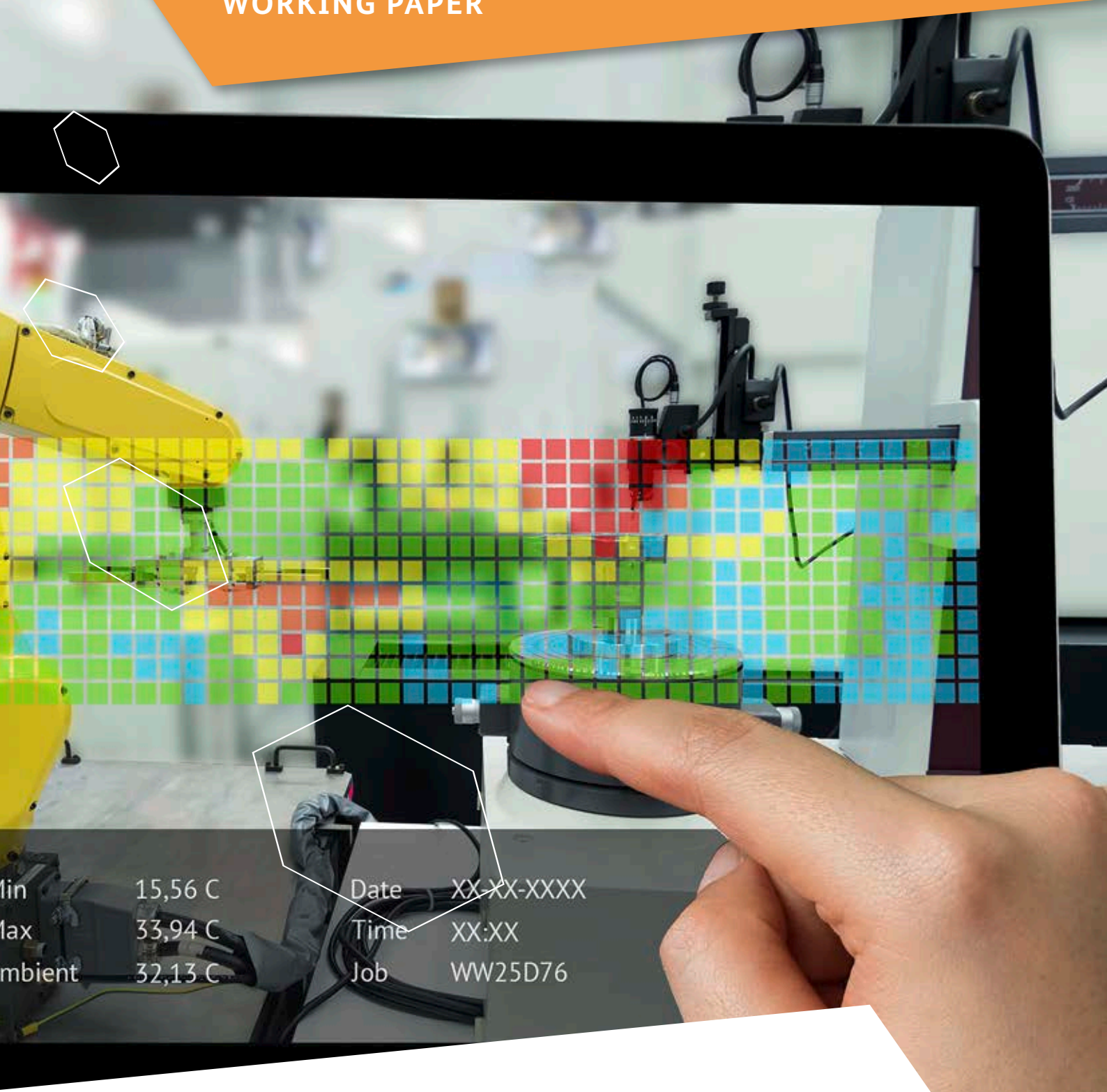
PLATTFORM INDUSTRIE 4.0

WORKING PAPER



Industrie 4.0 Plug-and-Produce for Adaptable Factories: Example Use Case Definition, Models, and Implementation

In cooperation with

ZVEI: Die Elektroindustrie

Imprint

Published by
Federal Ministry for Economic Affairs and Energy (BMWi)
Public Relations
10119 Berlin
www.bmwi.de

Text and editing
Plattform Industrie 4.0
Bertolt-Brecht-Platz 3
10117 Berlin

Design and production
PRpetuum GmbH, Munich

Status
June 2017

Print
Unterleider Medien GmbH, Rödermark

Illustrations
zapp2photo – Fotolia (Title), Nataliya Hora – Fotolia (p. 7), industrieblick – Fotolia (p. 9), Kzenon – Fotolia (p. 22), Sergey Nivens – Fotolia (p. 24), Helene – Fotolia (p. 41), MITO images – Fotolia (p. 46), Syda Productions – Fotolia (p. 56), zapp2photo – Fotolia (p. 60)

Zertifikat seit 2002
audit berufundfamilie

The Federal Ministry for Economic Affairs and Energy was awarded the audit berufundfamilie® for its family-friendly staff policy. The certificate is granted by berufundfamilie gGmbH, an initiative of the Hertie Foundation.

This publication as well as further publications can be obtained from:
Federal Ministry for Economic Affairs
and Energy (BMWi)
Public Relations
E-mail: publikationen@bundesregierung.de
www.bmwi.de

Central procurement service:
Tel.: +49 30 182722721
Fax: +49 30 18102722721

Content

1. Introduction

The Platform Industrie 4.0 aims at identifying and evaluating existing standards that are needed to implement Industrie 4.0 application scenarios across vendor borders. If the standards are not suited or incomplete for the application scenarios, these gaps shall be identified to start corresponding standardization initiatives. The "DIN Normungsroadmap" Industrie 4.0 provides a broad list of standards and norms that are potentially relevant in this context. On the other hand, the Platform Industrie 4.0 working group 2 on research and innovation has specified nine application scenarios, which may help to separate Industrie 4.0 concepts into more manageable parts.[1] These application scenarios are meant to be representative and aim to cover many aspects of Industrie 4.0. They provide a starting point for a deeper evaluation of standards, since they allow mirroring existing standards against concrete application requirements if they are further refined.

Participants of the Plattform Industrie 4.0 have created the "Reference Architecture Model Industrie 4.0" [20], which shall also assist the task of identifying, classifying, and evaluating existing standards for Industrie 4.0. It induces a top-down approach to this tasks and does not explicitly distinguish between different application scenarios that may require different standards. In contrast, this document follows a bottom-up approach and is scoped around a single application scenario (Adaptable Factories).

Figure 1 provides an overview of existing whitepapers in the context of Industrie 4.0. For the different application scenarios, different perspectives need to be analyzed. The coarse-grained descriptions need to be refined into use case specifications according to IEC 62559-2. Then, conceptual models and reference architectures can be designed to implement the use cases. Usually, the architectural description is not sufficient to evaluate standards in a meaningful way, thus prototypical implementation can help to add more depths to the discussion and find out about the practical limits of certain technologies. Subsequently, research and standardization needs can be identified and addressed.

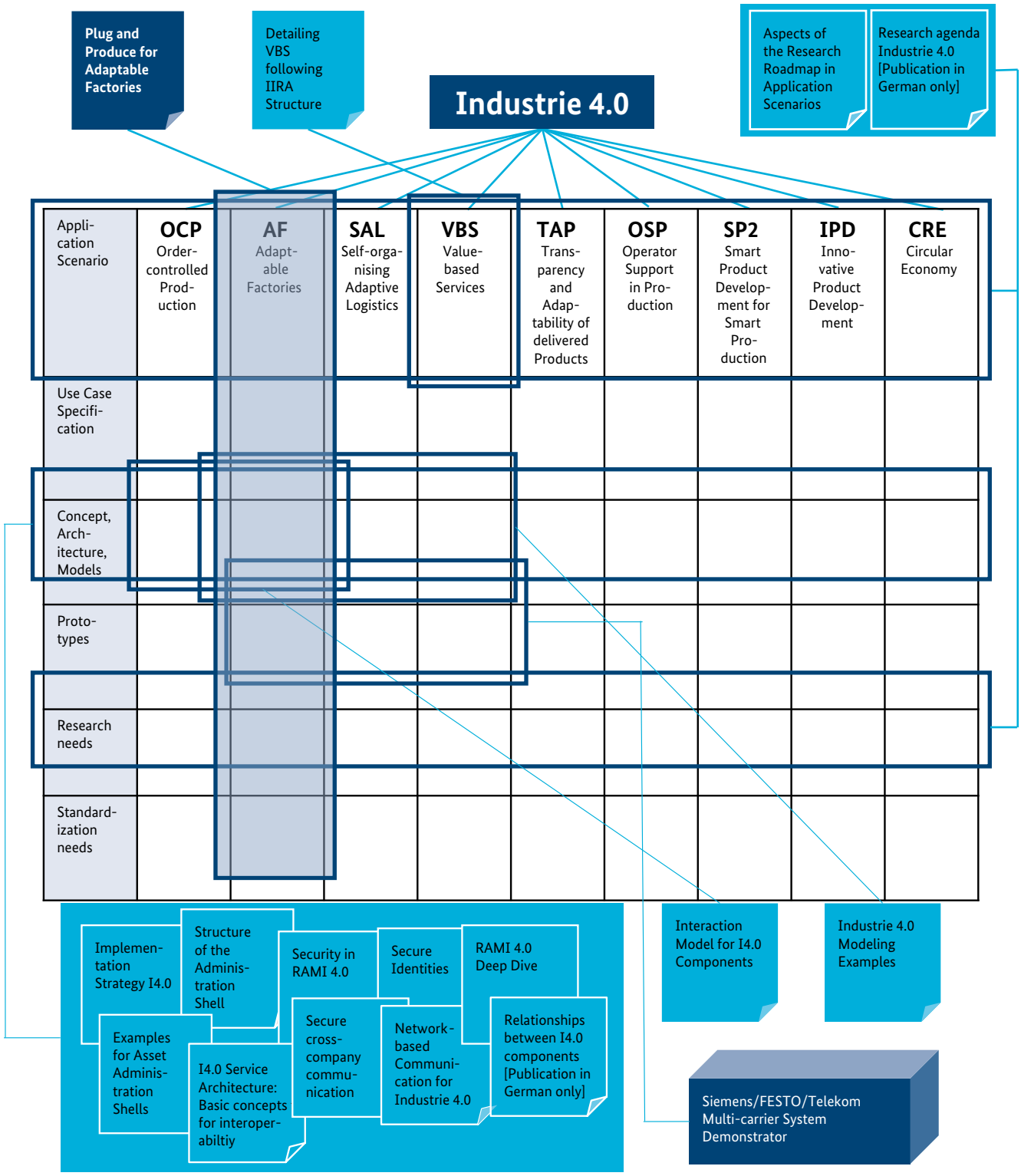
Up to now, most whitepapers elaborate concepts, architectures, and models for Industrie 4.0, often referring to existing standards in their discussions. But there is still a lack in use case specifications, prototypes, and thorough evaluation of standards. Therefore, this paper aims to make a first step to address these gaps in the specific context of the application scenario "Adaptable Factories". The application scenario itself is again multi-faceted and can be broken down into many different use cases. This document focuses on the special use case of the plug-and-produce integration of a field device into a production facility. With this restricted scope, it cannot address all facets of the whole application scenario.

In the RAMI 4.0 model, the use case mainly affects the lifecycle stage of product instances, specifically the production phase. Regarding the hierarchy levels of RAMI 4.0 the use case concentrates on Field Devices and their connection to the higher-level systems, e.g., control devices, station, work units, etc. The use case is independent of a particular product to be produced. From the RAMI 4.0 layers, mainly the communication and information layer are touched by the use case. The general assumption is that the field device as well as other assets needed for configuration and integration represent individual Industrie 4.0 components with Asset Administration Shells.

The use case has been selected, because a number of standards for its realization have already been developed in the last decade. Thus, the step to put use case into practice may not be as high as for other scenarios. This does not imply that there is no additional conceptual work required for the implementation of this use cases. Also, the existing standards may support only a restricted version of the use case and may need to be extended for even better automation.

The document provides a use case specification, a conceptual model for its realization, as well as the high-level mapping to different technologies. It also provides the description of an example realization to make the concepts more tangible. Still, the description may not be detailed enough to illustrate all facets of the use case. The aim is to use only standards for the implementation of the scenario, so that a vendor-neutral Plug&Produce becomes possible. At the end the document discusses standards useful for the example implementation and provides a first evaluation.

Figure 1: Overview of Whitepapers on Industrie 4.0



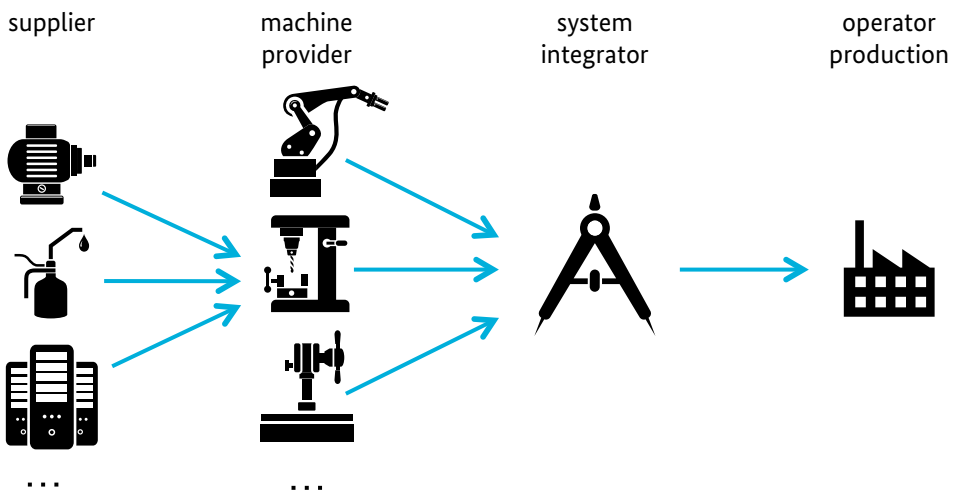
2. Application Scenario „Adaptable Factories“

To set the context, a brief summary of the application scenario „Adaptable Factories“[1] follows. This application scenario sketches a flexible production line that can be re-configured during runtime. For today's production lines, a re-configuration to manufacture a new product variant or to benefit from new production capabilities implies a significant overhead in manual and thus expensive work. This leads to rather static production facilities, whose owner are reluctant towards innovation and process changes.

