Automation of an industrial OPC UA FX use case through the usage of proactive Asset Administration Shells

Katharina Justmann[1], Ludwig Leurs[2], Jesko Hermann[3], Martin Ruskowski[4]

Abstract: Flexible and order-driven production are essential concepts for smart factories that effectively carry out lot-size-one production. These concepts depend on manufacturer-independent information exchange throughout the production lifecycle. While Open Platform Communication Unified Architecture Field eXchange (OPC UA FX) enables standardized communication between shop floor controllers, its configuration requires additional effort. To achieve flexible communication in changing production environments the configuration needs to be automated. The Asset Administration Shell (AAS) and the associated bidding process offer a potential solution to achieve this automation. The concept of a combination of the AAS and OPC UA FX was already formulated [DWR22a]. In this paper this concept is extended with activities of the Capability, Skill, and Service model to automate the configuration and production planning process of an OPC UA FX demonstrator. This paper shows that the information models and concepts are complementary yet lack generic mappings between the defined structures and semantics.

Keywords: OPC UA Field eXchange (OPC UA FX), Asset Administration Shell (AAS), Bidding Process, Capability Skill and Service model, Industry 4.0

1 Introduction

In Industry 4.0 (I4.0), one of the main goals is to achieve flexible manufacturing by enhancing communication between the various assets that are part of the production process. Different technologies have been developed and standardized throughout the last years to achieve manufacturer independent communication. These technologies establish the foundation for flexibility in production processes and the mass production of customized lot-size-one products. Two of these standards are the Asset Administration Shell (AAS) and Open Platform Communication Unified Architecture Field eXchange (OPC UA FX). This paper shows the necessary automation of the configuration process

[1] Bosch Rexroth AG, Berliner Str. 25, 64711 Erbach, Germany, Katharina.Justmann@boschrexroth.de
[2] Bosch Rexroth AG, Bürgermeister-Dr.-Nebel-Straße 2, 97816 Lohr am Main, Germany, Ludwig.Leurs@boschrexroth.de
[3] Technologie-Initiative SmartFactory KL e.V., Trippstadter Str. 122, 67663 Kaiserslautern, Germany, Jesko.Hermann@smartfactory.de
[4] RPTU Kaiserslautern, Lehrstuhl WSKL, Gottlieb-Daimler-Str. 42, 67663 Kaiserslautern, Germany, Martin.Ruskowski@rptu.de

defined in the OPC UA FX specifications by applying these two standards to the already existing OPC UA FX demonstrator of the OPC Foundation [OP23a].

The AAS represents information and capabilities of assets in a virtual context. This information is represented by defined submodules of the AAS, allowing the standardized definition of different aspects of industrial assets throughout the entire asset lifecycle [ID23a]. On the other hand, OPC UA enables standardized vertical and horizontal communication in industrial automation. The Field Level Communication (FLC) initiative of the OPC Foundation has standardized OPC UA FX to enable a manufacturer-independent real-time controller-to-controller (C2C) communication on the shop floor. OPC UA FX enables the creation and reconfiguration of OPC UA PubSub communication between automation components during runtime [OP22a].

In the context of flexible production with the request of fast on-the-fly reconfiguration, the OPC UA FX specification does not propose an automated way for the configuration process. In Part 80 of the OPC UA FX specifications the configuration process is executed through control or system engineers that decide which process lines need to interact with each other [OP22b]. Additionally, some configuration parameters needed for establishing a connection are provided during the engineering stage. To improve these matters, defined AAS processes can help. A first conceptual approach to create a combination of the processes of the AAS and OPC UA FX was described by Ch. Diedrich et al. [DWR22a]. The combination is based on the bidding process specified in VDI/VDE 2193-2 [VD20].

This publication will further develop the concept by applying it to an industrial use case implemented in the OPC UA FX demonstrator of the OPC Foundation. A group of controller manufacturers designed and implemented the demonstrator to exhibit the applicability and interoperability of OPC UA FX in a flexible production use case. In the demonstrator users need to choose manually which controllers should be part of the production line. In this paper the demonstrator will be enhanced to showcase the automated process of creating production lines on-the-fly based on product description and the bidding process. This is achieved by enabling the product and the automation components as I4.0 components with proactive AAS. The behaviour of the different types of AAS are described in chapter 2.4. Additionally, this paper shows how the existing Capability, Skill and Service model is used to map automation functions from OPC UA FX to the AAS. The mapped automation functions and the data stored in the AAS is then used to create OPC UA FX communication connections.

