapping algorithm to Primer.

3.3. Implementation of Mapping Service for Cleaner, Handler and SCADA

The OPC UA information models of the process units are embedded in their respective OPC UA servers. To access these information models, the connection to the OPC UA servers must be established first. Since each OPC UA process unit has its own dedicated OPC UA server, the mapping service is implemented in a separate document for each unit. The following section presents the implementation of the mapping service for a single server. The same implementation is used for all other servers, only the server URL and the name of the ontology document need to be adapted.

The service uses the opcua-asyncio Python library to connect to the OPC UA server, allowing seamless navigation and access to the nodes, their properties and relationships. Initially, the OPC UA references are mapped to RDF, starting from the *References* node which serves as the primary reference node from which all other references inherit. Its *NodeId* (ns:0; i=31) is predefined within the mapping service and serves as a consistent starting point for mapping all OPC UA references. For this use case, the *NodeId* is constant across all servers and is unaffected by server restarts.

To map the primary reference node as an object property, the function create_object_property_for_reference() is implemented. This function creates an object property for the primary reference node and translates its attributes into OWL AnnotationProperties. Afterward, the get_reference_children() function is employed recursively to find all child references of the current node and map them as object properties in OWL. These child references are linked to their parent using rdfs:SubPropertyOf. This mapping approach captures the hierarchical structure of OPC UA references, by placing the subordinate references in a hierarchical relationship to the parent references. In addition, the attributes of the references such as *DisplayName*, *NodeId*, *NodeClass* are passed as AnnotationProperty in OWL. The get_reference_children() function also takes into account the *InverseName* of a reference and creates a corresponding object property for the inverse reference. This allows the representation of inverse relationships of the OPC UA references within the OWL ontology.

After successfully transforming all references, the mapping service proceeds to identify the skill nodes that are located under the *Objects* node with the *NodeId* (ns=0; id=85). To do this, the get_type_def() function is used to search for the node connected to the *Objects* node via the *HasTypeDefinition* reference. The node found represents the type class of the *Objects* node. Next, the get_super_class() function is used to recursively identify parent classes. These classes are then mapped as OWL classes using the create_class() function and linked to their subclasses using rdfs:subClassOf. Once the type class (e.g., *FolderType*) and parent classes (e.g., *BaseObjectType*) are mapped, the instances, such as *Objects*, of the corresponding class can be created using the create_instance() function. The create_instance() function then calls the get_referenced_nodes() function, to further capture the child nodes that are connected to the instance node by a specified reference. In this use case, the *Objects* node is connected to child nodes through the *Organizes* reference. Therefore, the Organizes reference is passed to the get_referenced_nodes() function to identify and map the child skill nodes to the ontology. To ensure that only the relevant nodes are included in the ontology and to omit other irrelevant nodes such as *Aliases*, *DeviceSet*, *DeviceTopology*, etc., only the nodes instantiated by the *StateMachineType* class are transformed into the ontology, as they represent the skill nodes. Then, the *HasComponent* reference is passed to the get_referenced_nodes() function, as skills are linked to their child variables and objects through this reference. These objects and variables of a skill can therefore be called as components. To map these components in OWL, the functions create_instance(), get_superclass() and get_referenced_node() are called. Through this recursive process, all relevant nodes, and their relationships for the OPC UA skills are effectively mapped to the ontology.

For value and timestamp attributes of a node, data type properties are created. The get_data_type_property() function finds nodes linked by the *HasProperty* reference and creates data type properties for them. It establishes the data tzpe property's domain based on the inherited type class. To set the range of each data type property, it maps the OPC UA data type of the property node's value to the appropriate XSD (XML Schema Definition) data format. Finally, RDF triples are created by connecting the instance with its value using newly created data type propert. The resulting ontology is then stored in XML format.

Figure 5 provides an illustrative example of mapping OPC UA type classes and instances to RDF format.