Analogous to the "plug-and-play" concept for desktop computers, the vision is to have "plug-and-produce" capabilities for future production lines. New field devices or production modules shall be integrated into production lines with minimal or no manual overhead, thus greatly increasing the production flexibility, while preserving any pre-cautions for safety and preservation of intellectual properties. This can also aid rapid "scaling-up" and "scaling-down" production lines in case of fluctuating customer demands.

Such an adaptable factory would consist of modular production facilities that are intelligent and interoperable and can be easily re-composed or extended to address customer needs. Today (Figure 2), a system integrator has the responsibility to setup the production facilities and developing a control system for the entire factory. In an adaptable factory, the production modules would contain a machine-readable, semantically unambiguous self-description of their properties and capabilities. This enables a (semi-)automatic integration of the production modules to achieve a production goal and may drastically reduce the manual work of today's system integrator.

Figure 2: The "Adaptable Factory" value network (1)



There are a few concrete examples from this application scenario. A new field device automatically receives network connectivity and is advertised in the system. Another field device replaces an older one and can import the parametrization of the device it replaces to reduce the configuration effort. Upon modifying the production facility, control-related changes are detected and propagated to all relevant systems. It shall be possible to generate new visualizations from the production facilities for a Manufacturing Execution System (MES).

The application scenario can be refined into many, distinct use case specifications. Examples are:

- Plug&Produce for individual field devices for basic operation
- Plug&Produce for individual field devices with skills negotiation
- Plug&Produce for production modules using NAMUR Module Type Package (MTP)
- Plug&Produce for MES systems managing self-contained production resources
- Plug&Produce for autonomous work pieces initiating their own production

In the scope of this paper, only the first listed use case (PnP for field devices) is detailed and analyzed. This is not intended to devalue other use cases, rather it should considered as a starting point. It was selected, because available standards already cover significant parts of it, so that an implementation based on standards is almost possible without additional standardization efforts. The full application scenario "Adaptable Factories" cannot be covered by this single, selected use case. However, many concepts from the deeper analysis of this use case can potentially be reused for other use cases as well.

3. Use Case
„Plug And Produce for Field Devices”

The following subsections are structured along the use case template from IEC 62559-2. While the structure and the content aims to be self-contained, readers should make themselves familiar with the template to better comprehend the content. In the context of this whitepaper, the following specification should be considered as a preliminary proposal for a use case specification, not a finalized artifact. Eventually, the specification may require refinement from domain experts on an international scale, which was not possible in the creation process of this white paper. The specification aims to be technology-neutral, but provides references to standards that maybe helpful for implementation.

3.1 Description of the Use Case

3.1.1 Name of Use Case

Use Case Identification		
ID	**Area/Domain(s)/ Zone(s)**	**Name of Use Case**
AF01	Industrial Automation Systems/Instance Production/Field Device	Plug and Produce for Field Devices (PnPFD).

3.1.2 Version Management

Version Management				
Version No.	**Date**	**Name of Author(s)**	**Changes**	**Approval Status**
0.1	2016-11-23	Heiko Koziolek	Initial Draft	Working draft
0.2	2017-03-30	Heiko Koziolek	Filled out all sections.	Working draft
1.0	2017-04-26	Heiko Koziolek	Incorporated reviewer comments.	Ready for approval.

3.1.3 Scope and Objectives of Use Case

Scope and Objectives of Use Case	
Scope	PnPFD automates the integration and configuration of a field device within a production system. It is concerned with the process of connecting a field device to a plant network, configuring it automatically, and enabling it to participate in a production process.
Objective(s)	The use case aims at reducing commissioning times for field devices. This could not only speed up installing and maintaining a field device, but may make the whole production process more flexible since changes become effortless. This in turn can make more product variants economically feasible supporting customer desires for more individualized products.
Related business case(s)	To be determined.

3.1.4 Narrative of Use Case

Narrative of Use Case
Short description
The use case consists of six phases[1]: physical connection, discovery, basic communication, capability assessment, configuration, and integration.
Complete description
1. **Physical Connection:** Any new or replaced device needs to be plugged into the network, either by using a cable or wireless connection. This step may involve additional procedures to prepare the network for the reconfiguration, e.g., pausing certain services or putting the system into a degraded reconfiguration mode. 2. **Discovery:** After physical connection, other devices or a device management server need to realize the presence of a new device in order to start the automated integration. For example, pings, broadcasts, or IP scanners are means to find connected devices on a network. 3. **Basic Communication:** This step opens a simple communication between the device management server and the connected device. The device management server retrieves basic information, e.g., a device description. Real-time communication is not required for this step. 4. **Capability Assessment:** The device management server evaluates the identity, functionality and requirements of the new device based on the device description retrieved in the former step. This may involve interpreting the device description and matching the production requirements to the device capabilities to determine how to (re-)configure the device and the overall production system. 5. **Configuration:** In this step the device information needs to be integrated into the existing network system for example to allow for real-time configuration. Plant information models may need to be updated to reflect the new situation. Parameters of the device itself may be set based on the needs of the production system. This may involve human interaction or automatic inference of certain parameters. 6. **Integration:** Besides configuring the individual device or newly connect production module, process control systems, manufacturing execution systems and enterprise resource planning systems may need reconfiguration. Plant operators may need to be informed of the new device and pre-configured control logic may be activated or even generated.

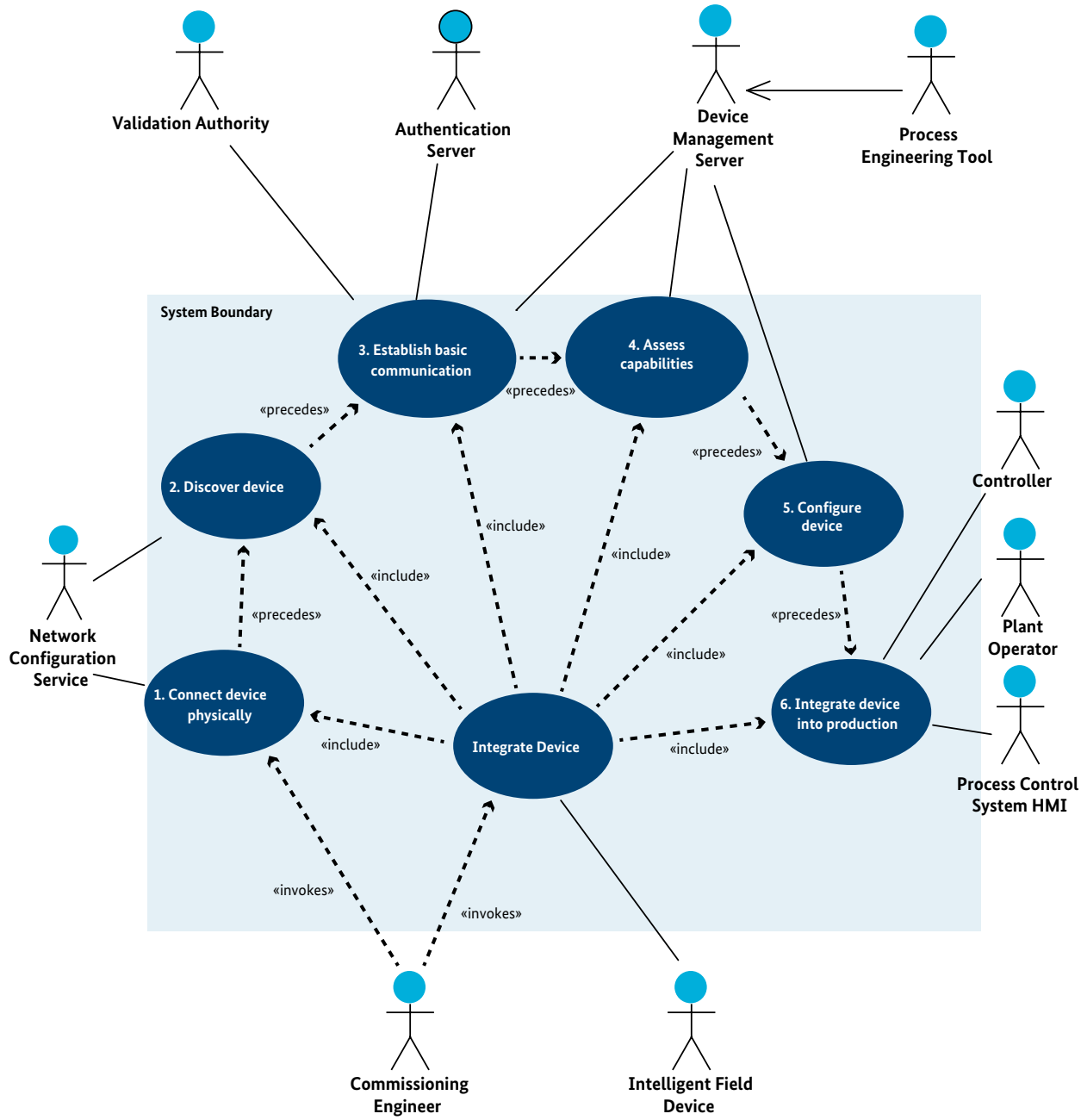
3.1.5 Key Performance indicators

Key Performance Indicators			
ID	**Name**	**Description**	**Reference to mentioned use case objectives**
SR	Setup Rate	Ratio between setting up a system and utilizing the system (ISO 22400-2)	Reduce commissioning times
AV	Availability	Ratio between downtime of a system and utilizing the system (ISO 22400-2)	Reduce commissioning times

1 The phases are derived from the 5-step model proposed by Reinhart et al. in 2010 [17]. A sixth phase “Integration” was added to reflect the need to not only configure the particular field device, but also the surrounding devices in the production process, such as controllers and plant operator consoles.

3.2 Diagrams of Use Case

Figure 3: uc Plug and Produce for Field Devices (Use Case View)



3.3 Technical Details

3.3.1 Actors

Actors			
Grouping		**Group Description**	
Actor Name *see Actor List*	**Actor Type** *see Actor List*	**Actor Description** *see Actor List*	**Further information specific to this Use Case**
Device Management Server	Application	This application detects newly connected devices, assesses their capabilities, configures the devices, and sets them into an operational mode. Example: FDI Server.	IEC 62769-3 (FDI), IEC 62541 (OPC UA), IEC 62453 (FDT)
Intelligent Field Device	Device	A field device can be an actuator or sensor. It is equipped with standardized communication capabilities and a self-description.	IEC 62541 (OPC UA), eCl@ss, NAMUR NE 131
Controller	Device	A controller can execute control algorithms based on sensor inputs and passes actions to actuators. It is equipped with standardized communication capabilities and a self-description.	IEC 62541 (OPC UA)
Network Configuration Service	Service	The network configuration service provides IP addresses, network masks, gateway information and naming services for its clients.	RFC 2131 (DHCP), IEC 62541-12 (OPC UA Discovery)
Validation Authority	Service	Verifies the validity of a digital certificate.	X.509, RFC 5280 (Public Key Infrastructure)
Authentication Service	Service	A central database that can verify user tokens provided by clients. It may also tell servers what access rights the user has. The authentication service depends on the user identity token. It could be a certificate authority, a Kerberos ticket granting service, a WS-Trust Server or a proprietary database of some sort.	IEC 62541-4 (OPC UA)
Process Engineering Tool	Application	The application provides configuration data for a plant, which may include specific configuration parameters for field devices.	IEC 62424 (AutomationML), NAMUR NE 150, Whitepaper “AutomationML and eCl@ss Integration”, VDMA-Einheitsblatt 66415
Process Control System HMI	System	Hardware/Software system that visualizes operator screens to supervise an industrial process.	
Commissioning Engineer	Human	Responsible for the installation and commissioning of an automation system.	
Plant Operator	Human	Supervises the operation of an industrial plant via the Process Control System HMI.	

3.3.2 Further Information to the Use Case for Classification / Mapping

Key Performance Indicators
Relation to Other Use Cases
To be determined.
Level of Depth
Detailed
Prioritisation
High
Generic, Regional or National Relation
Generic
Viewpoint
System use case
Further Keywords for Classification
Plug-and-produce, Plug and work, Plug and play, auto-configuration, device discovery, seamless integration

3.4 Step by Step Analysis of Use Case

3.4.1 Overview of Scenarios

Scenario Conditions					
No.	**Scenario Name**	**Primary Actor**	**Triggering Event**	**Pre-Condition**	**Post-Condition**
1	Connect device physically	Commissioning Engineer	Device replacement request	Device ready to be plugged-in and powered on. Process is set into a state that allows adding the device (e.g., it is halted).	Device plugged in into network and power supply, powered on.
2	Discover device	Network Configuration Service	Device requests IP address	Device connected physically	Device has IP address and is advertised within the network
3	Establish basic communication	Device Management Server	Notification of newly connected device	Device Management Server scans for new devices, Validation Authority, and Authentication Server are set up	Device Management Server is connected to the device, properly authenticated and can issues service requests
4	Assess capabilities	Device Management Server	Connection established	Device carries a self-description and or default configuration	Device Management Server knows how to use and configure the device
5	Configure device	Device Management Server	Device capabilities assessed	Device ready to be re-configured	Device configured for the desired purpose, ready to be integrated into process

3.4.1 Overview of Scenarios (continued)

Scenario Conditions					
No.	**Scenario Name**	**Primary Actor**	**Triggering Event**	**Pre-Condition**	**Post-Condition**
6	Integrate device into production	Process Control System	Notification about newly configured and ready-to-operate device	Device configured	Device actively supports the production process
7	Identity Validation failed	Device Management Server	Notification of newly connected device	Device Management Server scans for new devices, Validation Authority, and Authentication Server are set up	Device Management Server discards the device
8	Device Replacement	Device Management Server	Device capabilities assessed	Device ready to be re-configured	Device configured based on imported configuration from replacement device. Ready to be integrated into process.