2 Background and Related Work

This chapter gives a short overview of the current state of the art regarding OPC UA FX, the Capability, Skills and Service model and latest developments about the AAS. Additionally, it will present related work like the OPC UA FX demonstrator and a concept that presents first steps for enhancing OPC UA FX with the AAS.

2.1 OPC UA Field eXchange

The OPC UA FX specifications define a manufacturer-independent horizontal communication on the field level between controllers (C2C) and in the future communication to or between field devices. The current release of the OPC UA FX standard uses PublishSubscribe (PubSub) architecture with the possibility to additionally use parts of the Time-Sensitive Networking (TSN) standard. OPC UA FX defines the entities *AutomationComponent* and *ConnectionManager*. *AutomationComponents* represent assets that can perform one or more automation functions. These functions are defined as *FunctionalEntities* with input and output parameters, configuration parameters, diagnostic information, and additional identification properties in the scope of OPC UA FX. *FunctionalEntities* can represent simple subtasks like moving a robot arm to a specific position or multiple subtasks combined into larger tasks, such as entire process steps. *AutomationComponents* also include information about the assets that are used for identification purposes. The structure of *AutomationComponents* and a *ConnectionManager* are illustrated in Fig. 1. The *ConnectionManager* connects to *AutomationComponents* via OPC UA Client/Server and calls the *EstablishConnections* method to establish the communication between two *AutomationComponents*. The *ConnectionManager* creates the communication based on *ConnectionConfigurationSets* which include all necessary parameters needed for configuration. These sets are either created through an offline descriptor that is read into the *ConnectionManager* as a file or based on inputs via methods in the OPC UA address space of the *ConnectionManager*. The established communication between two *AutomationComponents* is called logical connection and is based on the OPC UA PubSub standard [OP22a].

