Towards Self-Configuring Plug & Produce Robot Systems Based on Ontologies

Christian Eymüller, Julian Hanke, Alexander Poeppel, and Wolfgang Reif
Institute for Software and Systems Engineering
University of Augsburg
Augsburg, Germany
{eymueller, hanke, poeppel, reif}@isse.de

***Abstract*—The industry is undergoing a disruptive shift from mass production to individually manufactured one-offs. To achieve this transformation, the production facilities must also be flexible and able to configure themselves according to requirements. Often it is not enough to reconfigure the software, but the hardware must also be adapted accordingly. Such customizable hardware and software systems are called Plug & Produce systems. For such Plug & Produce robot systems, we have developed an automatic configuration generation for hardware configurations based on capability/skill descriptions of individual hardware components. The configuration creation is done with the help of an ontology, in which information about modular hardware components, their capabilities, and the interrelationships of their interfaces (geometric, electrical, and data) is stored.**

***Index Terms*—Plug & Produce, Ontologies, Self-Configuring Robots, Skills, CPPS**

I. Introduction

In the next few years, the goal of industry and research is to develop a variety of autonomous robots that can cope with a wide range of tasks autonomously [1]. In addition to changing tasks, the environment or working conditions may also change. To achieve this goal, the robots need to know about the tasks, environment, and physical structure. Concerning Industry 4.0 or smart factories, there is always mention of Cyber-Physical Production Systems (CPPS), a specific kind of Cyber-Physical System (CPS) in the domain of smart manufacturing [2]. A CPPS connects a production system's physical and digital dimensions into a uniform environment and enriches hard- and software components and physical products with knowledge and data. This knowledge and data range from simple descriptions of objects to elaborate descriptions of processes or complex interrelationships of objects. This reusable machine-readable knowledge is decisively responsible for the fulfillment of executing tasks autonomously.

In order to perform many different tasks, the production cell must be designed as flexibly as possible, and the system must be able to reconfigure itself. In this context, reconfiguration means that the system needs to reconfigure itself physically and logically to execute a specific task. Such systems are known as industrial Plug-and-Produce systems [3], [4]. The logical reconfiguration in such a system is often realized through analyzing industrial standard operating procedures and decomposing these procedures into individual reusable skills [5]. These reusable skills offer the possibility to model a flexible manufacturing process that can be adapted by changing the parameters or the composition of its skills [6]. Physical reconfiguration requires knowledge of the interrelationships among components and between components and their skills. For example, if the task is to grip a component, in addition to the logical composition of the gripping, a tool (e.g., a gripper) must also be attached to the robot to be able to execute the task physically. Semantic web technologies, like RDF, OWL, or SPARQL, can be used to represent these relationships and make them machine-readable [7].

In order to create logical and physical reconfigurable industrial robot systems more easily in the future, we introduce a concept of using machine-readable descriptions to automatically generate hardware configurations for robot systems depending on a given task. The following points must be taken into account:

- A hardware configuration must be physically feasible. Electronic, data, and fastening interfaces of the tool and robot must match or be adapted accordingly.
- Skills that are also composed of skills of several different components must be considered. For example, the skill *move component* requires a gripper that has the ability to *grip* and a robot that can *move*.
- Additional constraints must be taken into account, e.g., a robot has only a certain payload.

The paper is structured as follows to illustrate the concept. Section II summarises the current state of the art in semantic descriptions of robotic and production systems and provides an overview of autonomously reconfigurable robot systems. Our approach is explained in detail in Section III. Section IV shows the implementation of the ontology and how is used for configuration genesis. Section V presents a case study to evaluate the approach. Conclusions are drawn in Section VI.