3.4.2 Steps – Scenarios

Scenario								
Scenario Name		**No. 1 – Connect device physically**						
Step No.	**Event**	**Name of Process/ Activity**	**Description of Process/ Activity**	**Service**	**Information Producer (Actor)**	**Information Receiver (Actor)**	**Information Exchanged (IDs)**	**Requirements, R-ID**
1	Device mounting request	Prepare device for connection	Bring the device to the location it shall be plugged in, unpack it, plug-in network cable into the device, plug-in power cable into power supply	EXE-CUTE	CE[2]	IFD		
2	Device prepared	Plug-in Device	Connect network cable into network port of the network equipment or bring the device in proximity of the wireless connection point.	EXE-CUTE	CE	IFD		
3	Device plugged-in	Turn on device	Switch power button on	EXE-CUTE	CE	IFD		

2 These abbreviations refer to the actors listed in Section 4.3.1 (e.g., CE = Commissioning Engineer)

3.4.2 Steps – Scenarios (continued)

Scenario								
Scenario Name		**No. 2 – Discover device**						
Step No.	**Event**	**Name of Process/ Activity**	**Description of Process/ Activity**	**Service**	**Information Producer (Actor)**	**Information Receiver (Actor)**	**Information Exchanged (IDs)**	**Requirements, R-ID**
1	Device switched on	Recognize newly connected device	Detect device connection on lower network layer.	GET	IFD	NCS		
2	Device recognized	Assign network address	Determine a free network address, assign it to the device, send it to the device	CREATE	NCS	IFD		
3	Device is addressable via the network	Notify Device Management	Announce the device to the Device Management Server to start configuring it	GET	NCS	DMS		

Scenario								
Scenario Name		**No. 3 – Establish basic communication**						
Step No.	**Event**	**Name of Process/ Activity**	**Description of Process/ Activity**	**Service**	**Information Producer (Actor)**	**Information Receiver (Actor)**	**Information Exchanged (IDs)**	**Requirements, R-ID**
1	Notification of device availability	Connect to the device	Contact the device and create a new session context on the device for the following interaction	EXE-CUTE	IFD	DMS		
2	Connected to device	Get device certificate	Retrieve device certificate from the device	GET	IFD	DMS		
3	Device certificate available	Validate certificate	Contact Validation Authority and validate the certificate to ensure valid identity of the device	GET	VA	DMS		
4	Device certificate validated	Authorize connection	Contact authentication service to authorize the device management server for reading and writing device parameters	GET	AS	IFD		

3.4.2 Steps – Scenarios (continued)

Scenario								
Scenario Name		**No. 4 – Assess capabilities**						
Step No.	**Event**	**Name of Process/ Activity**	**Description of Process/ Activity**	**Service**	**Information Producer (Actor)**	**Information Receiver (Actor)**	**Information Exchanged (IDs)**	**Requirements, R-ID**
1	Connection established	Assess device type	Read device type, determine if usable in current configuration	GET	IFD	DMS		
2	Device type identified	Read device properties	Download default device properties and default configuration parameters from the device	GET	IFD	DMS		
3	Device properties retrieved	Assess device capabilities	Match device properties with internal plant configuration, identify matching configuration parameters for the device type	EXE-CUTE	PET	DMS		

Scenario								
Scenario Name		**No. 5 – Configure Device**						
Step No.	**Event**	**Name of Process/ Activity**	**Description of Process/ Activity**	**Service**	**Information Producer (Actor)**	**Information Receiver (Actor)**	**Information Exchanged (IDs)**	**Requirements, R-ID**
1	Device capabilities determined	Prepare device for configuration	Lock the device for writing parameters	EXE-CUTE	DMS	IFD		
2	Device locked	Download configuration parameters	Write configuration parameter to the device, overwrite default configuration	GET	DMS	IFD		
3	Configuration written	Calibrate device	Start calibration routine of the device, wait for completion	EXE-CUTE	DMS	IFD		
4	Device calibrated	Activate device	Set device into operation mode	EXE-CUTE	DMS	IFD		

3.4.2 Steps – Scenarios (continued)

Scenario								
Scenario Name		**No. 6 – Integrate device into production**						
Step No.	**Event**	**Name of Process/ Activity**	**Description of Process/ Activity**	**Service**	**Information Producer (Actor)**	**Information Receiver (Actor)**	**Information Exchanged (IDs)**	**Requirements, R-ID**
1	Device operational	Notify relevant system of device availability	Send a notification to systems that have registered an interest in the particular device	GET	DMS	PCS, CTR, PO		
2	Device availability notification	Subscribe to information from the device	Read or set up a subscription for regular information updates from the device.	REPORT	IFD	PCS, CTR, PO		
3	Device information read	Integrate device information	Use information read from the device in the overall production process, e.g., use the sensor values as input for specific algorithms	EXE-CUTE	PCS, CTR, PO	PCS, CTR, PO		

Scenario								
Scenario Name		**No. 7 – Identity validation failed**						
Step No.	**Event**	**Name of Process/ Activity**	**Description of Process/ Activity**	**Service**	**Information Producer (Actor)**	**Information Receiver (Actor)**	**Information Exchanged (IDs)**	**Requirements, R-ID**
1	Notification of device availability	Connect to the device	Contact the device and create a new session context on the device for the following interaction	EXE-CUTE	IFD	DMS		
2	Connected to device	Get device certificate	Retrieve device certificate from the device	GET	IFD	DMS		
3	Device certificate available	Validate certificate	Contact Validation Authority and validate the certificate to ensure valid identity of the device	GET	VA	DMS		
4	Device identity validation failed	Disconnect and discard the device	Disconnect from the device, mark it as not reliable, notify other interested systems	EXE-CUTE	DMS	IFD, PCS, CTR, PO		

3.4.2 Steps – Scenarios (continued)

Scenario								
Scenario Name		**No. 8 – Device replacement**						
Step No.	**Event**	**Name of Process/ Activity**	**Description of Process/ Activity**	**Service**	**Information Producer (Actor)**	**Information Receiver (Actor)**	**Information Exchanged (IDs)**	**Requirements, R-ID**
1	Device capabilities determined	Import configuration	Retrieve the configuration of the device to be replaced.	GET	DMS	IFD-old		
1	Device configuration retrieved	Prepare device for configuration	Lock the device for writing parameters	EXE-CUTE	DMS	IFD		
2	Device locked	Download configuration parameters	Write configuration parameter to the device, overwrite default configuration	GET	DMS	IFD		
3	Configuration written	Calibrate device	Start calibration routine of the device, wait for completion	EXE-CUTE	DMS	IFD		
4	Device calibrated	Activate device	Set device into operation mode	EXE-CUTE	DMS	IFD		

3.5 Requirements

The following requirements have been derived from the former use case specification. Appendix C provides a longer list of requirements derived from the description of the application scenario “Adaptable Factory”, which may span also other use case specifications.

Requirements (optional)		
Categories for Requirements	**Categories Name**	**Category Description**
3.2	SemUnd	Semantic Understanding
Requirement ID	**Requirements Name**	**Requirement Description**
SemUnd-1	Self-describing Module	Each device shall include a self-description (e.g., properties, default configuration parameters) that allows fast and robust re-configuration of a production line.
SemUnd-2	Modular Engineering	The system[3] shall allow engineers to execute a modular engineering (i.e., allowing independent work on parts of the engineering data), where libraries of reused modules are employed.

3 „The system“ refers to an automation system supporting “plug and produce” for field devices. This may for example be a system comprising of all the actors listed in Section 4.3.1. It is left intentionally unspecified which actors shall realize the requirement, so that different technical implementation are possible.

Categories for Requirements	Categories Name	Category Description
4.1	NetInt	Network Interoperability
Requirement ID	**Requirements Name**	**Requirement Description**
NetInt-1	Interoperable devices	The system shall be composed out of devices that are interoperable.

Categories for Requirements	Categories Name	Category Description
5.1	BasConn	Basic Connectivity
Requirement ID	**Requirements Name**	**Requirement Description**
BasConn-1	Automatic Network Connectivity	Each newly connected field device shall receive network connectivity without human interaction.

Categories for Requirements	Categories Name	Category Description
6.7	QoS	Quality-of-Service
Requirement ID	**Requirements Name**	**Requirement Description**
QoS-1	Fast Configuration	The system shall make field devices operational in less than a minute[4].

Categories for Requirements	Categories Name	Category Description
6.8	DisConf	Discovery and Configuration
Requirement ID	**Requirements Name**	**Requirement Description**
DisConf-1	Autonomous constraint detection	The system shall enable field devices to determine constraints to their production procedures.
DisConf-2	Device Connection Notification	Each newly connected field device shall be advertised to all interested system parts.
DisConf-3	Compute require modification	The system shall detect necessary control-required (e.g., activating a control loop) and software-required (e.g., updating a database) modifications upon connecting a new device.
DisConf-4	Auto-update system for new device	The system shall propagate the necessary control-required and software-required modifications upon connecting a new field device to all relevant system parts.

Categories for Requirements	Categories Name	Category Description
7.5	ConnHMI	Connections and HMI
Requirement ID	**Requirements**	**Requirement Description**
ConnHMI-1	Auto-visualization	The system shall allow automatic creation of visualizations for device parameters.

4 The timing required here is based on the fact that the device undergoes a process of network discovery, capability assessment, configuration, and integration into a larger system. The initial network discovery may be performed much faster, while the integration into a system may be much longer, for example if it requires human approval. The “one-minute” requirement is stated here as an engineering goal simply to make it more tangible. Such a time would be desirable from a user perspective, but the context of the field device may prevent the realization of such a requirement. The requirement shall contribute to improving the KPI “setup rate”.

The following table lists several standards potentially relevant for the implementation of a requirement:

Requirement ID	Requirements Name	Candidate Standard
SemUnd-1	Self-describing modules	eCl@ss, IEC 62541-100, FDI, FDT, NAMUR MTP
SemUnd-2	Modular engineering	IEC 62714 (AutomationML), FDT, NAMUR MTP
NetInt-1	Interoperable modules	IEC 62541 (OPC UA), IEC 62769 (FDI), IEC 62453 (FDT), NAMUR MTP
BasConn-1	Automatic netwock connectivity	RFC 3927, IEC 62541-12
QoS-1	Fast configuration	
DisConf-1	Autonomous constraints detection	
DisConf-2	Module connection notification	IEC 62541-12
DisConf-3	Compute required modifications	
DisConf-4	Auto-update system for new device	IEC 62541, IEC 62769 (FDI), IEC 62453 (FDT)
ConnHmi-1	Auto-visualization	IEC 62769 (FDI), IEC 62453 (FDT), NAMUR MTP

3.6 Common Terms and Definitions

Common Terms and Definitions	
Term	**Definition**
AS	Authentication Service
AV	Availability
CE	Commissioning Engineer
CTR	Controller (e.g., PLC or industry PC)
DMS	Device Management Server
FDI	Field Device Integration
FDT	Field Device Tool
HMI	Human Machine Interface
IFD	Intelligent Field Device
MTP	Module Type Package
NCS	Network configuration service
OPC UA	OPC Unified Architecture
PCS	Process Control System
PET	Process engineering tool
PnPFD	Plug and Produce for Field Devices
PO	Plant Operator
SR	Setup Rate
VA	Validation Authority

In context of Industrie 4.0, first the "Recommendations for implementing for implementing the strategic initiative Industrie 4.0" [2] from April 2013 described a plug-and-produce scenario. In 2016 the scenario was slightly refined as the scenario "adaptable factory" by Working Group 2 of the Plattform Industrie 4.0 [1]. Influenced by this scenario description, Siemens, SAP, Telekom, and Festo created a demonstrator for a modular manufacturing line with a flexible transportation system that allowed re-configurations (more details in [3]). This demonstrator used proprietary technologies, e.g., it included vendor-specific device descriptions. Standard-based module descriptions and communication technologies could enable Plug-and-Produce capabilities across vendor borders. In October 2016, Working Group 2 of the Plattform Industrie 4.0 released a revised version of the application scenario document [6], which also included mappings to the IIC testbeds, the ZVEI use cases for Industrie 4.0, and the Smart Service Welt.

The Plattform Industrie 4.0 has defined the architectural viewpoint model RAMI 4.0 [4] and postulated the concept of so-called asset administration shells (AAS), which are standardized interfaces to Industrie 4.0 component data and services. The structure of these AAS is still under discussion [5] [7], but an orientation towards existing standards is likely. In this context, the GMA Fachausschuss 7.21 is creating a glossary to define important terms of Industrie 4.0 [8]. Some of these terms refer to elements of an AAS. The same working group also defined elements of a related service architecture, which discusses several concepts for AASs [9].

There are additional ongoing industry initiatives, which work on concepts relevant for the application scenario "Adaptable Factories". The FieldComm group is defining the FDI standard [10], which tries to reduce the efforts for integrating field devices and provides device descriptions useful for a plug-and-produce scenario for field devices. The FDT group [11] is working on the FDT IIoT Server (FITS) concept which intends to simplify the device management for the whole lifecycle. Several NAMUR groups are working on the Module Type Package specification to describe package units, i.e. process modules [12]. This could be considered as a kind of AAS for process modules, which also relies on industry standards and allows cross-vendor interoperability. The DFKI Smart Factory in Kaiserslautern built an Industrie 4.0 demonstrator for Hannover Fair 2014, which involved process modules from different vendors and relied on a universal plug-in connector [14].

There are also academics works tackling plug-and-produce challenges for field devices. Krüning and Epple [15] described an exploration agent for PROFINET IO devices. In a similar manner, Dürkop et al. [16] proposed an auto-configuration service for PROFINET IO devices that uses DHCP and OPC UA Discovery to access GSD device descriptions. Reinhart et al. [17] used a configuration manager component to automatically configure Ethernet Powerlink devices and later [18] proposed a unified PnP architecture for automatic integration of field devices. Jasperneite et al. [19] discussed two different kinds of plug-and-produce demonstrators in the Smart Factory OWL.

5. Metamodel for Virtual Entities

To support the automatic integration of field devices or process modules, devices need to be equipped with device descriptions and capabilities for downloading configurations and activating them. This section describes an exemplary meta-model for capturing such information. The model is not specific for a particular use case, but has a generic structure oriented toward existing standards. It may not cover all Industrie 4.0 application scenarios, and is simply used in the scope of this paper to make the use case of auto-integrating field device more tangible.