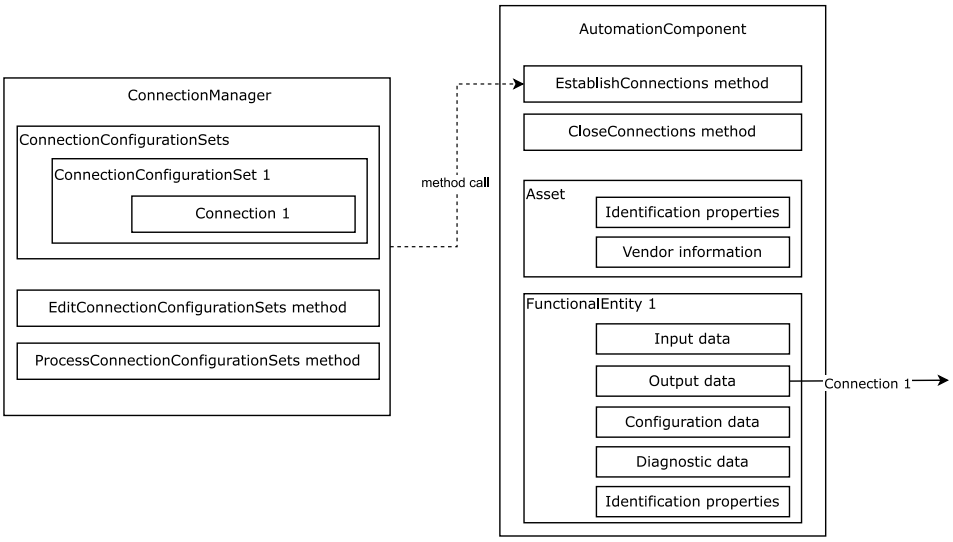


Fig. 1: Structure of OPC UA FX components

2.2 OPC UA FX Demonstrator

To showcase the OPC UA FX specifications various manufacturers implemented an OPC UA FX demonstrator in the Prototyping technical working group that is part of the Unified Architecture Field eXchange working group of the OPC Foundation [OP23b]. This demonstrator presents the interoperability and on-the-fly configuration and reconfiguration during runtime of OPC UA PubSub communication connections. The controllers in the demonstrator are setup in a star network topology. The chosen user story for the demonstrator is a beer bottling plant. Each controller is an *AutomationComponent*, and each process step is modelled as a *FunctionalEntity*. The output parameter of a *FunctionalEntity* corresponds to the input parameter of the successor *FunctionalEntity*. The process steps range from selecting and washing different sizes of bottles to capping and labelling a bottle. For each process step, more than one controller represents a step to allow for more customization possibilities and to create more than one process line in parallel. For example, one controller fills a bottle with Pilsner, while another would be responsible for bottling stout beer. An overview of the demonstrator with a possible setup of communication can be seen in Fig. 2. Through a user interface of the *ConnectionManager* the user can manually select the *AutomationComponents* and the *FunctionalEntities* available in the controller that should be connected. Besides that, the user additionally decides on other parameters like publishing intervals, timeouts and the interface used for transmission. Based on this input the *ConnectionManager* creates the OPC UA PubSub configuration. The *ConnectionManager* then connects via OPC UA Client/Server to the selected *AutomationComponents* and establishes the connections between the controllers. The user needs to repeat these configuration steps for each connection that needs to be established for the entire production line.

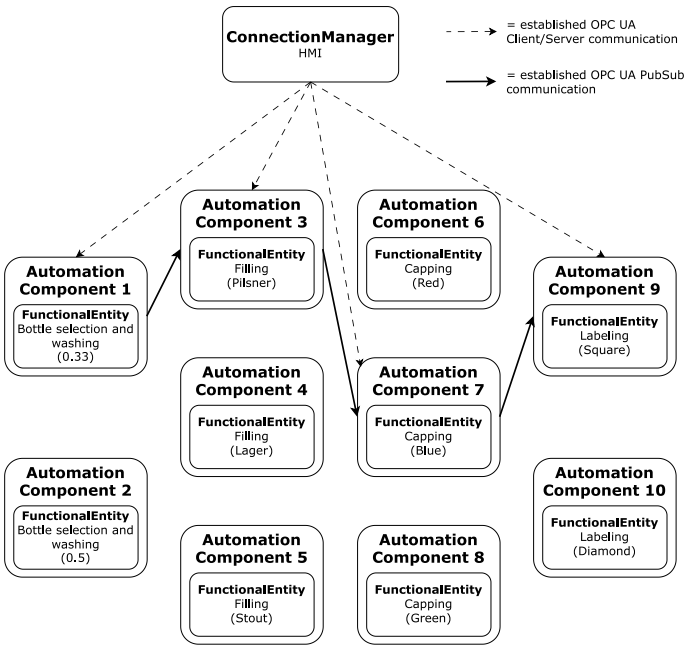


Fig. 2: Example for established communication in the OPC UA FX demonstrator

2.3 Capability, Skill, and Service Model

A working group of the "Plattform Industrie 4.0" initiative developed the Capability, Skill, and Service model (CSS model) which extends the Product, Process, and Resource model (PPR model) [Di22b]. Resources in the scope of the PPR model produce products by completing process steps, like a labelling machine for instance that puts a label onto a surface. This is one sub-process in the scope of the production process of bottling beverages for example. The implemented and executable task of the labelling machine is represented as a Skill in the CSS model. A Skill consists of a *SkillInterface* and Parameters. The *SkillInterface* is used to execute a Skill while the Parameters represent the input, output, control, and diagnostic parameters of the Skill. The abstract description of a Skill is the so-called Capabilities of a production resource. Capabilities have *CapabilityConstraints*. These constraints are conditions that need to be fulfilled either prior to, during or following the execution of the task that is linked to the Capability in question. *CapabilityConstraints* can be used to create an order of the required Capabilities if production steps are dependent on each other. Services are a representation of Capabilities in an economic sense. They extend the description of the Capability with information such as energy consumption of the production process, delivery dates, costs or even details about laws that are followed [Di22b]. Capabilities and Services also have properties. These properties describe characteristics of the entities. Capability properties

can also have direct impacts on the parameters of Skills. For example, if a Capability's property describes the colour of a label, this value serves as an input for the corresponding Skill. Additionally, the CSS model defines various activities that describe how Capabilities, Skills and Services are used in an industrial use case [Di22b].