II. State of the Art

Numerous approaches use semantics for solving geometric constraints to facilitate component design or automate robot assembly. Ambler and Popplestone [8] were one of the first that used CAD data to generate assembly programs for robots. On the basis of this, continuous further development was carried out, and additional relationships were added between the component to be assembled, the resources (e.g. robots, production machines) that are needed, and the skills of the resources [9], [10]. These approaches usually have a strict 1-to-1 relationship between a resource and a skill. Nevertheless, there are also approaches that assign combinations of several components to one skill [11]. For example, a robot with different end effectors can execute different skills. However, there are also strong restrictions here, since in the example of Profanter et al. only tools with a corresponding tool change adapter can be used, and also the interfaces (data, power supply) are identical for all tools. Romiti et al. presented a reconfigurable cobot with different joint modules and end effector modules [12]. Nevertheless, even in this approach, the individual modules have a uniform interface for mounting, data, and power supply, making it easier to create configurations. There is also good preliminary work from Siltala et al., who has developed an ontology for the formal description of hardware interfaces [13]. In the ontology, categories are defined for mechanical, electrical, service and communication interfaces. However, it is not shown how these can be configured or parameterized. In addition, the ontology has a very abstract description of the interfaces and uses mainly standardized interface descriptions, which is difficult for non-standardized interfaces. By integrating another ontology, the relationship between capabilities and the components in the overall architecture was also taken into account [14].

For our approach, we have also focused on ontologies, as these are easily extendable at runtime and can easily scale rules and constraints for complex systems with many components. Besides the classical semantic web technologies RDF and OWL, for the matchmaking of the interfaces and the rules on how they can be connected to each other the Semantic Web Rule Language (SWRL) is used [15]. SWRL offers the possibility to integrate complex rules beyond OWL rules into the ontology. Since additional constraints for configurations are to be considered, and these are usually not static, they are integrated into the configuration queries. The semantic web query language SPARQL (SPARQL Protocol And RDF Query Language) is used for these configuration queries [16]. To the best of our knowledge, our approach is the first that calculates hardware configurations for robot systems which take different geometric, data, and power supply interfaces into account, combines it with skill descriptions of the individual components, and takes additional constraints into account.

III. Concept

A. Base Architecture: RealCaPP

The base architecture is outlined in Figure 1. The *Real-time Capable Plug & Produce Architecture (RealCaPP)* is a uniform architecture with uniform interfaces so that industrial components can be easily integrated into new or existing systems or so that plants can adapt their skills [17]. The architecture follows concepts of the asset administration shell (AAS) and uses a common middleware to connect all components. The middleware has been divided into a non-realtime communication channel (OPC UA) for the configuration of

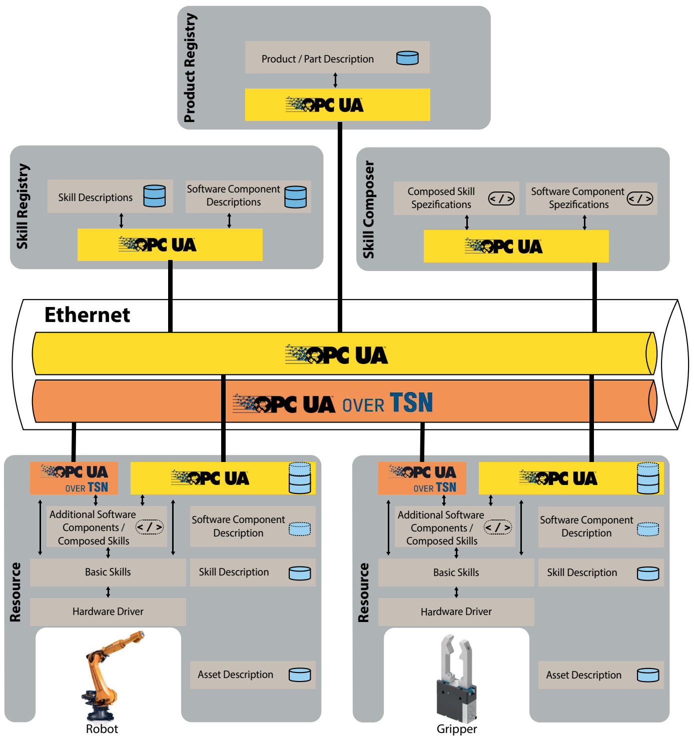

Fig. 1. System architecure for a real-time capable plug & produce environment [17]