5.1 Static Views: Metamodel

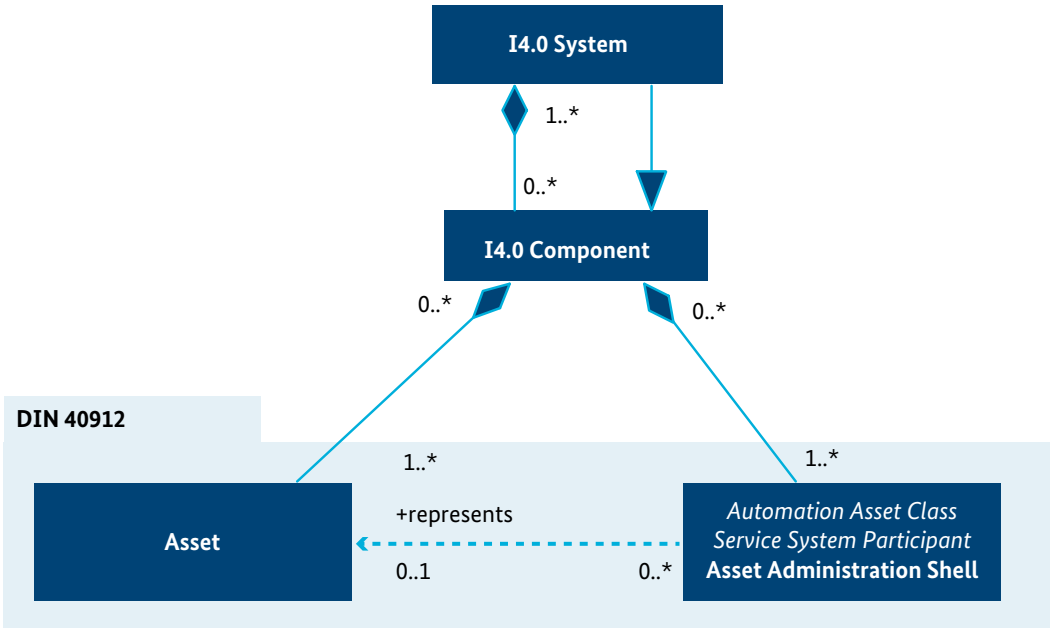
An I4.0 System consists of a set of interacting I4.0 Components (Figure 4), as defined in the Industrie 4.0 implementation strategy [4]. Each I4.0 component in turn consists of 0 to many Assets (e.g., field device, robot, ERP) and 0 to many Asset Administration Shells (AAS). Thus, the I4.0 components as such is a logical construct, which has no separate physical manifestation besides the asset and the AAS. It does not induce a specific deployment to computing resources, as detailed in Section 6.5. The AAS represents the asset and provides data and services for the asset. An I4.0 System is again an I4.0 component, thus a hierarchical structure can be built.

Figure 5 shows the top-level structure of the AAS, which is detailed in subsequent figures. The structure is not tied specifically to a device integration scenario, but could be used in many the Industrie 4.0 application scenarios. The following paragraphs will describe the idea of each element, but later focus on such elements needed to implement the use case “Plug and Produce for Field Devices”.

Root node of the model is the **Asset Administration Shell** (AAS), as postulated in DIN 91345. In the IoTA Reference model the corresponding model element is called *Virtual Entity*, which better aligns with the terminology of cyber-physical systems that consist of a physical and virtual entity. The AAS contains a Header and a Body.

Figure 4: I4.0 System composed out of I4.0 Components

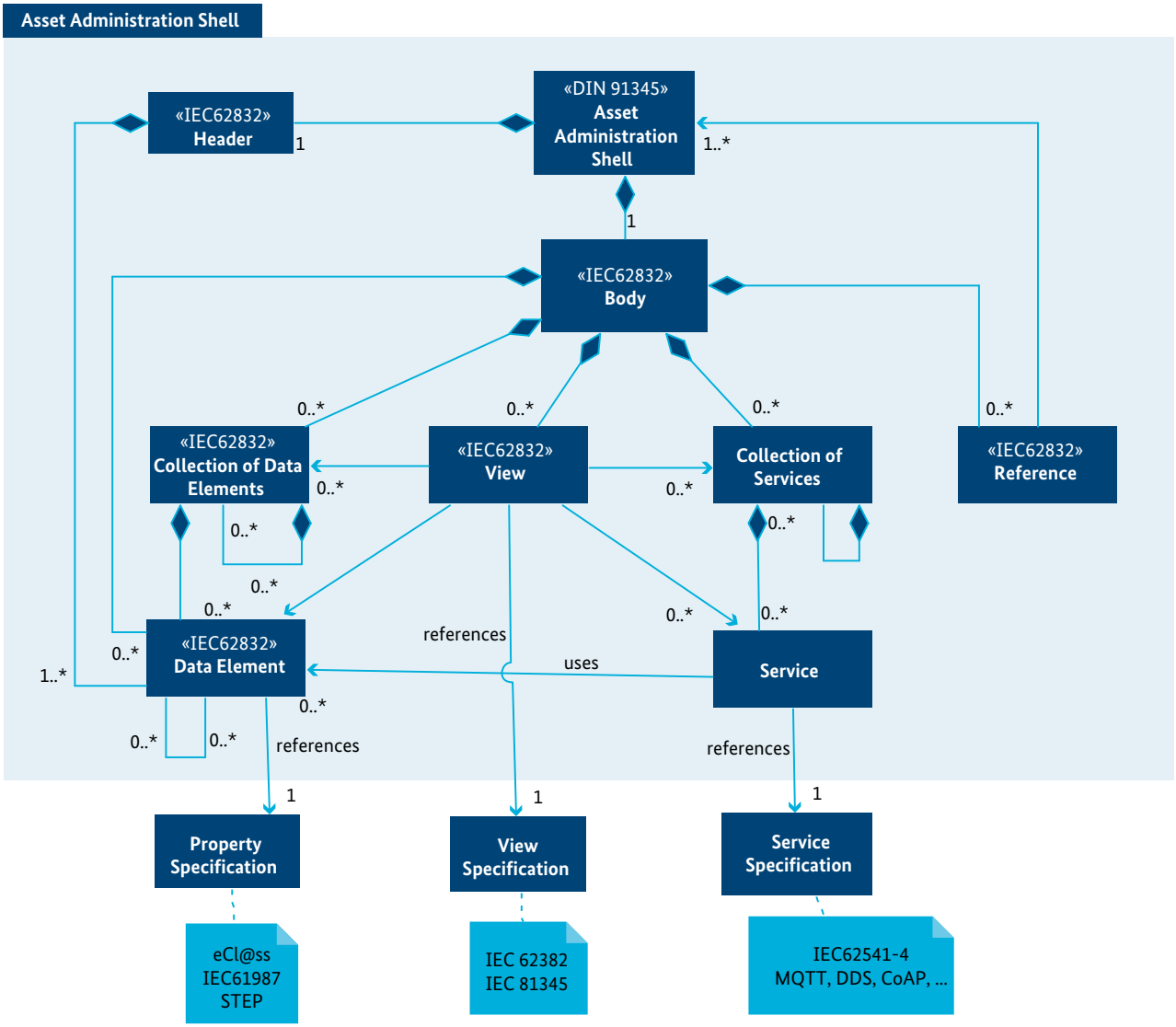
class I4.0 Component and System



Header: the *Header* contains minimal information to identify both the AAS itself and its managed assets. It has also been postulated by the ZVEI SG "Security" that the *Header* should contain classification of the required level of security and a reference to the AAS body, where more security-related properties (e.g., key sizes, authentication, etc.) can be retrieved if needed. The *Header* is mandatory and must be specified by any AAS. Using the identifier, clients can retrieve both type and instance information about the managed assets. For example, the identifier could be looked up in public asset type repositories, e.g., eCl@ss or IEC CCD. Information about the asset can be stored either on the device itself and be retrieved directly from the AAS or on some other storage unit or repository, requiring an additional connection.

Figure 5: Asset Administration Shell Structure

class AdminShell



The concept of a *Header* originates from IEC 62832 (Digital Factory), where it shall contain information for the plant owner that "allows differentiating the automation asset instances from each other in the Digital Factory through their lifecycle". IEC 62832 defines specific data elements for the *Header*, namely dataSpecification, classCode, preferredName, structureElement Classification, designAuthority, and timeDate. These attributes are specific for the use case of the digital factory and may not be useful in any I40 sceneario. Therefore, detailed specification of the *Header's* data elements has been omitted in this document and left for future work. Examples for AAS Headers are included in the ZVEI whitepaper "Beispiele zur Verwaltungsschale der Industrie 4.0-Komponente - Basisteil".

Body: the *Body* is a container for properties, supported views, services, and references. It can be considered the main source of information of the AAS and contains all information and functions to perform different applications with the managed assets. The concept of a *Body* also originates from IEC 62832 (Digital Factory). There, the *Body* is merely a collection of data elements and does not contain views, services, or references. Data suppliers provide the information contained in the body. In case of I40 components, the *Body* shall not only be a passive container for properties, but also a provider of higher-level services allowing the invocation of certain functionalities.

Collection of Data Elements: this class groups a set of related data elements. It is supposed to ease a human exploration of the data elements by providing a pre-defined and possibly standardized structure. The *Collection of Data Elements* resembles the folders known from file systems of general purpose operating systems and are sometimes referred to as "partial models". Grouping may not be needed for pure machine-to-machine communication, which only requires standardized properties and their semantics. The concept of grouping data elements exists in various standards, such as eCl@ss, STEP, IEC61897, and IEC62832 (Digital Factory).

Data Element: a *Data Element* is a key/value pair modeling a static property or dynamically changing process value of the managed assets. The key references a globally unique identifier that allows looking up the data elements semantics, while the value resembles the concrete characteristic of the data element for the managed asset. *Data Elements* could for example refer to product features (e.g., range of a temperature sensor), states of the managed assets (e.g., running, paused, stopped), process values (e.g., concrete valve position), asset health information (e.g., maintenance required), or any other information relevant for a given use case.

The concept of *Data Elements* is left as generic as possible here to not constrain the model towards a certain use case. Attributes of Data Elements may be based on the standard attributes out of IEC61360, of which eCl@ss uses a subset. It is also conceivable to have different specializations of Data Elements in the model, e.g., product features, process values, or asset states. In line with IEC 68832 (Digital Factory), there may also be proprietary, i.e., vendor-specific, non-standardized *Data Elements*, to realize some differentiating features or functions, but these would typically not be exposed via the AAS but through other means. In the current model, Data Elements are only "assurances" and do not model for example requirements for properties as in other models. Multiple catalogs of standardized *Data Elements* are available, e.g., IEC61897, eCl@ss, STEP.

View: this class provides means to filter the contents of the AAS, so that only information relevant for a particular user role is shown. The number of *Data Elements* and *Services* for a typical industrial asset may be in the range of hundreds or even thousands, so that Views allow humans or tools to restrict the complexity when exploring or interacting with the AAS. For example, there could be Views for maintenance personal or Views that focus on the location-related *Data Elements* and *Services*. Different *Views* may reference the same *Data Elements* and *Services*, there is no strict partitioning between them. For example, the Data Element "height" may occur both in a view focusing on the physical dimensions of the asset and in a view focusing on the product features.

The concept of *Views* also exists in IEC61832 (Digital Factory). There, it is not part of the *Body*, but rather Views are managed centrally and stored in the Digital Factory repository with references to the *Data Elements* of assets. In case of I40-components, centrally managed Views may make AAS not self-contained, which would complicate replacement. It would also prevent the asset vendor to specify own views specifically for the AAS.

Collection of Services: an AAS may contain a set of *Collection of Services*, which can be called by users of the AAS and execute some functionality of or on the managed asset. Analogous to the *Collection of Data Elements*, this class

allows grouping related services and makes exploring the AAS more convenient, but is not necessarily needed for machine-to-machine communication. This concept does not exist in IEC 62832 (Digital Factory) or other standards, but is proposed in this report.

Service: *Services* provide interfaces to executable functionality of the AAS. Usually, they would refer to some higher-level, often asset-specific functionality (e.g., "close valve", "calibrate", "drill hole"). This may include administrative services to retrieve historical data or alarm conditions. To allow interactions with *Services* there input and output parameters need to be defined, their exceptional behavior needs to be specified, and the interaction paradigm needs to be made clear. Often, services interact with user in a request/response scheme, where the users issues the request and waits to get a response back. It is also possible to have services manipulating an internal state of the AAS, e.g., authorizing a user in a session.

Service signatures (i.e., name, input, output, exceptions) could be specified according to a CORBA syntax. The concrete model is still under discussion. The GMA 7.21 has proposed a number of application-agnostic basic services, which could be offered by any component (Figure 6). No standards for application-specific services or respective service catalogs are known. Interactions with industrial assets are today often based on vendor-specific means. IEC 62541-4 (OPC UA Services) defines around thirty basic services (e.g., read/write, subscribe, etc.) for OPC UA servers and provide a template for generic AAS services.