2.4 Asset Administration Shell

The AAS is a standardized way to create I4.0 components by enabling manufactures to provide the digital twin of an industrial asset. I4.0 components are industrial assets consisting of their physical and virtual representation [BVZ15]. Through various submodules the AAS can represent different aspects of assets like functionality or physical dimensions. Currently a working group of the Industrial Digital Twin Association (IDTA) is in the process of defining a submodule to represent Capabilities in the AAS as specified in the CSS-model [ID23b]. Besides the stored information also interaction patterns and semantics for an I4.0 language are defined in the scope of the AAS specifications. Three interaction patterns, passive, reactive, and proactive, are specified for the AAS. The first interaction pattern supports only passive interactions where the asset information is stored as a file format. A reactive AAS can provide the asset information via a server interface. The last type allows the AAS to proactively negotiate with another AAS. These proactive AAS use a bidding process to interact [VD20, Pl20]. When equipping an asset with a proactive AAS, the AAS consists of a reactive or passive AAS and a proactive part. The passive or reactive AAS is responsible for the storage of the asset information while the reactive part is in control of handling interactions and the bidding process [Gr22]. The bidding process defined in VDI/VDE 2193-2 describes how a customer requests different services from a contractor to fulfil the required process steps to produce a product [VD20].

2.5 Enhancing OPC UA FX with the bidding process

The concept used to enhance the OPC UA FX demonstrator was first published by Ch. Diedrich et al. [DWR22a]. The publication describes a concept for the combination of OPC UA FX and the AAS. The combination enables the configuration via OPC UA FX with the help of data from the AAS. Different submodules of the AAS contain the information needed for the *ConnectionManager* to configure the communication between the *AutomationComponents* of two adjacent process steps. Which *AutomationComponents* need to be connected is decided based on the outcome of a bidding process. One *AutomationComponent* in the scope of OPC UA FX provides one or more *FunctionalEntities*. These *FunctionalEntities* are mapped to the Services presented in the AAS. A ServiceProvider then makes an offer based on the available Service. A ServiceRequester can accept this offer based on different parameters like cost or energy consumption. If a ServiceRequester accepts the offer, the additional configuration parameters needed for the *ConnectionManager* are read out of the AAS. These parameters are then given to the *ConnectionManager*, which automatically establishes the connections between the controller of each process step [DWR22a].

3 Process of establishing connections through the bidding process

In this chapter the bidding process in the scope of the OPC UA FX demonstrator is shown. The sequence diagrams Fig. 3 to Fig. 5 illustrate the described process.

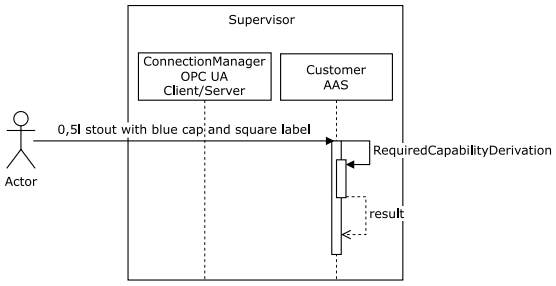
To automate the OPC UA FX demonstrator via a bidding process each *AutomationComponent* is equipped with a reactive and a proactive AAS [Gr22]. The information about each of these automation components, its Capabilities, its asset description and its OPC UA Server connection information are represented in the AAS to allow an automated configuration.

In the bidding process defined in VDI/VDE 2193-2 the user first provides either a description of the requested product or a set of processes or required Capabilities as defined in a submodule template. Such a product description could include description of the physical dimensions and features of the requested product or similar specifications. Regarding the OPC UA FX demonstrator use case the user can select a set of features for producing a bottle via an HMI. Using the following example product description, the process of handling this description, the bidding process and the following C2C configuration will be shown. The example description reads as follows: a 0.5-litre bottle of stout beer with a blue cap and square label.

