Plug&Produce Integration of Components into OPC UA based data-space

Santosh Kumar Panda[1], Tizian Schröder[2], Lukasz Wisniewski[1], and Christian Diedrich[2]

[1]inIT - Institute Industrial IT, OWL University of Applied Science, 32657 Lemgo, Germany
{firstname.lastname}@hs-owl.de
[2]Institute for Automation Technology, Otto von Guericke University, 39106 Magdeburg, Germany
{firstname.lastname}@ovgu.de

***Abstract*—Current manufacturing and production systems are becoming more flexible and adaptable. Such systems demand rapid changeover and short configuration time of machines. With the adaptation of IT systems in the automation industry, machines and devices will have self-descriptive semantic device data description to represent their data elements and services. This paper proposes an approach to integrate new machines and devices into an existing production system without any manual intervention. The proposed concept provides a Plug & Produce system architecture and evaluates its capabilities based on OPC UA. The concept also describes an easy and fast integration of the digital representation of new field devices into an existing production system.**

***Index Terms*—Industry 4.0, Asset Administration Shell, Plug & Produce, OPC UA, EDDL**

I. Introduction

The number of machines and their interconnectivity in the modern industrial environment is increasing with rising levels of digitalization and automation. The fourth industrial revolution also known as Industry 4.0 aims to further increase the interconnectivity and intelligence of industrial component to provide smart and self-organizing machines and optimizing production processes [1]. Within such component, machines and devices also known as assets[1] are capable of autonomously exchanging information and provide a vast amount of data and services [1], [2]. Industry 4.0 also introduces flexible and adaptable manufacturing concept to cope with the increasing demands for customization [3]. In such flexible manufacturing system, new assets need to be integrated optimally without any manual effort [4].

To support the fast integration of newly connected assets into an existing production system, the assets need to have their virtual representation equipped with data elements and services to describe their functionality. The virtual representation of assets can be implemented in a software component such as an Asset Administration Shell (AAS) as defined in Reference Architecture Model for Industry 4.0 [5]. An AAS encompasses assets by providing their digital representation and technical functionalities through semantic annotation which is coherent across the production system. The combination of AAS and its embodied assets forms the Industry 4.0 component. Fig.1 illustrates an Industry 4.0 component comprising of assets and the AAS. Open Platform Connection Unified Architecture (OPC UA) server can host the AAS and provides a semantic data-space for each asset. But, the configuration of OPC UA servers containing the relevant asset information (online data, identification data, and capability data with its metadata) is a specific know-how and time-consuming process.

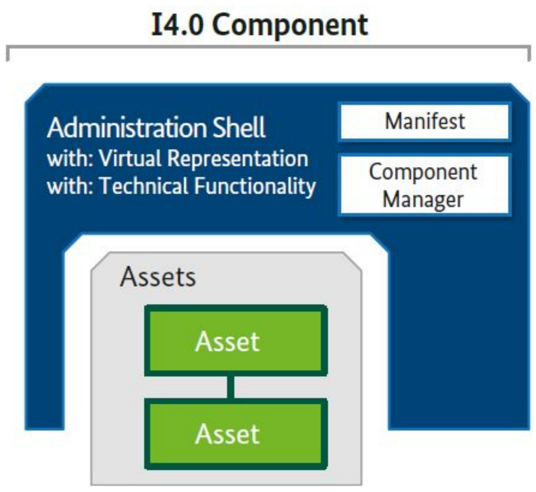


Fig. 1. Asset Administration Shell combined with assets to represent Industry 4.0 component. [6].

Analogous to the Plug & Play for computers, the aim is to have a Plug & Produce capabilities for assets to be integrated into an existing production system with minimum resource and time consumption. Therefore a method has to be defined to integrate and allow other assets to discover them across the layers of the Reference Architecture Model for Industry 4.0 [5]. This can be achieved by implementing OPC UA servers for corresponding assets, and to define a process for configuration without any manual intervention.