3.4. Implementation of Mapping Service for Primer

The OPC UA information model of the Priming unit differs from other units in terms of node and relationship structure. The implementation must handle these differences. In the Priming unit, the skill nodes are located under the *MAIN* node and connected by the *HasComponent* reference, unlike other units where they are listed under the *Objects* node and connected by the *Organizes* reference. The variables and objects subordinate to the skill nodes remain unchanged. The Priming unit has an additional object node named *skill_function* that is subordinate to the *priming_skill* node. This skill node contains *Parameters* node linked to the other variables through the *HasProperty* reference. The previously used exception to transform only those nodes that are linked to their parents via the *Organizes* reference and are of type *StateMachineType* in the ontology, can therefore no longer be used for the Primer. This exception has to be removed from the implementation. The variables of the skills have no other properties, while the methods, such as *Start*, have properties like *InputArguments* and *OutputArguments*. Therefore, a data type property is created for each method property and its domain is set to the method class. In addition, custom type classes derived from the BaseObjectType class are defined for the skills, states and state transitions, but these custom type classes are linked to the nodes, such as *PLC1*, which is instantiated from *FolderType*, through the *Organizes* reference. To maintain the hierarchical structure and avoid confusion, these folder nodes, *PLC1* and *BeckhoffControlTypes*, are excluded from the implementation, because they do not represent type classes and do not share inheritance relationships with their subordinates through *HasSubType* reference. The type classes for the skills (e.g., *clamping_skill*) are subclasses of *BaseObjectType*. This does not require any changes to the existing implementation, as it already supports the creation of OWL classes and subclasses.