Figure 6: Sets of Services defined by GMA 7.21

class Services

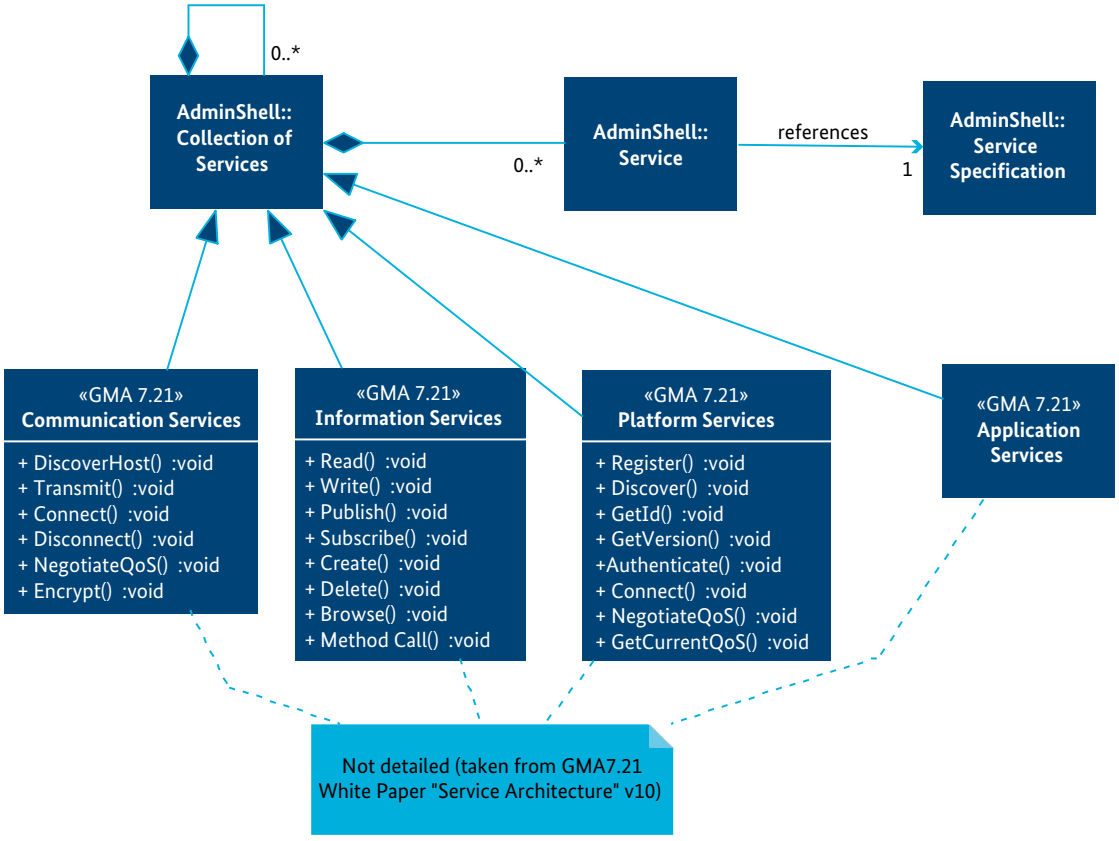
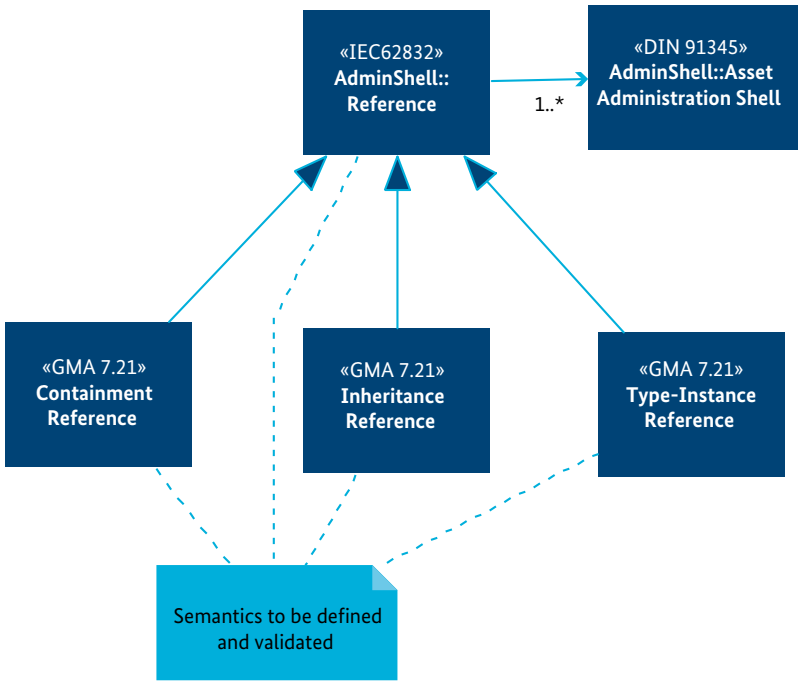


Figure 7: Different Types of References by GMA 7.21

class References



References: AAS may reference other AASs for certain use cases. It may be useful to know to which other assets the managed asset is connected, in which plant segment it is located, and from which other asset it is derived. These reference may change dynamically while operating the asset, as the plant layout may change. Figure 7 shows specializations for *Reference* proposed by GMA 7.21, which are for example also directly supported by OPC UA.

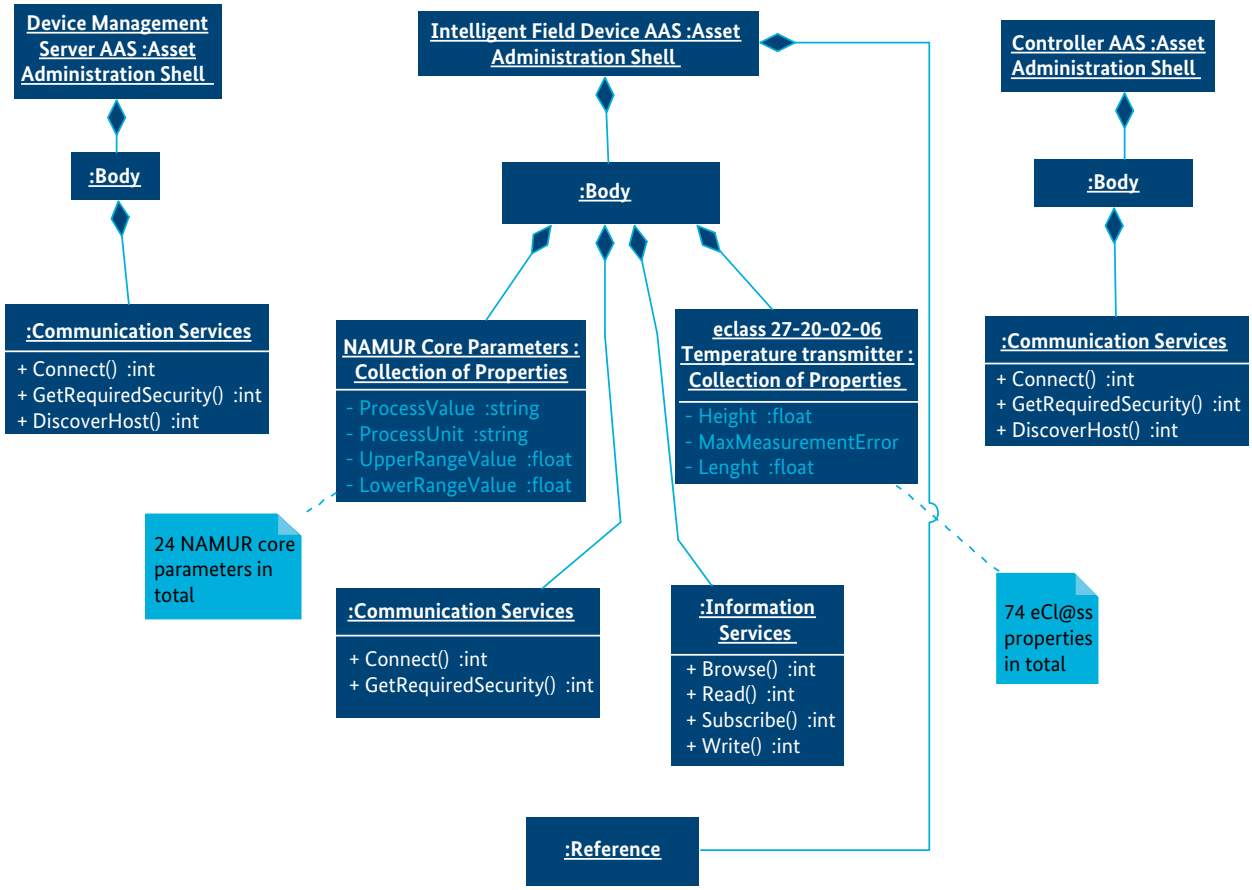
The meta-model described so far should be considered as a minimal set of concepts needed to enable most I40 application scenarios. The modeling itself aimed to be conservative and only introduce concepts needed for standardization and vendor-neutral communication. Implementation details are not tackled in this conceptual model. Technology mappings in subsequent subsections show how this conceptual model can be realized. The meta-model also aimed at re-using concepts from existing standards (mostly IEC 62832, IEC61360, and IEC 62541) as much as possible in order to keep the additionally needed standardization as small as possible. Additional concepts should be introduced into the model only if certain application scenarios require them. It is still unclear whether a generic AAS model for all use cases is achievable and practical. Application-specific variants streamlined for certain resource constraints and communication needs may need to be investigated.

5.2 Static View: Example for Device Integration

Figure 8 shows an example instance of the meta-model for the device integration scenario sketched in Section 4. The instance shown in this subsection is still technology-agnostic and can be implemented with different concrete technologies. The example includes three AASs, for a Device Management Server (e.g., an FDI server), an intelligent field device (e.g., a temperature transmitter), and an automation controller. For the FDI server and the controller, only communication services are shown here. The temperature transmitter has a *Body* that contains two *Collection of Data Elements*: the NAMUR Core Parameters according to NAMUR Recommendation NE131 and the property list for eCl@ss 27-20-02-06 for temperature transmitters. The NE131 list has 24 data elements in total, which are not all modelled in the figure for brevity. The eCl@ss list for the temperature transmitter has 74 data elements in total. The temperature transmitter AAS *Body* also provide information services to read and write its data elements.

Figure 8: Asset Administration Shell Structure: Example Instance

object PnP Asset Admin Shells



5.3 Dynamic Views

AAS Services as depicted in Figure 6 can be involved in different interactions. For illustration purposes, the following view provides a typical examples. The AAS in the example interact according to a client/server interaction pattern.

Figure 9 shows the basic interaction between two AASs. AAS1 establishes a connection to AAS2 and then browses the stored data elements and services. It then registers interest for a particular data element by setting up a subscription. Afterwards AA2 sends updated value for the data element to AA1, either periodically or upon changes. After AA1 has finished its work, it ends the subscription to AAS2 and disconnects.

Using the specified default services, different types of interactions can be realized. More information can be found in [7].

5.4 Dynamic Views: Example for Device Integration

Figure 10 provides some details on how the different AASs would interact during runtime when integrating a new intelligent field device. Besides Device Management Server, Intelligent Field Device, and Controller, also a Process Engineering Tool and a Process Control System participate in the interaction. Each of these entities provides an AAS. First the Device Management Server imports configuration data from the Process Engineering Tool, so that it is able to match device configuration from that tool with actual devices connected for example during commissioning or run time. Then it continuously scans the network for newly connected devices. Once a commissioning engineer plugs the intelligent field device into the network, it gets an IP address and then announces its presence on the network with a broadcast.

Figure 9: Browse and Subscribe between AASs

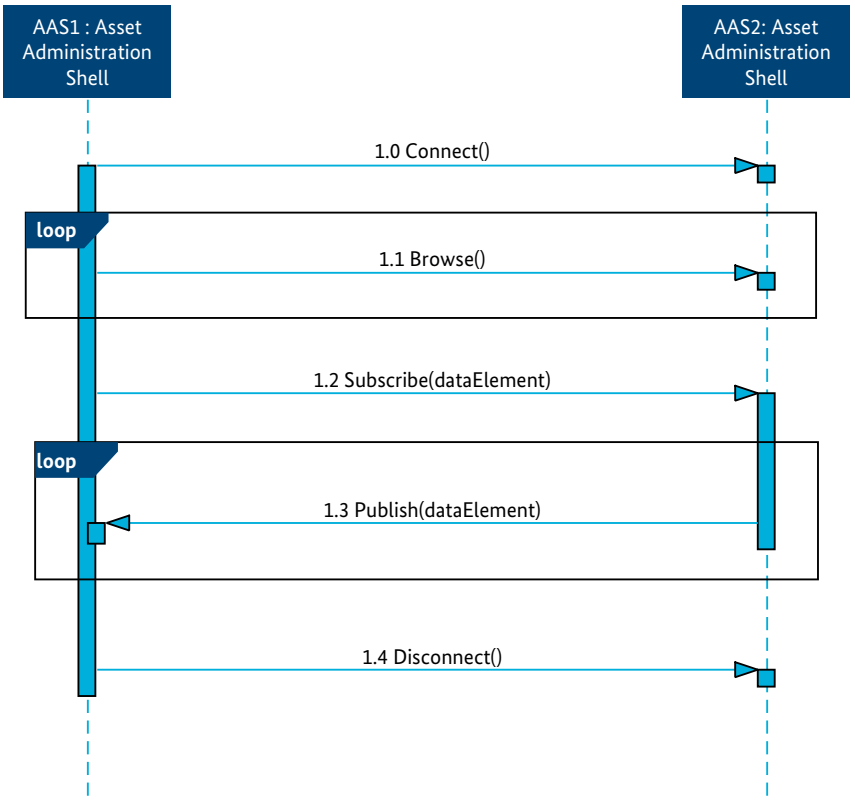
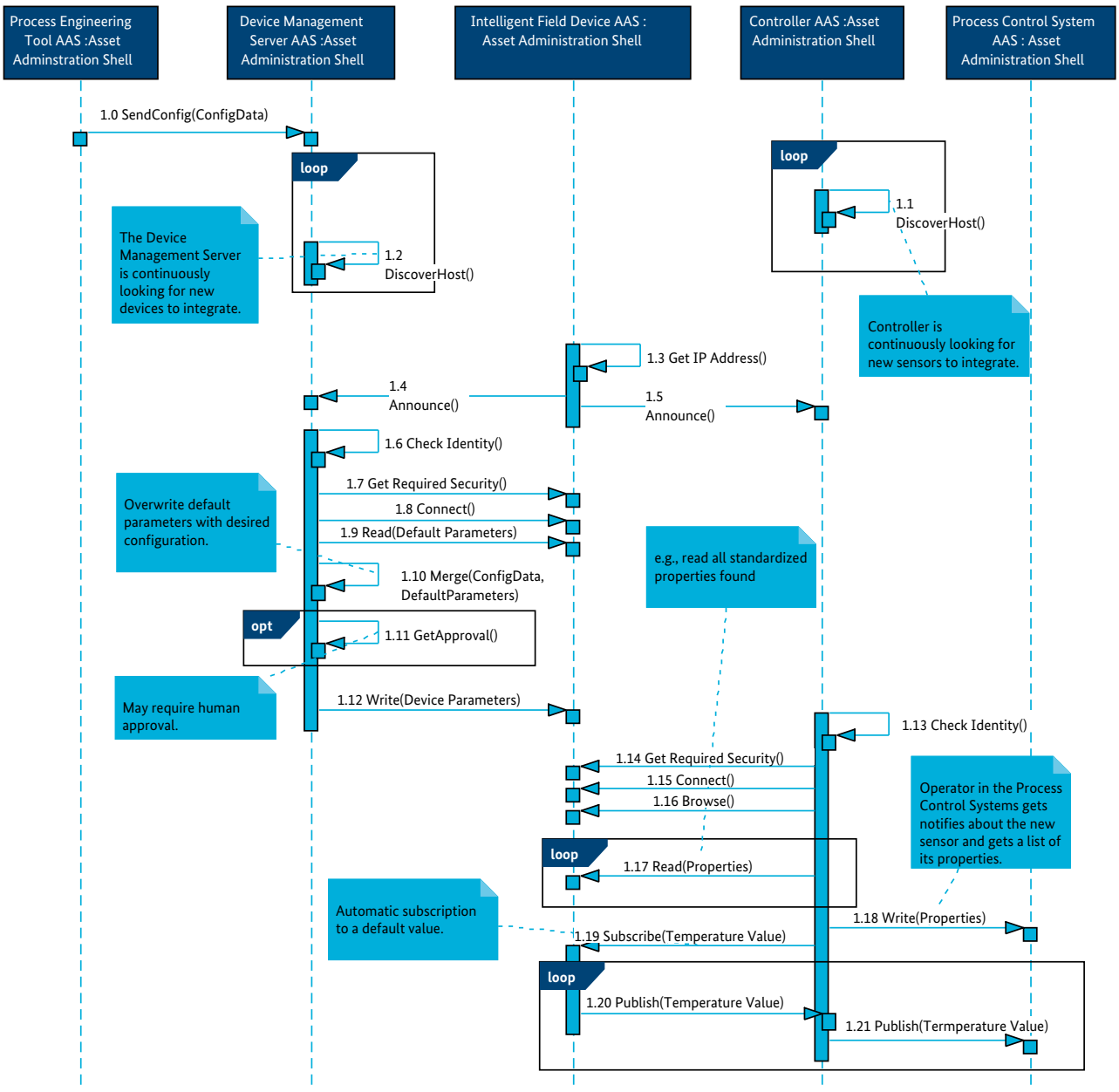