Section II gives an overview of OPC UA and its discovery process. Section III details the requirements for the Plug & Produce integration of components. Section IV proposes an architecture for the integration of components in OPC UA

[1]An asset can be an item that has value for an organization such as machines and devices.

based data-space. Section V details the working principle for the integration of components and section VI summarizes and concludes the paper.

II. OPC UA Discovery

OPC Unified Architecture or abbreviated as OPC UA is a platform-independent service-oriented architecture that integrates all the functionality of the classic OPC into one extensible framework [7]. Open Platform Communications (OPC) is a series of open standards and specifications developed by the OPC Foundation[2], an international organization with a goal of developing a vendor and platform independent, secure, and interoperable data exchange standard for industrial automation. In 2008, OPC Foundation published the Unified Architecture, which has the potential to become the industry standard for machine-to-machine and machine-to-enterprise communication protocol and defined in the IEC 62541 [7]. OPC UA incorporates all features of classic OPC standard like OPC Data Acess, Alarm & Events, and Historical Data Access but defines an object-oriented modelling capability with type hierarchy and inheritance for the information of a system. OPC UA defines the communication mechanism through a client-server or a publisher-subscriber model. OPC UA allows information to be connected in different ways by extending additional vendor-specific information to the OPC UA base model. Therefore various organizations like PLCopen[2] (An industry standard for PLC programming language), FDI[3], ISA 95[4], AutomationML[5] and many others developed their domain-specific information model on the top of the base information model [8].

The discovery process in OPC UA allows clients to find servers on the same network and then discover how to connect to the server to access its information [9]. The clients and servers should be implemented with Discovery services to accomplish the discovery process. The individual servers must be registered to a discovery server which provides a way for clients to find the registered servers. The different discovery servers in OPC UA are Local Discovery Server (LDS), Local Discovery Server with Multicast Extension (LDS-ME), and Global Discovery Server (GDS) [9].

A. Local Discovery Server:

The LDS is an OPC UA instance running on a host and maintains the information of all the OPC UA servers available on the same host. A client can connect to the LDS and obtain a list of servers and their discovery URL by calling *FindServer* service. Then, the client can find the endpoints of registered servers with the *GetEndpoints* service.

[2]http://www.opcfoundation.org
[2]http://www.plcopen.org/
[3]http://www.fdi-cooperation.com/
[4]https://www.isa.org/isa95/
[5]https://www.automationml.org

B. LDS with Multicast Extension:

The LDS-ME maintains the discovery information of all the servers that have been announced on the local multicast subnet. In the case of LDS-ME server, servers on the same host can register themselves directly, whereas multicast announcements detect other LDS-ME servers. In LDS-ME, the server registers with the discovery server through *RegisterServer2* service and request the discovery server to announce it on the multicast subnet.

C. Global Discovery Server:

For large systems with multiple application, a discovery server must be able to discover servers among multiple subnets where hostnames cannot be discovered directly. Thus, the discovery server for the enterprise-wide application is known as Global Discovery Server. The GDS maintains the discovery information of all the servers those are available in the same administrative domain. It also provides methods for OPC UA application to search for other OPC UA applications. The clients can use *call* service instead of *FindServer* in case of the GDS.

III. Requirements

The aim of this paper is to provide a solution for the effortless integration of digital representation of new assets into OPC UA data-space. However, there are various challenges present to achieve the desired outcome. In this section, the requirements are discussed to overcome the challenges. It only considers the requirements for field devices, to be integrated through their digital representation.

A. Semantic Device Description

An intelligent manufacturing system contains various field devices which may use different methods to describe their functionality. A device description language can be adopted for describing the digital communication characteristics of field device and its parameters– device status, diagnostic data, and configuration details. Within a Plug and Produce system, these information need to be presented in a neutral and vendor-independent way, so that all participating field devices can deal with and process them appropriately [10]. Electronic Device Description Language (EDDL) is a platform and vendor independent, text-based field device description language which can be used as the semantic device description [11]. EDDL is adopted by major field device manufacturers and provide textual approach to describe data elements, services, and user interface for field devices [11].