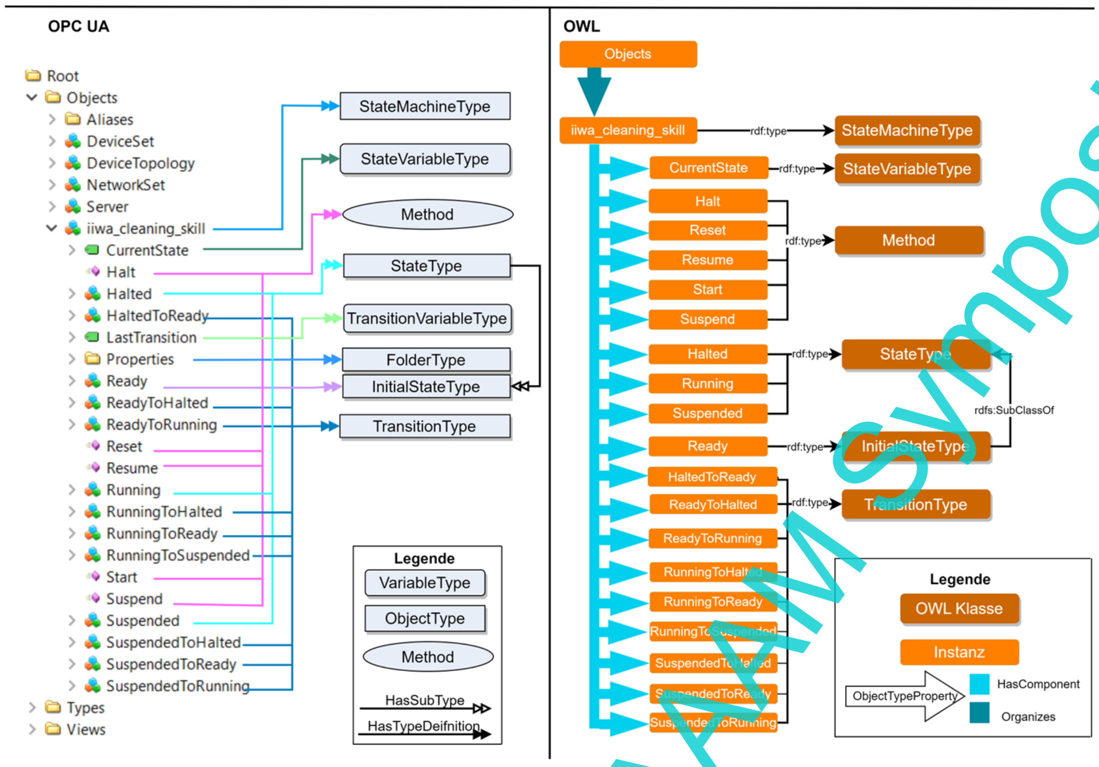


Figure 5. Example of Mapping OPC UA Skills in OWL

3.5. Implementation of Query Service

The query service is implemented using the Python Dash library for an interactive and user-friendly interface with four buttons: *Start Transformation* to select an OPC UA server for ontology transformation and four additional *View Ontology* buttons to select the ontology post-transformation. Clicking the *Start Transformation* button executes the code to transform the chosen OPC UA server into the ontology. Once the transformation process is done, the user can select an ontology using corresponding *View Ontology* button. The interface directs users to a new page where the visualization of the associated ontologies is generated using the dash-cytoscape library. Additionally, two dropdown menus are integrated to allow the user to execute specific SPARQL queries and retrieve the desired values from the ontology.

4. Summary and outlook

In the context of this work, a concept for translating OPC UA information models into ontologies with a focus on OPC UA Skills was developed. This concept was implemented and validated using an industrial robot-based use case of glass pane completion. It has been shown that OPC UA information models containing the OPC UA Skills can be transformed into ontologies and that the desired information about their execution can be easily retrieved from the ontology by SPARQL queries. However, the developed concept requires a full compliance of the models to be translated with the OPC UA skill concept. The skill information models that are not created according to the OPC UA Skill standard should be treated differently. As in the case of the Priming unit application example, an analysis of the structure of the OPC UA information model is necessary to adapt the implementation of the concept to the different structure of the OPC UA information model, so that all required nodes are transferred to the ontology. Especially in the cases where user-defined type nodes are present in the OPC UA servers or different references between the nodes are used, additional mappings and contexts have to be considered.

In future work, it could be advantageous to extend the translation system with a checking mechanism that can check whether the OPC UA server to be translated is built or complete according to the OPC UA standards. In the case that the OPC UA server is not complete, then the system should give suggestions to the user to complete the server structure or create the missing nodes automatically. Furthermore, the algorithm is to be used as part of the assistance system to be developed for knowledge-based control process reconfiguration, in the context of the scientific project described above.

5. Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany ´s Excellence Strategy – EXC-2023 Internet of Production – 390621612.

Furthermore, a vivid exchange with Chair for Production Metrology and Quality Management at the WZL of the RWTH Aachen University (WZL-IMA), especially with Michael Zeng, is highly appreciated.

6. References

[1] B. Bajic, I. Cosic, B. Katalinic, S. Moraca, M. Lazarevic und A. Rikalovic (2019), Edge Computing vs. Cloud Computing: Challenges and Opportunities in Industry 4.0, 30TH DAAAM International Symposium on Intelligent Manufacturing and Automation, doi:10.2507/30th.daaam.proceedings.120

[2] M. Obdenbusch (2018), Referenzarchitektur für cloudbasiertes Condition Monitoring am Beispiel von Verpackungsmaschinen, Apprimus Wissenschaftsverlag, ISBN:978-3-86359-573-9, Aachen

[3] I. Reithner, M. Papa, B. Lueger, M. Ćato, S. Hollerer und R. Seemann (2020), Development and Implementation of a Secure Production Nretwork, 31ST DAAAM International Symposium on Intelligent Manufacturing and Automation, Vienna, Austria, doi: 10.2507/31st.daaam.proceedings.102

[4] S. Profanter, A. Perzylo, M. Rickert und A. Knoll (2021), A Generic Plug & Produce System Composed of Semantic OPC UA Skills, IEEE Open Journal of the Industrial Electronics Society, p. 128–141, doi:10.1109/OJIES.2021.3055461

[5] plattform-i40: Information Model for Capabilities, Skills & Services, Available: https://www.plattformi40.de/IP/Redaktion/EN/Downloads/Publikation/CapabilitiesSkillsServices.html.