Figure 10: Asset Administration Shells used in an example scenario

sd Plug and Produce for Field Devices (Dynamic View)2



The Device Management Server then first checks the identity of the newly connected device, possibly via a third-party (not detailed here). This provides some assurance that no malicious devices are integrated into the production process. After establishing the basic connection with the device, the Device Management Server reads its default configuration parameters. These parameters are then matched against the configuration imported from the Process Engineering Tool earlier. Both configuration need to be merged and then it may be necessary to get human approval for the configuration before downloading it to the device.

Once the Device Management Server has downloaded the configuration parameter to the Intelligent Field Device it is set operational and can now be used by other assets. In this example, a controller establishes a connection with the Intelligent Field Device, reads all its properties and reports them to a process control system, where they could be displayed to an engineer or human operator. Afterwards it subscribes for the default process value, in this case a temperature value, which is forwarded to the process control system, where it could be integrated into a HMI or a control loop.

Note that the interaction diagram in Figure 10 only provides a specific example of interactions for illustration purposes. An actual implementation would likely require more interactions and differentiate different possible outcomes of the scenario (e.g. successful connection, authorization failed, parameter set not recognized, parameter merging failed, etc.). The example merely serves to explain how the AAS model from Section 6.1 can be instrumental for a device integration use case as described in Section 4.

5.5 Deployment Views

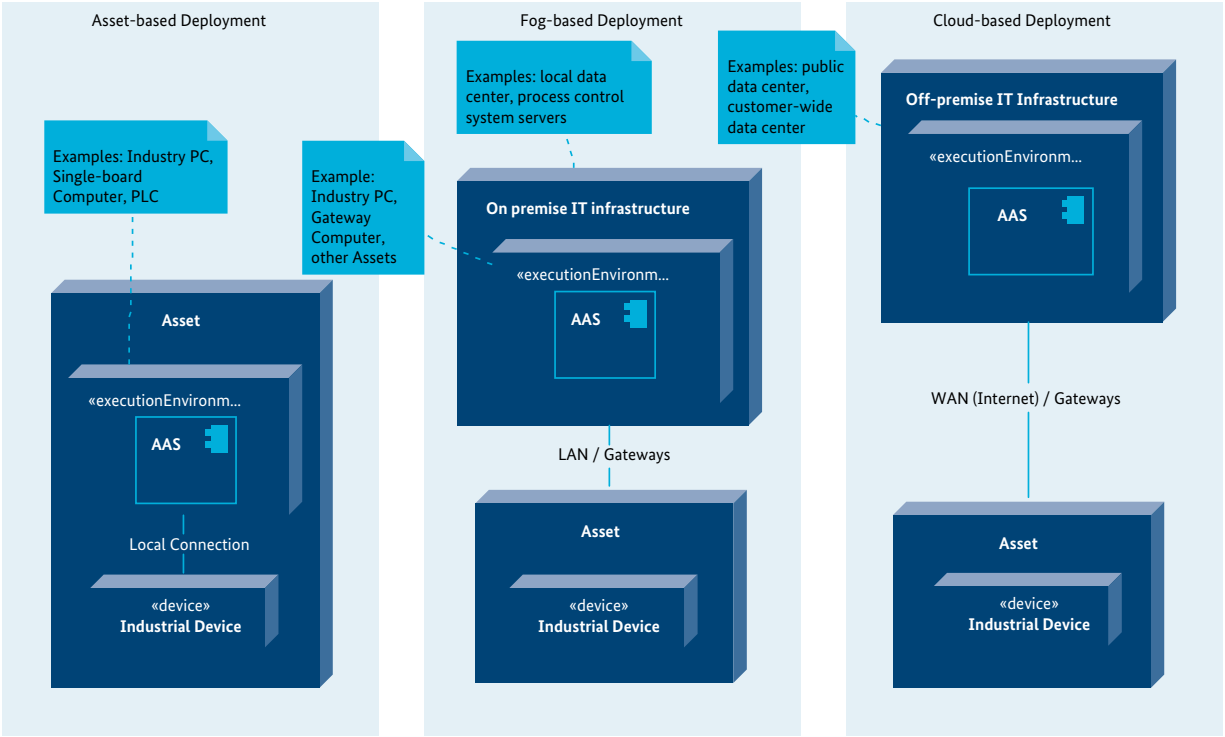
Asset Administration Shells can be deployed to a computing infrastructure in different variants, each having different benefits and drawbacks. The following discusses deployment from four different views, i. e., physical proximity, distribution, virtualization, and lifecycle. More deployment views are conceivable. In general, there is no mandatory specification of a particular deployment. Different use cases favor different deployments.

5.5.1 View "Physical Proximity to the Asset"

This view is characterized by the physical proximity of the asset to the AAS and the type of connection between asset and AAS.

Asset-based Deployment: The AAS is located "within" the industrial asset resulting in a cyber-physical deployment unit. The asset includes some kind of execution environ-

Figure 11: deployment Proximity



ment, such as a PLC or Industry PC that allows hosting an AAS. The execution environment can both be the regular computing node of the industrial device, or an additional auxiliary computing node that is added to the device. The AAS can manipulate the managed industrial device via a local connection, i.e., some kind of internal or external computer bus, e.g., a fieldbus.

Benefits:

- Self-contained asset, information is directly available from the asset. May simplify maintenance tasks.
- Exploits available cheap computing power
- Eases root cause analysis
- Requires fewer network configuration

Drawbacks:

- Requires dedicated IT administration (e.g., security patches, backups)
- Requires dedicated privacy measures (e.g., authorization, authentication)

Fog-based Deployment: The AAS is located in a computing node physically separated from the asset, but residing in the local IT infrastructure of a factory or plant. The computing node may be for example a process control system, an Industrial PC in a local data center, a gateway, a switch with computing capabilities, or even the spare computing capacity of another asset. The AAS is connected to the asset via a LAN, for example using OPC UA client / server transport over TCP/IP.

Benefits:

- Central IT administration (e.g., security patches) and privacy management (e.g., access rights)
- Better utilization of available IT infrastructure (e.g., economies of scale)
- Easier to establish a trust boundary around the local infrastructure, sensitive data can be kept secure

Drawbacks:

- May need expensive local IT infrastructure and local IT administration (compared to cloud deployment)
- More complicated configuration (e.g., additional network connections)
- Lower reliability, as LAN connection may be affected
- Maintenance more complicated since there is no physical proximity between AAS and asset
- More delay for data provision compared to asset-based deployment. Data actuality now depends on LAN/network QoS, e.g. latency.

Cloud-based Deployment: The AAS is located in a computing node physically separated from the asset, residing in customer-wide cloud infrastructure (e.g., BASF data center, ExxonMobile data center), an automation vendor cloud infrastructure (e.g., Siemens Mindsphere, GE Predix), or a public cloud infrastructure (e.g., Amazon Web Services or Microsoft Azure). The connection to the asset is established via the Internet, for example using OPC UA over AMQP.

Benefits

- Potentially cheaper IT operation and administration due to economies of scale
- Eases world-wide remote access after initial configuration
- Potentially high availability, redundancy, performance
- Close proximity to analytic services

Drawbacks

- Customer concerns regarding sensitive information in public computing environment
- Connection between AAS and asset over Internet less reliable, less deterministic, slower

5.5.2 View "Distribution to Multiple Nodes"

This view is characterized by the splitting of AAS data elements and services to different computing nodes.

Centralized AAS: There is only a single AAS for an asset. All data elements and services are at a central location. If data elements shall be added, changed, or removed, always the centralized AAS needs to be queried. Such an AAS would be created in the earliest planning phases of an asset and afterwards be used as a kind of repository for all organizations contributing to the information in the AAS. The AAS would cover planning as well as runtime information. There is a unique entry point (e.g., an IP address) for the AAS.

Benefits:

- Lower complexity for clients, only one communication partner
- Easy to manage, administrate, secure
- No risk for data inconsistencies

Drawbacks:

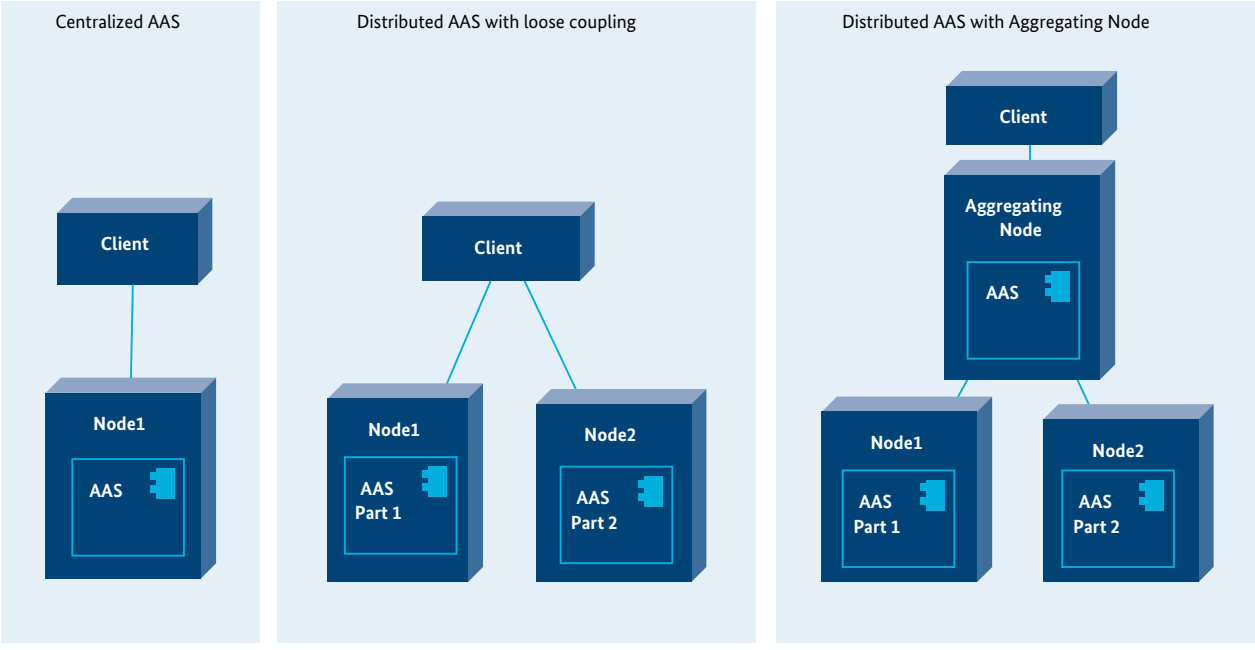
- Does not reflect today's systems well, i.e., many nodes storing information about an asset
- Complicates manipulating the AAS from different organizations (e.g., engineering department, service department) and in different lifecycle phases

Distributed AAS with loose coupling: Different network nodes store information for a single asset under the same asset identifier. There are multiple entry points for the AAS (e.g., IP addresses). Information on the asset in the different nodes may or may not be overlapping. Clients requesting access to the AAS would either query each node individually or use some kind of network functionality to send out queries for information.

Benefits:

- Enables different organizations to easier work in parallel with the asset, may simplify maintenance
- More natural to host planning data and runtime data in different nodes

Figure 12: deployment Distribution



- Reflects distribution of data sources and services in today's distributed automation systems
- No single point of failure enables graceful system degradation
- Ability to seamlessly non-invasive extent AAS contents by adding new nodes
- Can lead to better system performance as it avoids having a single bottleneck

Drawbacks:

- Potential data inconsistencies: different nodes may report different values for single data elements, clients would be responsible for resolving conflicts (also true in today's systems)
- Potentially conflicting service requests: if an AAS service is hosted in multiple nodes, it may lead to race conditions
- No clear responsibilities for manipulating AAS data elements and services

Distributed AAS with Aggregating Node: Different network nodes store information for a single asset under the same asset identifier, but there is an aggregating node that provides a single entry points for clients (e.g., an aggregating OPC UA server). The aggregating node can either create local copies of the data from the distributed AAS nodes or simply provide references to the distributed nodes. The aggregating node may provide some measures to assure data consistency, e.g., providing a single reference for a specific data element in case this data element is stored in different AASs.

Benefits:

- Central management of potential data inconsistencies in the aggregating node
- Lower complexity for clients, only one communication partner
- Single authority for manipulating AAS items

Drawbacks:

- Requires additional setup and maintenance of aggregating node
- Potentially lower performance due to multiple involved network nodes (in case references are used)
- Potentially single-point of failure

5.5.3 View "Virtualizing Asset Administration Shells"

AAS may be deployed in different types of execution environments. An execution environment is required to both make the data elements of the AAS accessible and to allow executing AAS services, which may require operating system services. It is assumed that some kind of connection to the physical device according to the alternatives described in Section 6.5.1 is available.