B. Interoperability

Interoperability represents a charateristic of a production system in which its components are capable of exchanging information with one another without any restriction. All entities involved in a Plug and Produce set up must have the same understanding and processing of the relevant information. Thus, they have to be interoperable at different levels– Physical and Data. Interoperability in a Plug and Produce set up can be defined as the degree of integration of entities based on their interaction to achieve a common goal [10]. OPC UA provides an industrial interoperable solution through its Service Oriented Architecture.

C. *Integration of New Asset*

With the increasing requirement for higher and effective production, assets are needed to be plugged into the production system and start operation automatically. However, integration of the newly connected assets in the production are still executed manually and thereby it is error-prone and time-consuming. To accomplish an effortless integration of newly connected asset, each asset must receive the network connectivity automatically and should be advertised across the production system. It is also necessary to ensure the authenticity and security during the auto-configuration [12]. Dynamic Host Configuration Protocol (DHCP) can be used to assign IP address to newly connected device through DHCP server [13]. OPC UA Discovery can be used to advertise the newly connected asset across the production system. The further can be found in Using OPC UA for the autoconfiguration of real-time Ethernet systems by Lars Dürkop et al. [13] regarding auto-configuration in real-time Ethernet system.

D. *Standardized Information Model*

An information model specifies the entities and the relationship among themselves in a production system. There are a lot of interconnected assets present in a system. Every asset should have a standard information model with syntax and semantics understood by every participant of the system to provide the topology of it. Besides that, it should be platform and vendor independent. However, it is still a primary concern in current industry methods to provide such information model because of the complex structure of production system. OPC UA Companion Specification provides a solution by providing a standard information model for various organisations. In this work, OPC UA companion specification for Devices (IEC 62541-100) [14] is being considered for the standard information meta model.

IV. Plug & Produce Architecture

The proposed architecture for Plug and Produce integration consists a controller or an industrial PC which is responsible for detecting other underlying Industry 4.0 components. Such component comprises of a field device, its own OPC UA server, and the communication ability between the server and the actual device. To achieve the Plug and Produce integration without any system specific pre-configuration, the controller or industrial PC needs to detect other underlying components automatically. OPC UA Discovery [9] provides the functionality for detecting components of a system within a network. Thus, the Plug and Produce system architecture is based on the discovery features of OPC UA. A similar hierarchical architecture has also been defined by S. Profanter et al. in ”OPC UA for plug produce: Automatic device discovery using LDS-ME” for detecting Industry 4.0 components [3].

The controller or PC has its discovery server which is used for registering the Industry 4.0 components comprising of field devices and provides the connection information for them. Each Industry 4.0 component provides the information model through its server’s *AddressSpace*. The information model is based on the mapping between the semantic description of the device and the OPC UA nodes and attributes. Every device server is configured automatically by generating configuration files at run-time and configured them without any manual interventions to avoid resource and time consumption. Servers of devices also establish a run-time communication with their respective device to retrieve or update data at run-time. OPC UA clients can connect to the discovery server and fetch the endpoints for registered servers and connects to the servers to access the *AddressSpace*. Fig.2 illustrates the Plug and Produce architecture considered in this paper.

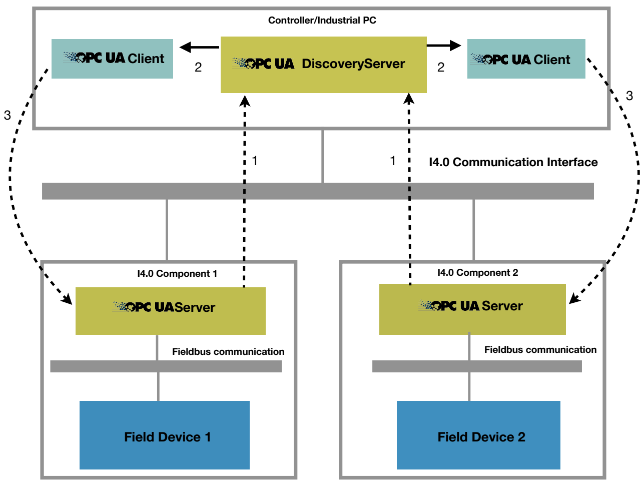


Fig. 2. Architecture for Plug and Produce system. 1. Server registers with the Discovery server. 2. Client fetches the endpoints. 3. Client connects to the server.