assets as well as the exchange of descriptions and a real-time communication channel (OPC UA over TSN) to transmit control signals. Each resource (e.g., a robot or a gripper) has an AAS, which consists of a resource hardware driver, basic executable skills of the resource (e.g., the skill grip of a gripper or the skill move to position of a robot) and the ability to load additional software components, for example, additional or composed skills of one or more resources (e.g., pick and place as a combination of the resources robot and gripper). In addition to the executable components of a resource, there are descriptions of the resource itself and information on the respective software components and skills. Globally in the system, this information is summarized in registries to make it easier to find required skills or information in the system. A product register is available to integrate product or part descriptions into the system in order to have the possibility to adapt capabilities to parts or products. In the further course, the focus is mainly on the individual descriptions and how they can be used together.

B. RealCaPP Ontology

Figure 2 shows the class hierarchy of the RealCaPP ontology. The main class is *Resource* which represents the actual resource of the AAS, for example, a robot. All the information stored in the ontology for a resource is linked to this class. The skill descriptions of the basic skills are linked to the resource via the property *hasBasicSkill*. Each resource also has various interfaces that belong to a resource. This is represented by the *hasInterface* property. Interfaces can currently be divided into three different classes, but additional classes can easily be added. The first class is the geometric interface. This interface describes possible connections for the mechanical fastening of other resources. These interfaces vary from automatic tool change systems to screw connections between resources. Now that geometrical connections can be mapped, many resources still need connectors for power and data supply. The class *ElectricalInterface* represents power connections. The power supply also differentiates between alternating and direct current, and different voltages. The last type of interface is the *DataInterface*. In an industrial plant, there are countless data interfaces from analog to digital transmissions, different topologies, compatibilities between data interfaces, and so on. Therefore the description of data transmission is one of the biggest challenges.

In order to establish a connection between several resources, it is necessary that all interfaces can be connected accordingly. So it is not enough to say that a gripper is physically attached to a robot and can perform its gripping task without power or data supply. Therefore, the resource must also map the interfaces required to execute the corresponding skills. Connections between corresponding interfaces are established via the *connectable* property. In order to define which interfaces can be connected to each other, expert knowledge must be used. With these descriptions, the first necessary requirement can now be checked off: *the hardware configuration must be physically feasible*. This is the case if two or more resources are geometrically, electrically, and data connectable.

The second requirement that *skills that are also composed of skills of several different components must be considered* can be implemented by combining the *connectable* property, the information of the skill composer, and the descriptions of the basic resource skills. In the skill composer, expert knowledge is used to define rules for how complex skills are composed of other skills. For example, the skill pick and place is composed of the basic skills move to position and grip. If the system has to execute a task pick and place, it first needs to reconfigure to a system that is composed of one or more resources, which has the skills grip and move. Subsequently, it must be checked whether all resources of this system can be connected accordingly (geometrically + electrically + data connectable). If this is the case, the configuration can be used to execute the corresponding task. Otherwise, another configuration must be selected, or the user must be informed accordingly that no corresponding configuration was found and what requirements are missing for executing the task. The configurations are not limited to one or two resources - resource chains can also be formed. For example, placing an adapter plate between the robot and the gripper may be necessary because the gripper cannot be attached directly to the robot (not geometrically connectable). Similar constellations are also possible with data connections or power connections through additional bus couplers or voltage converters.

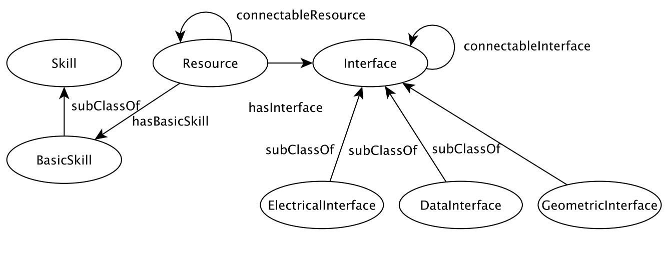

Fig. 2. Excerpt of the class hierarchy of the RealCaPP Ontology

Due to the chaining of multiple resources, it is essential that additional constraints can be checked. If there are physical restrictions, for example, the limited payload of the robot, this must be taken into account accordingly when creating the configuration. The descriptions of the individual resources (e.g., the weight of a resource) allow general statements to be made about a configuration (e.g., the total weight of the configuration). This composite information can be used for corresponding constraints, which is the last requirement for the configurations (*Additional constraints must be taken into account*). Obviously, also other aggregation functions can be used for the constraints. For example, to limit the accuracy or response times of configurations, the maximum values of the individual devices must be considered.