This description is now used to generate a production order and create a digital representative of the product. This representative consists of the product features, requirements and later also the necessary production steps to create the product. The digital representative is an instance of an active and proactive AAS, allowing the product to be an I4.0 component and participating in the bidding process to directly create orders for its own features. The benefit of this is that information about the product is directly attached and stored during the production process. This information can later be used to comply with laws like the required digital product passport that is currently discussed in the European Union. Furthermore, a product that is represented as a I4.0 component allows more dynamic on-the-fly configurations that align with the concepts of adaptable manufacturing and self-organizing supply chains [Gr22].

An algorithm breaks the production description into features and smaller subtasks. Following the CSS model the derived subtasks or features can then be mapped either to Services or to Capabilities, depending on the additional requirements for the bidding process. When dealing with Shared Production involving multiple companies, also different laws or other non-production parameter need to be considered. In such cross-company settings, it is required to map the subtasks to Services. The OPC UA FX demonstrator showcases a use case for an adaptable factory that concentrates on internal production processes. In the scope of this use case derived features are mapped to Capabilities using the activity *RequiredCapabilityDerivation* defined in the CSS whitepaper [Di22b]. The given example results to a derivation of four Capabilities. The four Capabilities are washing a 0,5-litre bottle, bottling stout beer, closing the lid with a blue cap, and finally attaching a square label to the bottle. In the given example, the bottle

must be washed before it can be filled, and the capping process step needs to be executed after filling. These dependencies are expressed through *CapabilityConstraints* [Di22b] and are used to outline a work plan. The work plan is then stored in the AAS of the product. The resulting Capabilities of the *RequiredCapabilityDerivation* with the corresponding parameters and constraints are shown in Fig. 3.



Result			
	Capability	Parameter	Constraint
1.	select and wash bottle	size: 0.5l	
2.	fill bottle	content: stout	pre: finish select and wash bottle
3.	cap bottle	colour: blue	pre: finish fill bottle
4.	label bottle	shape: square	pre: finish cap bottle

Fig. 3: Sequence diagram and result of *RequiredCapabilityDerivation*

In the paper of Ch. Diedrich et al., the entities taking part in the bidding process are referred to as ServiceRequester and ServiceProvider [DWR22a]. However, as we are using Capabilities for mapping rather than Services in the scope of the OPC UA FX demonstrator these terms are not adopted in this paper. Instead, the terms specified in the specification of the bidding process, customers, and contractors [VD20], are used. The customer entity derives the required Capabilities from the product description and requests resources that can provide the required Capabilities to fulfil the production order. The request is made through a call for proposal which is send to the potential contractors [VD20]. To establish a more dynamic and optimized workflow in the demonstrator the customer submits requests for the next required production step outlined in the work plan instead of blocking a whole production line like in the original demonstrator. The contractor is the individual *AutomationComponent* which is responsible for the production step. It matches the required Capabilities from the proposal to its own offered Capabilities. This activity is defined as *CapabilityMatching* [Di22b]. If possible, the contractor can directly map the Capabilities. If not, the contractor can either reject the proposal as it is not able to fulfil the requirements or decompose the Capability into sub-Capabilities and attempt the matching again. *CapabilityMatching* not only compares the notations of Capabilities but also if the constraints are met and if the Capability properties match [Di22b]. A possible production step is the filling of the bottle with stout beer. Filling the bottle is a Capability of the *AutomationComponent*, filling it with stout is a property of the

Capability and also a constraint if the *AutomationComponent* is only able to fill bottles with stout and nothing else. Another constraint of this Capability is before filling, the bottle needs to be cleaned. If a set of matching Capabilities is found the contractor checks if it is feasible to produce the requested product based on the matched Capabilities, the given properties, and the capacities of the production resource using the *FeasibilityCheck* activity defined in the CCS whitepaper. Additionally, the *FeasibilityCheck* can calculate the process times and the costs of a Skill execution [Di22b]. This can be used in production settings where the aim is to optimize production with regards to changeover times. Within the scope of the OPC UA FX demonstrator the only property that is checked is whether the *AutomationComponent* is already part of a production line or not. If the *AutomationComponent* is already part of a production line, the contractor rejects the proposal; otherwise, it sends an offer to the customer. The offer includes parameters based on which the customer is able to decide on whether or not to accept the offer. In the scope of the demonstrator the offer is always accepted. Fig. 4 illustrates the sequence of steps from the customer's call for proposal to the submission of an offer.