V. Proposed Plug and Produce Working Principle for Field Devices

In a Plug & Produce capable production system, It is essential to integrate the digital information in an easy way so that time consumption and error would be minimal. Thus, an automated configuration process is defined to integrate the semantic data-spaces of newly connected assets to overcome the challenges. The concept is based on an automated generation and execution of the server configuration scripts, which are performed after an asset is connected to the production system. The working principle is only concerned to the configuration of field devices as defined in the architecture and other higher level assets are not considered in this principle. The steps followed by the process are Physical Connection, Discovery, Description, Generation, and Configuration [15].

A. *Physical Connection*

Physical Connection is the first phase of a Plug and Produce arrangement where the I4.0 component needs to be connected to a production system via wired or wireless methods. While connecting a new component into an existing set up, all necessary measures should be followed to prepare the production system for the reconfigure process, e.g., pausing certain services or putting the system into a degraded mode [16]. After connecting the component, the halted system or services can be brought back to operational mode.

B. Discovery

The Discovery process allows the asset to be identified by the Plug and Produce system to start the automated configuration process.

In an Ethernet-based system, the component must be assigned with an IP address via DHCP (Dynamic Host Configuration System) as soon as it is connected to the Plug and Produce system [17]. A DHCP server dynamically assigns an IP address, manages the TCP/IP settings, and other network configuration parameters to each connected devices in a network [17]. If a component is not enabled with DHCP, then Auto IP can be used to assign an IP address [17]. After the discovery process, the system will start further steps for the integration of the component.

For an offline component, e.g., a field device without an Ethernet-based control system, the discovery process can be obtained by checking for new field devices connected to the system concerning their Fieldbus addresses. When a field device is added, the discovery process checks whether the Fieldbus address is already present in the system, if not then the process considers the field device as newly plugged in and starts further processes for the Plug and Produce implementation.

C. Description

1) EDDL - Semantic Device Data Description: Many field devices from different manufacturers use various device integration technologies in a production system. Even devices from the same manufacturer may use different integration technology. Among these various technologies, EDDL and FDT/DTM are the commonly used integration techniques for field devices. Besides these technologies, Automation Markup Language or known as AutomationML [18] can also be used to provide the device description. In this work, EDDL has been considered as the device description language for field devices as it is text-based, platform independent, and used by major field device manufacturers.

2) OPC UA Information Model from EDDL: After the field device is discovered, the system must retrieve the EDDL description of the discovered device to create the information model. The EDDL description of the device can reside inside the device, inside the Plug and Produce system, or inside the cloud. In this principle,Electronic Device Description (EDD) files based on EDDL are located inside the Plug and Produce system. The system identifies the EDD file of the device through its Fieldbus address. EDD files are provided by device manufacturers either in text or binary format. In this work, only text-based EDD files are considered and also a mapping has been considered to map EDDL elements to OPC UA nodes.

The EDDL elements and the OPC UA NodeClass mapping is based on the FDI Technical Specification part 5 [19], which defines a mapping between EDDL elements and FDI package information and its underlying OPC UA information model. The field device is always represented as an OPC UA Object. The EDDL elements and OPC UA node class mapping is provided in the following table.