IV. Implementation

After the concept of self-configuring Plug & Produce robot systems has been presented, an overview is given of how the ontology and querying are realized. In the first step, information is collected from the entire system (all resources) so that all information is available in a uniform data source. For this purpose, the individual OWL files are combined to form an overall OWL ontology. In order to re-use the definition of connectivity, a combination of OWL property chains and SWRL rules have been used to define the ObjectProperty *connectableResource*. The *connectableResource* property has been split into the sub-properties *dataConnectableResource*, *powerConnectableResource* and *geoConnectableResource*. Listing 1 shows an example for the SWRL rule to define *dataConnectableResource* and a rule to combine the sub-properties to the property *connectableResource*. For clarity, the rule has been simplified so that all connectable sub-properties must always apply.

```
...
hasDataInterface( ?re1 , ?di1 ) ^ hasDataInterface( ?re2 , ?di2 ) ^
      dataConnectableInterface( ?di1 , ?di2 ) −>
      dataConnectableResource( ?re1 , ?re2 )
dataConnectableResource( ?re1 , ?re2 ) ^
      powerConnectableResource( ?re1 , ?re2 ) ^
      geoConnectableResource( ?re1 , ?re2 ) −>
      connectableResource( ?re1 , ?re2 )
...
```

Listing 1: SWRL rules for *dataConnactable* and *connectable*

Through the applied rules, it is evident in the ontology that resources are connectable to each other. Based on this, SPARQL queries can now be used to find appropriate hardware configurations. For this purpose, SPARQL query blocks are combined into complex queries. First, the task is broken down into individual basic skills. For this purpose, the ontology specifies the basic skills that make up composed skills. A resource list for each skill can be generated via the connection of the basic skills to the resource. All of these lists are merged by the **UNION** command to one global list (see Listing 2). If there are already known restrictions on the choice of resources, these restrictions can be supplemented by additional filters.

```
#for each basic skill (basicSkillX)
{
  SELECT ?resource, ?skill
  WHERE {?resource realcapp:hasBasicSkill ?skill.
            FILTER(?skill = basicSkillX) .
            #additional filters (optional)
            FILTER(...) .}
}
UNION
{
  SELECT ?resource, ?skill ...
}
```

Listing 2: SPARQL query for resource list

Then the system tries to find a configuration that contains resources with all the required basic skills. For this purpose first all possible configurations are preselected. Listing 3 shows the simplified structure of the query resulting in suitable hardware configurations. In the SPARQL query shown, the assumption was made that components can always be connected in general (geometric + electrical + data). More complex combinations are possible if geometry, electrical, or data connectivity is considered individually.

```
SELECT (GROUP_CONCAT( ?mid; SEPARATOR=";") AS ?
     configuration) (COUNT(?mid) as ?numberOfDevices)
WHERE { ?startdevice (realcapp:connectableResource)+ ?mid.
          ...
          ?mid (realcapp:connectableResource)∗ ?enddevice.
          FILTER(?startdevice = <startpoint>).
          ...
}
GROUP BY ?startdevice ?enddevice
```

Listing 3: SPARQL query for suitable hardware configurations

Afterward, all configurations with resources that do not fulfill the required basic skills are sorted out. As there can be solutions with many adaptors due to connection chains, by default, configurations with as few elements as possible are preferred (lowest ?numberOfDevices). As shown below, also other constraints can be used for the selection of the correct configuration.

With the help of aggregations, grouping, and additional filtering in the SPARQL query, it is possible to consider additional constraints for the configurations. Because the required data for the constraints (e.g., weight data) are needed in the surrounding **SELECT** statement, the **SELECT** and **WHERE** statements must be adjusted accordingly. Therefore, in the current implementation, it is necessary to adapt the respective SPARQL queries to specify additional constraints.