Operating System Deployment: Direct deployment of an AAS to an operating system (OS) usually means that the AAS executes as a dedicated OS process or within another OS process. The AAS would be a permanently running service that starts up with OS start and shuts down with the OS.

Benefits:

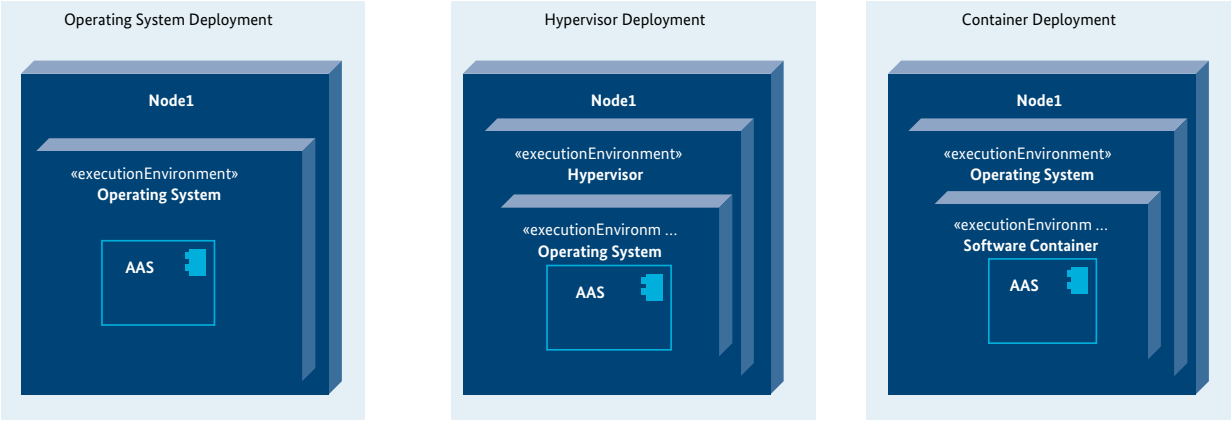
- Only one execution environment to administrate (e.g., configuration, security patches, backups), less complicated setup

Drawbacks:

- AAS process could affect other running services in the OS thus affecting asset operation
- In case of a real-time operating system, it may be more difficult to deploy the AAS

Hypervisor Deployment: AAS may be run in virtual machines which are managed by a hypervisor running in the host machine. This allows running multiple guest operating systems on a single node. The AAS would still run as an OS process, but share the physical hardware resources with other operating systems and applications.

Figure 13: deployment Virtualization



Benefits:

- Better utilization of powerful hardware
- May run RTOS and GPOS in parallel on the same hardware
- Virtual machines provide some form of isolation, i.e., the VM for the AAS may not be affected by faulty or insecure VMs running in parallel

Drawbacks

- More complicated set up and maintenance
- Possible performance losses

Container Deployment: Operating-system-level virtualization involves running applications inside software containers, on top of a regular operating system. Usually this involves some form of resource isolation, so that the applications view of the processes, network, and file systems is restricted.

Benefits

- Easy deployment of self-contained containers, flexibility, portability
- Isolation of applications to not affect the rest of the system in case of failures

Drawbacks

- Performance penalties
- Needs special OS support, only provided by few OSs

5.5.4 View "Lifecycle of AAS"

AASs may contain information from different sources, e.g. planning data from engineers, type data from manufacturers, or runtime data generated by an industrial process. The different sources also imply different deployments throughout the lifecycle of an industrial asset. The following distinguishes four transitions in the lifecycle of an asset and is inspired by the lifecycle phases underlying the AutomationML standard. These are not alternatives as in the former deployment views, but would need to be executed in a sequence.

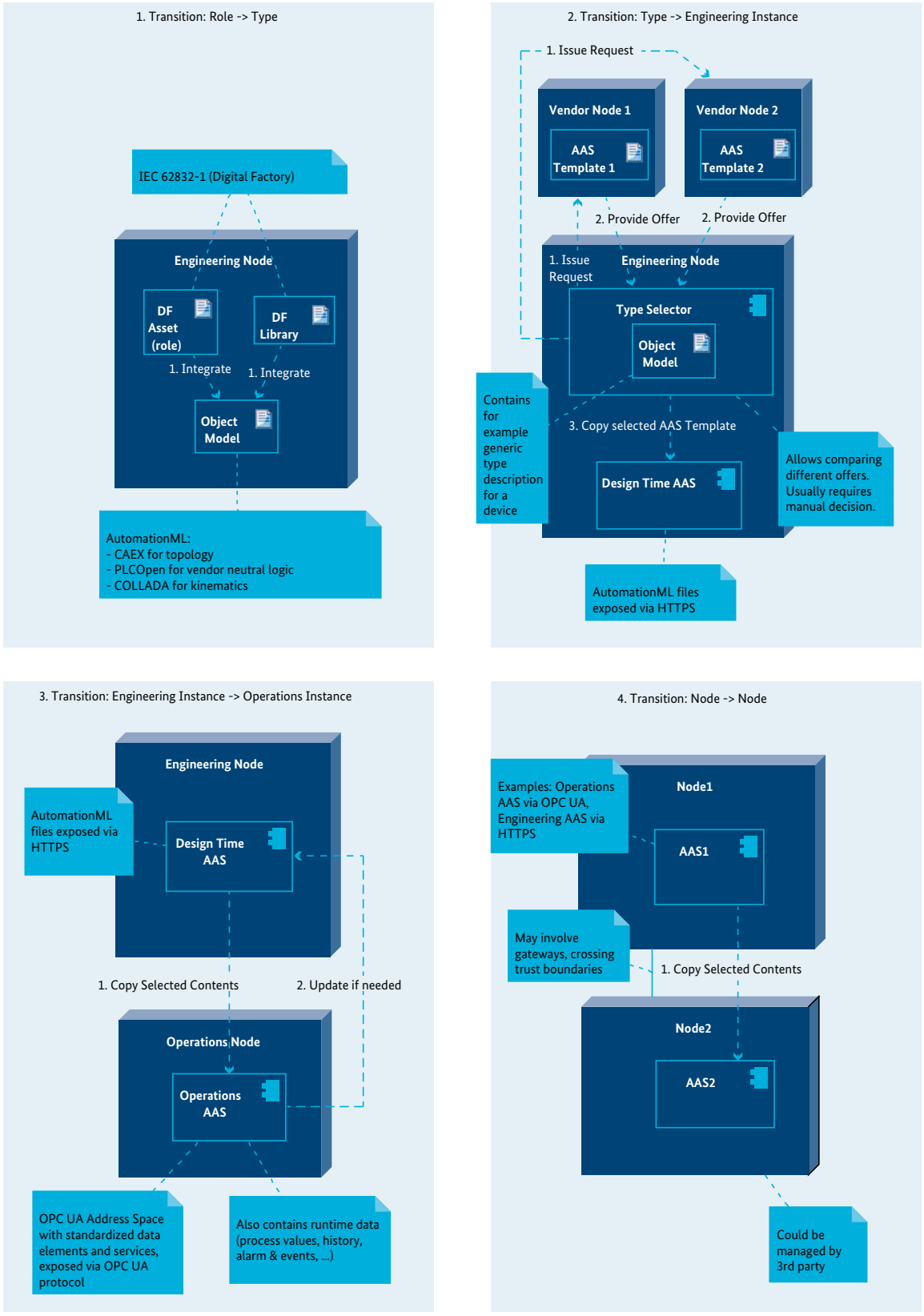
1. **Transition: Role → Type:** When planning an industrial plant, engineers first create a role-based DF Asset (from IEC 62832-1), which is still independent of particular devices. It may contain activities, such as mixing fluids, heating material, or drilling holes. Using a *DF Library*, the *DF Asset* can be transformed into an *Object Model*, which includes generic type descriptions for the required devices. This information could for example be specified in AutomationML, which provides CAEX for modeling a plant hierarchy, PLCOpen for describing generic control logic, and COLLADA for capturing kinematics and 3D data. The *Object Model* is a plain file without communication facilities, thus does not fulfill the criteria for an AAS.

2. **Transition: Type → Engineering Instance:** In this phase, the generic type descriptions are transformed into vendor-specific instances. To do so, a *Type Selector* component first takes the *Object Model* created in the former phase and issues requests for instances to the types to different vendors. As the Object Model contains device type descriptions in a standardized format (e.g., using eCl@ss classes and properties), device vendors can match these description to their own device library and provide fitting offers back to the *Type Selector*. These offers would be files and can be considered templates to create AASs. Usually the vendor would expose only a selection of its own AAS about a device type to the vendor, for example in order to protect its own IP. The *Type Selector* downloads the vendor proposals via HTTPS and compares them. It may present the engineer a visualization of the different vendor proposals, and the engineer decides for a certain vendor. Then the AAS template of the vendor is used to copy instance-specific information into the former *Object Model*. Additional steps may be needed at this point, e.g., converting the PLCOpen logic into some vendor specific language. The Object Model is now exposed via a HTTPS server, thus forming a Design Time AAS.

3. **Transition: Engineering Instance → Operations Instance:** During plant commissioning, engineers create an Operations AAS on an Operations Node (e.g., a plant side server, or an asset). They copy selected contents of the Design Time AAS into the Operations AAS, where they are exposed for example in an OPC UA address space. The Operations AAS will be used to store runtime data, e.g., process values, history, alarm & events, maintenance data. If data that was copied over from the Design Time AAS is changed during plant operation, the Operation AAS may need to update the Design Time AAS to assure data consistency. In case the Operations AAS is split among multiple nodes (as one variant in Section 6.5.2), then this transition from the logical Design Time AAS to the logical Operations AAS may involve numerous physical AASs, so that special attention needs to kept on data consistency.

4. **Transition: Node → Node:** During plant operation different parties may be interested in information and services from AAS for specific assets. For example, it may be desired to carry out data analytics on separate nodes with high computing capabilities, or a particular device needs replacement, so that its configuration would be transferred to the AAS of the asset replacing the device. In this case, the third party interested in AAS information and service would create its own AAS for the asset and copy over selected information from the original Operations AAS. This may require specific contracts between the parties, to stipulate which data may be used and which not. The same transition from node to node may also happen after a product with an AAS has been delivered to the customer and usage data shall be analyzed by a third party.

Figure 14: deployment lifecycle



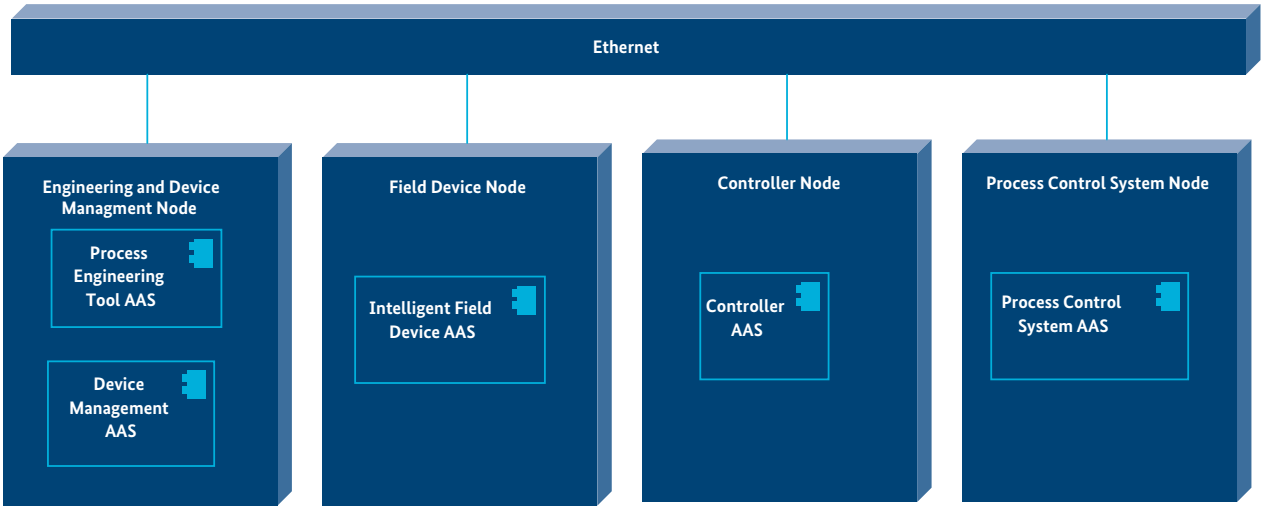
5.6 Deployment Views: Example for Device Integration

Figure 15 shows an example AAS deployment for the use case described in Section 4. This reflects a simple deployment, which is mapped to the deployment views as follows:

- Physical Proximity: Asset-based Deployment
- Distribution: Centralized AAS
- Nesting: No nesting
- Virtualization: Operating System Deployment, no virtualization
- lifecycle: Transition Engineering Instance → Operations Instance.

Note, that other deployment are not ruled out by this example and may have favorable benefits in specific contexts.

Figure 15: AAS Deployment: Example deployment PnP Deployment



6. Technology Mappings

The technology-agnostic models from Section 6 can be mapped to different technologies depending on the specific context they are used in. We describe a few exemplary technology mapping suggestions in the following.

6.1 Mapping AAS to OPC UA Device Interface (IEC 62541-100)

This subsection sketches how the technology-agnostic model from Section 6 can be mapped to the OPC UA Device Interface (IEC 62541-100), which is also used by FDI and FDT. In general, Data Elements can be expressed as UA Variables in the address space, and Services can be expressed as UA Methods. Figure 16 shows a mapping of the structures prescribed by both models focusing on the most relevant elements. IEC 62541-100 specifies not only a generic structure on how to arrange variables and objects, but also a number of concrete data elements (e.g., serial number) and services (e.g., locking).

Figure 16 indicates that an almost bi-directional mapping between both structures is possible. It would be possible to automatically transform existing IEC62541-100 Device Type models into models AAS models. Clients familiar with the AAS model semantics could thus also understand IEC 62541-100 models without any additional implementation.