TABLE I
EDDL TO OPC UA MAPPING

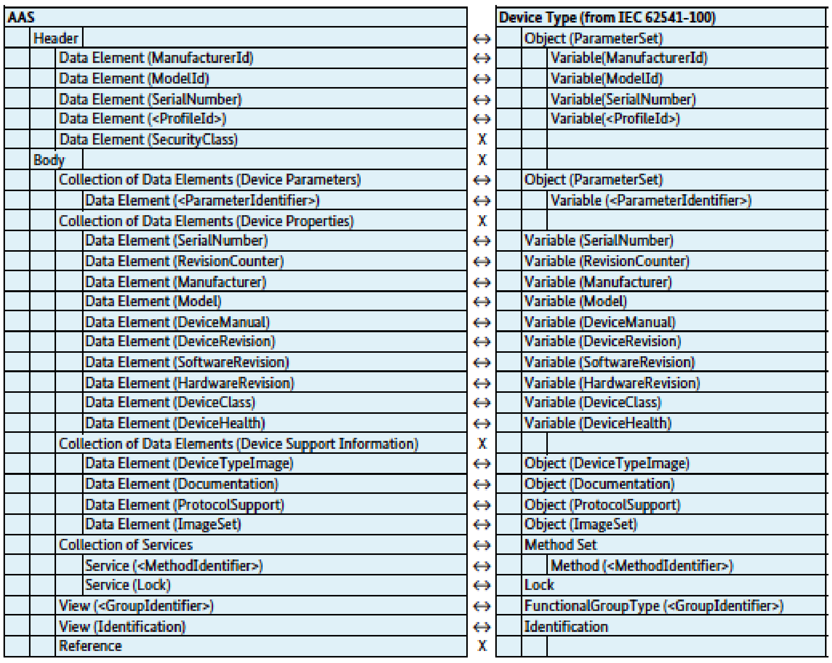
EDDL Elements	**OPC UA NodeClass**
VARIABLE	UAVariable
BLOCK	UAObject
RECORD	UAVariable
ARRAY	UAVariable
LIST	UAVariable
COLLECTION	UAVariable
METHOD	UAMethod
MENU	UAView

The characteristic of a VARIABLE element in EDDL is similar to the *UAVariable* in OPC UA. The *UAVariable* contains value and attributes such as *DataType, BrowseName, DisplayName, Description, AccessLevel, UserAccessLevel*. The VARIABLE identifier in EDDL is the *BrowseName* of the corresponding *UAVariable*. The *DisplayName* is the LABEL attribute of the EDDL VARIABLE. The *Description* of the *UAVariable* is the HELP attribute of the corresponding EDDL VARIABLE. The HANDLING attribute is the *AccessLevel* of the *UAVariable*. If the HANDLING of the EDDL VARIABLE is READ then the *AccessLevel* is *CurrentRead*, if it is WRITE then the *AccessLevel* is *CurrentWrite*, and if it is missing, then the *AccessLevel* will be both *CurrentRead* and *CurrentWrite*. The *UserAccessLevel* is defined based on the restriction to the user by the OPC UA server. The TYPE attribute can be transformed to the *DataType* attribute of the *UAVariable*. The other EDDL elements also follow the same mapping which is used for the VARIABLE.

The information metamodel is also based on OPC UA companion specification for Devices (IEC 62541-100) [14] and its mapping to AAS [16]. According to IEC 62541-100, the field device data elements can be expressed as *UAVAriable* in the address space, services can be expressed as *UAMethods*. The AAS header can be mapped to the *Identification* object in the address space. The combination of *ParameterSet* and *Methodset* can be defined as the body of the AAS. The detailed mapping between AAS and OPC UA for Devices is provided in Fig. 3.

D. Generation

The generation of OPC UA configuration file is performed after the information model is created. It incorporates the information model for the field device, which encompasses the OPC UA base nodeset, device companion specific nodeset, and the device information model. It also includes necessary OPC UA stack specific functions. It includes the discovery feature