V. Case Study

The concept was evaluated using a case study. Therefore, the ontology has been filled with instances of resources ($>$ 50 different resources). The resources include industrial robots, automatic tool changing systems with different electrical and data interfaces, adapter plates, various tools (e.g., gripper, screwer, etc.) with different electrical and data interfaces, and many more. Instances of basic skills (e.g., move to position, screw, grip) and resource interfaces were also created for the various resources. Also, knowledge of combinations of basic skills to composed skills was added to the ontology. In addition, expert knowledge was used to define which interface instances can be connected with each other accordingly (e.g., screwbit is connectable to screwer). The definition of the expert knowledge is done by SWRL rules by adding these to the ontology.

Based on this database, several queries for different configurations with varying constraints were performed. Afterward, the corresponding configurations were examined for correctness. The evaluation assessed whether all selected constraints were fulfilled and whether these configurations could actually be realized.

Figure 3 shows an exemplary section from the instantiated ontology for a configuration that has the composed skill screw in screw (= screw + move to position). The SWRL rules automatically add edges to the ontology. For example, from the expert knowledge that the interfaces ScrewerIfaceM and ScrewerIfaceF can be connected

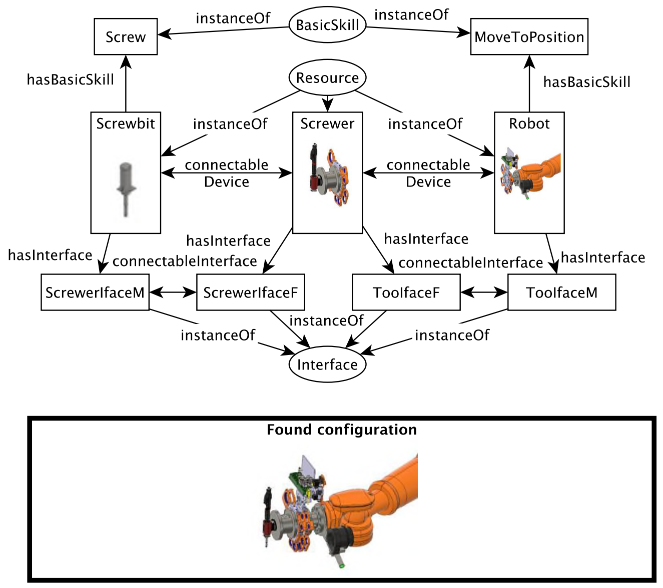

Fig. 3. Example of a configuration with the corresponding ontology data

Table I. Resource list output of Listing 2 for the the basic skills Screw and MoveToPosition

?resource	?skill
Screwbit	Screw
Robot	MoveToPosition
...	

to each other (connectableInterface), it is deduced that the resources Screwbit and Screwer can be connected to each other (connectableDevice). First the SPARQL query in Listing 2 was executed with the two basic skills Screw and MoveToPosition which outputs table I. By using the SPARQL query Listing 3 it is possible to search and find resource chains on the extended graph. In the example shown, the result is the following resource chain (Robot;Screwer;Screwbit) with a length of 3.

In the evaluation, the restriction that components are generally connectable was waived (simplification for the example). The connectability matchmaking was thus performed for all connection types (geometric, electric, data). It has been shown that as soon as there were resources with basic skills that made a configuration possible, it was also found. Sometimes even several correct configurations were found. With appropriate ASK commands in SPARQL, it is also possible to find out in advance if there is a corresponding configuration. Due to the knowledge preparation by the SWRL rules, the queries for the resource chains can be processed in less than one second (average query response time 0.87 s) with knowledge graphs with approximately 50 resources.

VI. Conclusion and Future Work

This paper introduced an approach for self-configuring Plug and Produce robot systems. For this purpose, a corresponding ontology was created, and a method to obtain possible hardware configurations capable of executing a given task was created by querying the ontology. Based on a case study, it was shown that the correct configuration could be found for a given task, or at least the conclusion can be made that there is no configuration with the given hardware resources. Moreover, extended constraints can be added, which should apply to the hardware configurations. With a knowledge base of about 50 resources, hardware configurations could be found in under a second.