[6] K. Dorofeev und A. Zoitl (2018), Skill-based Engineering Approach using OPC UA Programs, IEEE 16th International Conference on Industrial Informatics, doi: 10.1109/INDIN.2018.8471978.

[7] R. Schiekofer und M. Weyrich (2019), Querying OPC UA information models with SPARQL, 24th IEEE International Conference on Emerging Technologies and Factory Automation, p. 208–215, doi: 10.1109/ETFA.2019.8868246.

[8] A.Dengel (2012), Semantische Technologien: Gundlagen - Konzepte - Anwendungen, Spektrum Akademischer Verlag, ISBN: 978-3-8274-2663-5, Heidelberg.

[9] M. Merdan, A. Zoitl, G. Koppensteiner und F. Demmelmayr (2010), Semantische Technologien – Stand der Technik, Elektrotech. Inftech., p. 291–299, doi: 10.1007/s00502-010-0777-3.

[10] G. Saake, K. Uwe, A. Heuer (2018), Datenbanken-Konzepte und Sprachen, mitp Verlag, ISBN: 9783958457782

[11] A. Mueller, W. Jesse, S. Storms, W. Herfs und C. Brecher (2023), Ontology-based assistance system for control process reconfiguration of Robot-Based Applications, 4th Conference on Production Systems and Logics, Hannover, doi: 10.15488/13444.

[12] C. Brecher, M. Buchsbaum, A. Müller, K. Schilling, M. Obdenbusch, S. Staudacher and M. Chaikh Albasatineh (2021), Gaining IIoT insights by leveraging ontology-based modelling of raw data and Digital Shadows, 4th IEEE International Conference on Industrial Cyber-Physical Systems, doi: 10.1109/ICPS49255.2021.9468116

[13] J. Lipp und K. Schilling (2020), The semantic web in the Internet of Production. A strategic approach with use case examples, the Fourteenth International Conference on Advances in Semantic Processing, p. 68-72, ISBN: 978-1-61208-813-6

[14] S. Pieske, W. Herfs, M. Zenke, S. Storms und C. Brecher (2022), Semantic modeling of a cyber-physical biological production platform, 27th IEEE International Conference on Emerging Technologies and Factory Automation , Stuttgart, doi: 10.1109/ETFA52439.2022.9921616

[15] M. Graube, L. Urbas und J. Hladik (2016), Integrating industrial middleware in Linked Data collaboration networks, 21st International Conference on Emerging Technologies and Factory Automation , pp. 1-8, doi: 10.1109/ETFA.2016.7733710

[16] A. Bunte, O. Niggemann und B. Stein (2018), Integrating OWL Ontologies for Smart Services into AutumationML and OPC UA, 23rd International Conference on Emerging Technologies and Factory Automation, pp. 1383-1390, doi: 10.1109/ETFA.2018.8502593

[17] A. Weiss und S. Ihlenfeldt (2022) , Integration of OPC UA Information Models into Enterprise Knowledge Graphs, Journal of Machine Engineering, doi: 10.36897/jme/150008

[18] G. Steindl und W. Kastner (2021), Transforming OPC UA Information Models into Domain-Specific Ontologies, *4th IEEE International Conference on Industrial Cyber-Physical Systems*, , pp. 191-196, doi: 10.1109/ICPS49255.2021.9468254

[19] M. Graube, S. Hensel, C. Iatrou und L. Urbas (2017), Information models in OPC UA and their advantages and disadvantages, *22nd IEEE International Conference on Emerging Technologies and Factory Automation, doi:* 10.1109/ETFA.2017.8247691