AAS		Device Type (from IEC 62541-100)
Header	↔	Object (ParameterSet)
Data Element (ManufacturerId)	↔	Variable(ManufacturerId)
Data Element (ModelId)	↔	Variable(ModelId)
Data Element (SerialNumber)	↔	Variable(SerialNumber)
Data Element (<ProfileId>)	↔	Variable(<ProfileId>)
Data Element (SecurityClass)	X	
Body	X	
Collection of Data Elements (Device Parameters)	↔	Object (ParameterSet)
Data Element (<ParameterIdentifier>)	↔	Variable (<ParameterIdentifier>)
Collection of Data Elements (Device Properties)	X	
Data Element (SerialNumber)	↔	Variable (SerialNumber)
Data Element (RevisionCounter)	↔	Variable (RevisionCounter)
Data Element (Manufacturer)	↔	Variable (Manufacturer)
Data Element (Model)	↔	Variable (Model)
Data Element (DeviceManual)	↔	Variable (DeviceManual)
Data Element (DeviceRevision)	↔	Variable (DeviceRevision)
Data Element (SoftwareRevision)	↔	Variable (SoftwareRevision)
Data Element (HardwareRevision)	↔	Variable (HardwareRevision)
Data Element (DeviceClass)	↔	Variable (DeviceClass)
Data Element (DeviceHealth)	↔	Variable (DeviceHealth)
Collection of Data Elements (Device Support Information)	X	
Data Element (DeviceTypeImage)	↔	Object (DeviceTypeImage)
Data Element (Documentation)	↔	Object (Documentation)
Data Element (ProtocolSupport)	↔	Object (ProtocolSupport)
Data Element (ImageSet)	↔	Object (ImageSet)
Collection of Services	↔	Method Set
Service (<MethodIdentifier>)	↔	Method (<MethodIdentifier>)
Service (Lock)	↔	Lock
View (<GroupIdentifier>)	↔	FunctionalGroupType (<GroupIdentifier>)
View (Identification)	↔	Identification
Reference	X	

Fig. 3. Mapping between AAS and OPC UA for Devices [16].

of OPC UA to be registered with the local discovery server. It also incorporates all the Fieldbus specific functions to initialize the field device, also to establish a communication between the server and the actual device. The read or write of the value of parameters is incorporated based on the COMMAND of the EDDL file.

OPC UA servers are developed with the help of an OPC UA stack, which is developed in addition to the basic functionality provided by the OPC Foundation. An OPC UA stack has to be selected which supports all the requirements of this work. Thereby the configuration file generated for OPC UA server is stack specific and further implemented according to the requirement of the OPC UA stack.

E. Configuration

The discovery features are set according to the OPC UA Specification Part 12 [9], to be able to configure an Industry 4.0 component. A local discovery server with multicast extension is configured through multicast DNS implementation. The *RegisterServer2* service allows servers to register themselves on the same subnet as the discovery server. A discovery client is implemented to get a list of all the registered servers through the *FindServersOnNetwork* service. Also, *GetEndpoints* service allows returning the endpoints for all the registered server.

Each field device has its own OPC UA server, and is configured by including the information model XML file and the generated server configuration file. The requirement to configure the servers are to install all the stack-specific requirements.

After the run-time server configuration files are generated, this process configures the OPC UA servers automatically and executes them. The automated process executes the stack specific process to create the server executable and run it. Thus, the whole Plug and Produce integration is automated without any manual intervention. It also incorporates the value callback mechanism, which is invoked before reading or after writing a variable. The server reads or writes the parameters of the field device at run-time through the value callback mechanism. As soon as the server starts running, it registers with the running LDS-ME server. Then the client fetches the endpoint and connects to the server to access the device.

VI. Conclusion

The goal of this paper was to define and implement the Plug and Produce integration of components into OPC UA data-space. The requirements of a Plug and Produce integration for Industry 4.0 components were defined based on state of the art technologies, as well as work on the specific needs of an I4.0 production system. A concept fulfilling the requirements was detailed. An automated process has been proposed based on the discovery features of OPC UA to configure the respective OPC UA server for field devices. The automated process is capable to discover newly connected assets and starts the configuration as well as the execution of the OPC UA servers and made the assets operational. Also, OPC UA Discovery feature enables the assets to be discoverable across the production system.

This paper finds no objection to use OPC UA as the Industry 4.0 standard communication protocol for the Plug and Produce integration of components. OPC UA provides a rich and generic standard information model which can be mapped to the AAS expressing the data elements and services.