The approach shows that it is possible to find hardware configurations for robots for a given task automatically. However, there are still limitations. We currently assume a chain of components as configuration. In industrial settings, it is also possible to have tree structures or other topoloties. e.g., two end effectors on one robot. Especially for the data connectivity, multiple topologies have to be considered. For example, EtherCat connections do not support star topologies, which must be considered in the interfaces' description. In addition, it will be researched whether a robot can automatically assemble the hardware configurations found. Concepts are needed to define which connections can be made automatically (e.g., an automatic tool changer) or which can only be realized by a human (e.g., a screwed connection of the end effector). For large data sets with numerous resources (several thousand resources), we try to perform presorting to improve our approach's scalability.

References

[1] J. Michniewicz and G. Reinhart, "Cyber-physical robotics–automated analysis, programming and configuration of robot cells based on cyber-physical-systems," *Procedia Technology*, vol. 15, pp. 566–575, 2014.

[2] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda, "Cyber-physical systems in manufacturing," *CIRP Annals*, vol. 65, no. 2, pp. 621–641, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0007850616301974

[3] M. Onori, N. Lohse, J. Barata, and C. Hanisch, "The IDEAS project: plug & produce at shop-floor level," *Assembly automation*, 2012.

[4] J. Pfrommer, D. Stogl, K. Aleksandrov, S. E. Navarro, B. Hein, and J. Beyerer, "Plug & produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems," *at-Automatisierungstechnik*, vol. 63, no. 10, pp. 790–800, 2015.

[5] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Krüger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282–291, 2016.

[6] A. Björkelund, J. Malec, K. Nilsson, and P. Nugues, "Knowledge and skill representations for robotized production," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 8999 – 9004, 2011.

[7] T. Moser and S. Biffl, "Semantic Integration of Software and Systems Engineering Environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 1, pp. 38–50, 2012.

[8] A. Ambler and R. Popplestone, "Inferring the positions of bodies from specified spatial relationships," *Artificial Intelligence*, vol. 6, no. 2, pp. 157–174, 1975.

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

[10] Y. Pane, M. H. Arbo, E. Aertbeliën, and W. Decré, "A System Architecture for CAD-Based Robotic Assembly With Sensor-Based Skills," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1237–1249, 2020.

[11] S. Profanter, A. Perzylo, M. Rickert, and A. Knoll, "A Generic Plug & Produce System Composed of Semantic OPC UA Skills," *IEEE Open Journal of the Industrial Electronics Society*, vol. 2, pp. 128–141, 2021.

[12] E. Romiti, J. Malzahn, N. Kashiri, F. Iacobelli, M. Ruzzon, A. Laurenzi, E. M. Hoffman, L. Muratore, A. Margan, L. Baccelliere, S. Cordasco, and N. Tsagarakis, "Toward a Plug-and-Work Reconfigurable Cobot," *IEEE/ASME Transactions on Mechatronics*, pp. 1–13, 2021.

[13] N. Siltala, E. Järvenpää, and M. Lanz, "Creating resource combinations based on formally described hardware interfaces," in *Precision Assembly in the Digital Age*, S. Ratchev, Ed. Cham: Springer International Publishing, 2019, pp. 29–39.

[14] ——, "A method to evaluate interface compatibility during production system design and reconfiguration," *Procedia CIRP*, vol. 81, pp. 282–287, 2019, 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.

[15] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosofand, and M. Dean, "SWRL: A semantic web rule language combining OWL and RuleML," W3C Member Submission, W3C, 2004, last access on Nov 2022. [Online]. Available: http://www.w3.org/Submission/SWRL/

[16] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," W3C Recommendation, 2008, last access on Nov 2022. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/

[17] C. Eymüller, J. Hanke, A. Hoffmann, A. Poeppel, C. Wanninger, and W. Reif, "Towards a Real-Time Capable Plug & Produce Environment for Adaptable Factories," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2021, pp. 1–4.