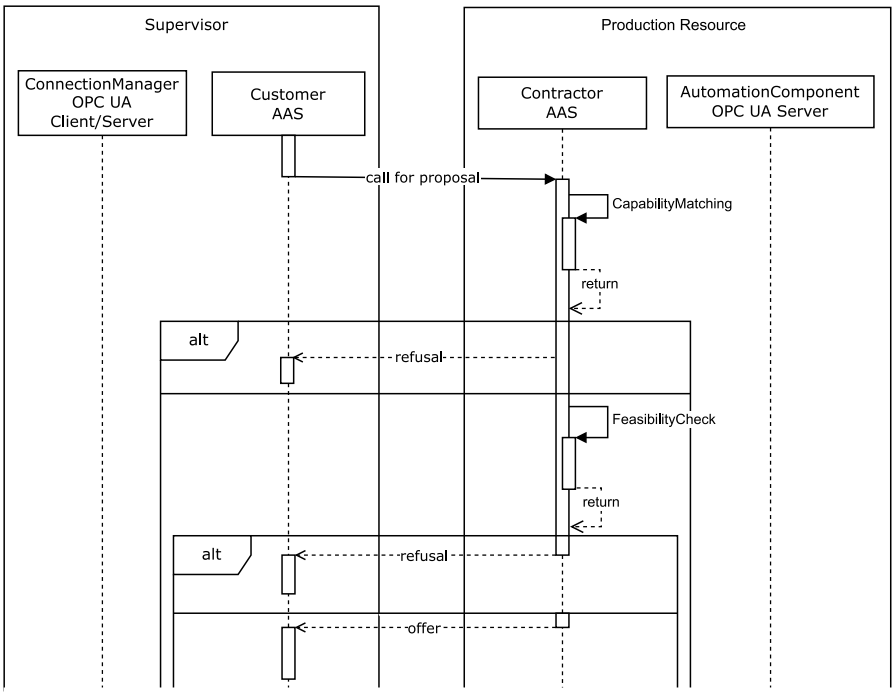


Fig. 4: Sequence diagram with *CapabilityMatching* and *FeasibilityCheck*

If the offer is accepted the configuration of the OPC UA FX communication is generated. To create such a configuration several parameters are required, which must be included in the *AutomationComponents* AAS. This enables the customer to read the configuration

parameters out of the contractor's AAS. Refer to Fig. 4 for a visual representation of this steps. The contractor passes the configuration parameters to the *ConnectionManager* entity which will create the OPC UA FX configuration. The *ConnectionManager* connects then as an OPC UA Client with the OPC UA Server of the *AutomationComponent* to call the FX specific methods for establishing a OPC UA FX connection.