OPC UA Discovery process allows easy integration of I4.0 components into the network without any network specific pre-configuration. Different hardware from different vendors can be combined to reduce the cost of the implementation due to the standard interface described in OPC UA. In the long term, a real vendor independent Plug and Produce integration can be achieved with OPC UA, based on Industry 4.0 standards. Further the working principle can be extended to other Industry 4.0 components besides field devices for the Plug & Produce integration.

An interface is being designed entirely on a PC as a proof of the concept by using Open62541[6] stack and PROFIBUS PA devices.

Acknowledgement

This work is supported by the German Federal Ministry of Education and Research in the project TACNET4.0: Highly reliable and real-time 5G networking for industry 4.0 The tactile Internet for production, robotics and digitalization of the industry.

References

[1] T. J. Schoepf, “White paper: The road to plug-and-produce,” Lumberg Automation, Tech. Rep., 2016.

[2] A. Gilchrist, *Industry 4.0: The Industrial Internet of Things*. Apress, 2016. [Online]. Available: https://books.google.de/books?id=DQDwjwEACAAJ

[3] S. Profanter, K. Dorofeev, A. Zoitl, and A. Knoll, “Opc ua for plug produce: Automatic device discovery using lds-me,” in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2017, pp. 1–8.

[4] L. Dürkop and J. Jasperneite, *Plug & Produce“ als Anwendungsfall von Industrie 4.0*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 1–14.

[5] *Reference Architectural Model Industrie 4.0 (RAMI 4.0): An Introduction*. Plattform Industrie 4.0, April 2016.

[6] “Structure of the administration shell,” Federal Ministry for Economic Affairs and Energy (BMWi), Tech. Rep., April 2016.

[7] Unified architecture. [Online]. Available:https://opcfoundation.org/about/opc-technologies/opc-ua/, Access date: May 2018.

[8] “Opc unified architecture specification part 5: Information model,” OPC Foundation, Tech. Rep., November 2017.

[9] “Opc unified architecture specification part 12: Discovery,” OPC Foundation, Tech. Rep., July 2015.

[10] M. Schleipen, A. Lüder, O. Sauer, H. Flatt, and J. Jasperneite, “Requirements and concept for plug-and-work,” *at-Automatisierungstechnik*, vol. 63, no. 10, pp. 801–820, 2015.

[11] M. Riedl, R. Simon, and M. Thron, *EDDL: electronic device description language*. Oldenbourg Wissensch.Vlg, 2002. [Online]. Available: https://books.google.de/books?id=HKVQAAAAMAAJ

[12] D. Lang, M. Friesen, M. Ehrlich, L. Wisniewski, and J. Jasperneite, “Pursuing the vision of industry 4.0: Secure plug and produce by means of the asset administration shell and blockchain technology,” July 2018.

[13] L. Dürkop, J. Imtiaz, H. Trsek, L. Wisniewski, and J. Jasperneite, “Using opc-ua for the autoconfiguration of real-time ethernet systems,” in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, July 2013, pp. 248–253.

[14] “Opc unified architecture for devices companion specification,” OPC Foundation, Tech. Rep., July 2013.

[15] T. Schroeder and C. Diedrich, “Zero-click-configuration von opc ua-servern für die umsetzung von verwaltungsschalen,” in *AUTOMATION 2018, Seamless Convergence of Automation & IT*, 2018.

[16] “Industrie 4.0 plug-and-produce for adaptable factories: Example use case definition, models, and implementation,” Federal Ministry for Economic Affairs and Energy (BMWi), Tech. Rep., June 2017.

[17] I. C. et. al., “Upnp device architecture,” UPnP Forum, Tech. Rep., October 2008.

[18] Automationml. [Online]. Available:http://www.automationml.org, Access date: June 2018. [Online]. Available: http://www.automationml.org

[19] “Fdi technical specification part 5: Fdi information model,” FDI Cooperation, LLC, Tech. Rep. FDI-2025, November 2014.

[6]www.open62541.org