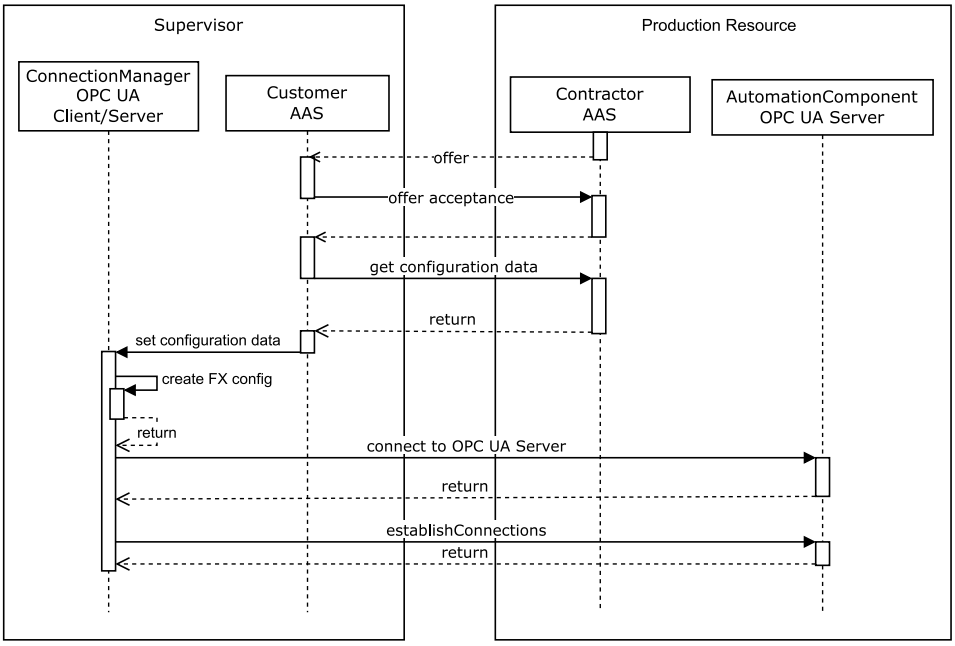


Fig. 5: Sequence diagram for getting the OPC UA FX configuration parameters

4 Conclusion and further Work

This paper utilised the concept by Ch. Diedrich et al. [DWR22a] to automate the configuration process of an OPC UA FX demonstrator during runtime in response to varying production requirements. In addition, it was shown that the CSS model offers a way to map implemented OPC UA FX *FunctionalEntities* to Capabilities and Skills within the AAS. It also illustrates how the activities defined for the CSS model enable quick on-the-fly production planning and reconfiguration of the communication between process resources even when dealing with abstract production descriptions as inputs. To fully achieve the I4.0 adaptable and optimized manufacturing concepts further work is required to implement the combination of I4.0 technologies in an actual industrial context. For instance, it is required to implement generic mappings between different domain specific semantics of various I4.0 principles.

5 References

[DWR22a] Diedrich, C.; Werner, T.; Riedl, M.: Interaktion zwischen Steuerungen auf der Basis von OPC UA FX und deren Konfiguration durch Verwaltungsschalen. In: Automation 2022, Baden-Baden, S.19-30, 2022.

[OP23a] OPC Foundation News, https://opcfoundation.org/news/press-releases/the-opc-foundation-releases-the-opc-ua-field-exchange-uafx-specifications/, Accessed 5 Oct 2023.

[ID23a] IDTA e.V.: Specification of the Asset Administration Shell Part 1: Metamodel, 2023.

[OP22a] OPC Foundation: OPC Unified Architecture Field eXchange (UAFX) Part 81: UAFX Connecting Devices and Information Model 1.00.00, 2022.

[OP22b] OPC Foundation: OPC Unified Architecture Field eXchange (UAFX) Part 80: UAFX Overview and Concepts 1.00.00, 2022.

[VD20] VDI/VDE-GMA: VDI/VDE 2193, Blatt 2 - Sprache für I4.0-Komponenten - Interaktionsprotokoll für Ausschreibungsverfahren, VDI/VDE-Richtlinien, Beuth Verlag GmbH, 2020.

[BVZ15] BITKOM e.V.; VDMA e.V.; ZVEI e.V.: Umsetzungsstrategie Industrie 4.0. Ergebnisbericht der Plattform Industrie 4.0, 2015.

[Di22b] Diedrich, C. et.al.: Information Model for Capabilities, Skills & Services. Definition of terminology and proposal for a technology-independent information model for capabilities and skills in flexible manufacturing. Plattform Industrie 4.0, Berlin, 2022.

[ID23b] IDTA e.V. Content Hub AAS Submodell Templates, https://industrialdigitaltwin.org/content-hub/teilmodelle, Accessed 29 Sep 2023.

[Pl20] Plattform Industrie 4.0: Verwaltungsschale in der Praxis. Wie definiere ich Teilmodelle, beispielhafte Teilmodelle und Interaktion zwischen Verwaltungsschalen (Version 1.0), BMWi, Berlin, 2020.

[Gr22] Grunau, S. et.al.: The Implementation of Proactive Asset Administration Shells: Evaluation of Possibilities and Realization in an Order Driven Production. In Kommunikation und Bildverarbeitung in der Automation. Technologien für die intelligente Automation, vol. 14. Springer Vieweg, Berlin u.a., 2022.

[OP23b] OPC Foundation Field Level Communications (FLC) Initiative, https://opcfoundation.org/flc/, Accessed 5 Oct 2023.