6.2 Mapping AAS to MQTT/HTTP

This subsection sketches a mapping of the technology-agnostic model from Section 6 to MQTT topics and potentially HTTP services. MQTT topics resemble data models, but do not foresee function calls. Thus, AAS data elements can be mapped to MQTT topic elements in a straight-forward way, but AAS services need to be realized via other means, e.g., additional web services. The top-level MQTT topic would be the AAS id, which must be globally unique, so that the AAS is unambiguously represented in the topic space. Data elements would be mapped to topics named after the globally unique data element identifier (e.g., from eCl@ss).

Figure 16: Technology Mapping AAS <–> OPC UA Device Interface

AAS		Device Type (from IEC 62541-100)	Comments
Header	↔	Object (ParameterSet)	Functional Group Type "Identification"
Data Element (ManufacturerId)	↔	Variable(ManufacturerId)	Only example in IEC62541-100
Data Element (ModelId)	↔	Variable(ModelId)	Only example in IEC62541-100
Data Element (SerialNumber)	↔	Variable(SerialNumber)	Only example in IEC62541-100
Data Element (<ProfileId>)	↔	Variable(<ProfileId>)	Only example in IEC62541-100
Data Element (SecurityClass)	X		Could be mapped to a Parameter
Body	X		No corresponding element in IEC 62541-100
Collection of Data Elements (Device Parameters)	↔	Object (ParameterSet)	
Data Element (<ParameterIdentifier>)	↔	Variable (<ParameterIdentifier>)	A number of parameters can be specified here
Collection of Data Elements (Device Properties)	X		No corresponding element in IEC 62541-100
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		No corresponding element in IEC 62541-100
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>)	A number of methods can be specified here
Service (Lock)	↔	Lock	Locking specifies additional services
View (<GroupIdentifier>)	↔	FunctionalGroupType (<GroupIdentifier>)	
View (Identification)	↔	Identification	
Reference	X		No corresponding element in IEC 62541-100

Figure 17: Technology Mapping AAS <–> MQTT Topics / HTTP services

AAS		MQTT	Comments
Header	↔	/aas Id/header	
Data Element	↔	/aas Id/header/Data Element Id	
Body	↔	/aas Id/body	
Collection of Data Elements	↔	/aas Id/body/Collection of Data Elements Id	
Data Element	↔	/aas Id/body/Collection of Data Elements Id/Data Element Id	
Collection of Services	X		
Service	↔	Custom HTTP service (GET/PUT)	
View	X		Use topic filters to emulate or define new topic referencing other topics
Reference	↔	/aas Id/body/references/	

MQTT does not directly support the concept of Views, but it provides a filter mechanism to use wildcards in topic names to create custom views. To map more complex views, which may also refer to services, additional MQTT topics would need to be created in the MQTT topic space and the data elements would need to be copied under these topics.

As default services, the MQTT protocol provides publish, subscribe, unsubscribe, connect, and disconnect. Other services, e.g., as in Figure 6 defined by GMA 7.21, would require additional HTTP services. MQTT does not provide information modelling means as sophisticated as the OPC UA framework and is thus more restricted as a technology for realizing I40 AAS. However, for resource-constrained and mobile devices it may provide a fitting solution.

6.3 Mapping AAS to NAMUR MTP

Premise of modular automation in process industries is to quickly build up chemical-pharmaceutical plants by assembling pre-built plant modules (e.g., for distilling, mixing, filtering etc.). Multiple field devices are combined into such a module, which has standardized hardware and software interfaces. In 2013 the NAMUR recommendation NE 148 ("Automation Requirements relating to Modularization of Process Plants") was published. Meanwhile, the German ZVEI created multiple working groups to establish standards for module descriptions called Module Type Packages (MTP).

The basic idea is to encapsulate the module internals and expose only high-level module states (e.g., running, restarted, paused) via a state model (e.g., based on ISA 88). A process control system can then issue state change requests (e.g., run, pause, stop) to the modules according to the production schedule, which the module then handles through its internal control logic. Process values can be communicated to the process control system via OPC UA. Besides the state-logic also custom visualizations for human interaction via the process control system can be integrated into the module type package. Engineering would be split into module engineering, which would be independent of the actual plant the module is used, and plant engineering, which would then only compose the plant modules on a higher level for a given plant.

The specification of NAMUR MTPs is still under discussion, so mapping AAS concepts to MTPs must still remain preliminary. In general, the services, state models, HMI elements and signal interfaces exposed by the OPC UA servers of individual modules could be considered as a kind of AAS for the process module, since they allow exploring the module and interacting with it. It is also beneficial that the MTP specification is vendor-neutral and oriented towards standards, e.g., IEC 61512, NE 150, IEC 62541, etc., therefore making it compatible with Industrie 4.0 interoperability goals.

The NAMUR MTP manifest is an AutomationML configuration file (in XML) that links the different parts of the MTP together and provides and entry point for an engineering tool importing such a package. It cannot be directly mapped to an AAS model, rather the included OPC UA server's address space model would be suited for such a mapping. This address space model is however not fully specified as of now, although first mappings exist [13]. In general AAS *Services* could be mapped to MTP *Service*, while AAS Data Elements could be mapped to MTP *Module States*, *HMI graphs* and *SWSignals*. The MTP also provides a link to a field device description according to FDI or FDT, which would then use the IEC 62541-100 model internally, whose AAS mapping is described in Section 7

6.4 Mapping AAS to OpenAAS

OpenAAS provides another structuring proposal for AASs. Figure 18 depicts the property metamodel of OpenAAS. There are additional OpenAAS specifications for services and interaction patterns.

Figure 19 show a rough mapping of the AAS metamodel to the OpenAAS model. OpenAAS uses *SubModels* to structure the AAS contents. It does not differentiated explicitly between a header and a body, but this concept could be mapped to two different *SubModels*. OpenAAS *SubModels* contain *PropertyValueStatementLists* and *Services*, which roughly correspond to *Collection of Data Elements* and *AAS Services*. OpenAAS also includes *Views*, which however only reference properties and *LifeCycleEntries*. The *LifeCycleArchive* and *LifeCycleEntries* of OpenAAS can be mapped to a separate Collection of Data Elements in the AAS model.

Figure 18: OpenAAS Property Model [21]

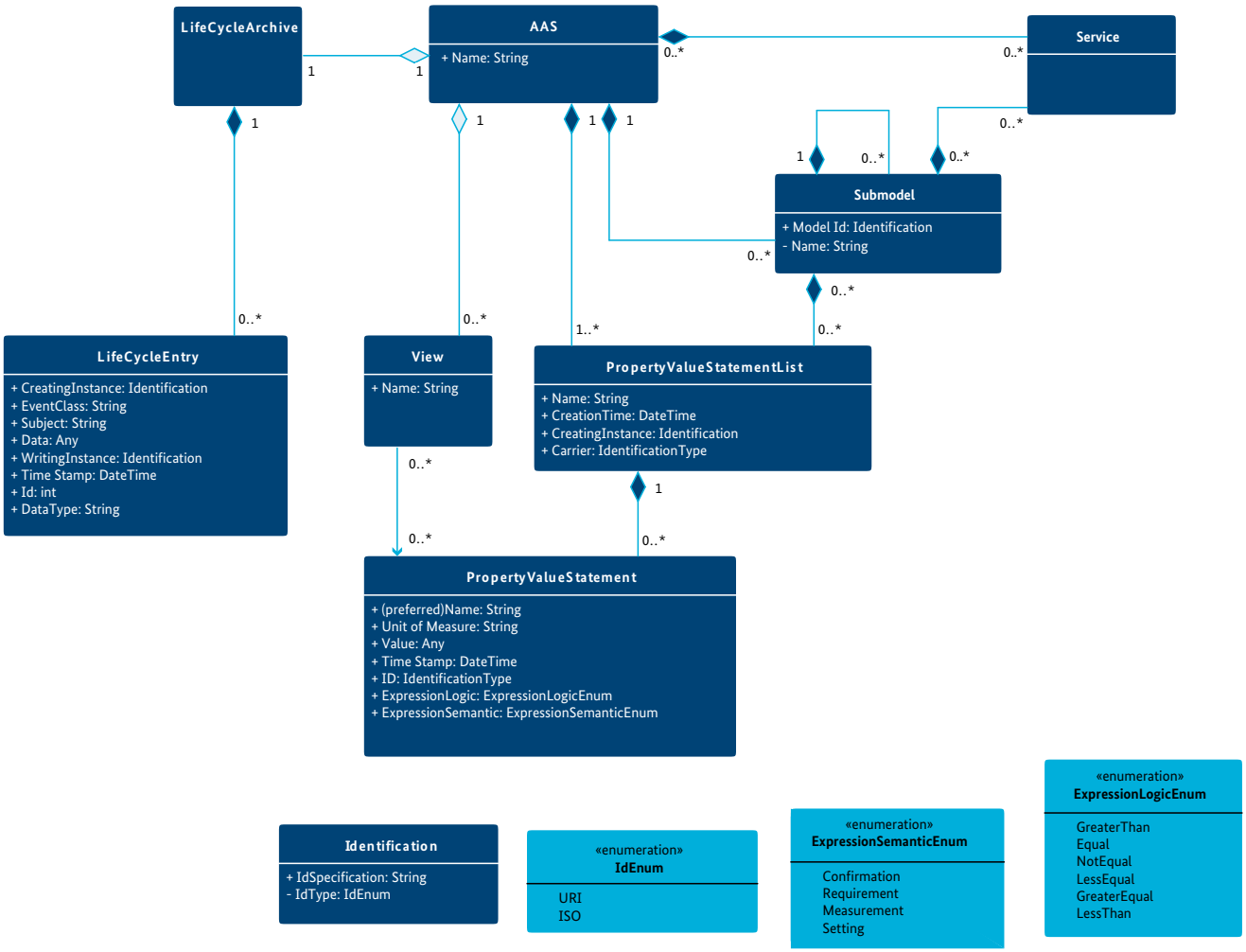


Figure 19: Technology Mapping AAS <–> OpenAAS

AAS		OpenAAS
Header	↔	SubModel
	X	PropertyValueStatementList
Data Element	↔	PropertyValueStatement
Body	X	
Collection of Data Elements	↔	SubModel
Collection of Data Elements	↔	PropertyValueStatementList
<Attributes to be defined>	↔	IdentificationType
Data Element	↔	PropertyValueStatement
Collection of Services	↔	SubModel
Service	↔	Service
View	↔	View
Reference	X	
Collection of Data Elements	↔	LifeCycleArchive
Data Element	↔	LifeCycleEntry

7. Example Realization: PnP using OPC UA & FDI

This section describes an example realization of the use case "Auto Integration of Field Device" specified in Section 4. As Section 7 has shown, the generic models from Section 6 can be mapped to different technologies and communication protocols. The example in this section uses OPC UA for the communication and the OPC UA Device Information Model to implement AAS data elements and services. Implementations based on other technologies are possible.

In the example, a commissioning engineer connects a temperature transmitter into a plant network, and a device management server automatically configures it and rudimentary integrates it into to the production. In the context of this document, the example should be considered as a vehicle to evaluate the capabilities and limits of existing standards, not as a normative reference implementation for the use case.

7.1 Static Views

Figure 20 shows the "Industrie 4.0 Components" and other components involved in this scenario, which correspond to actors listed in Section 3.3.1.

- **FDI Node I40 Component:** acts as the device management server and is responsible for transferring configuration data to field devices.
- **Temperature Transmitter I40 Component:** acts at the Intelligent Field Device to be plugged into the system. It carries a default parameterization, which can be overwritten by the FDI Node. It also carries a specific ID that was assigned to the device during engineering of the plant by the plant owner and transferred to the device vendor when the device was ordered. This ID allows looking up the planned configuration of the device during commissioning.
- **Controller I40 Component:** executes control logic based on the sensor values received from the temperature transmitter.
- **PCS I40 Component:** the process control system provides a human operator a supervision facility and can show the most recent sensor values received from the temperature transmitter.
- **DHCP Server:** assigns a network address to the newly connected component.

Figure 20: Industrie 4.0 Components

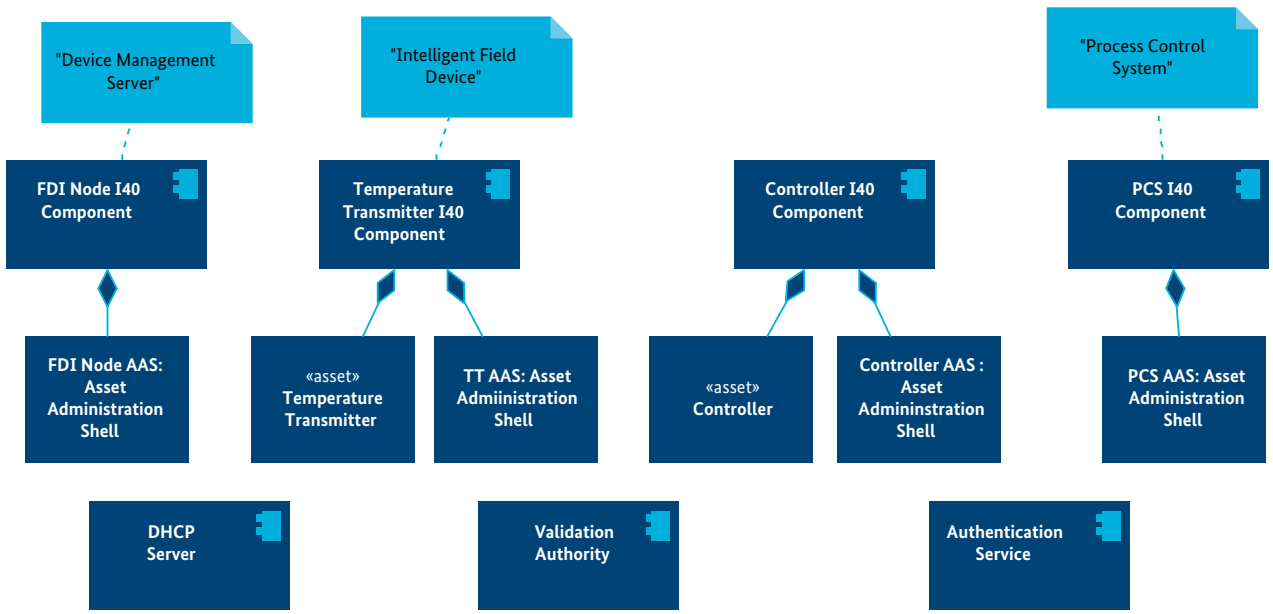
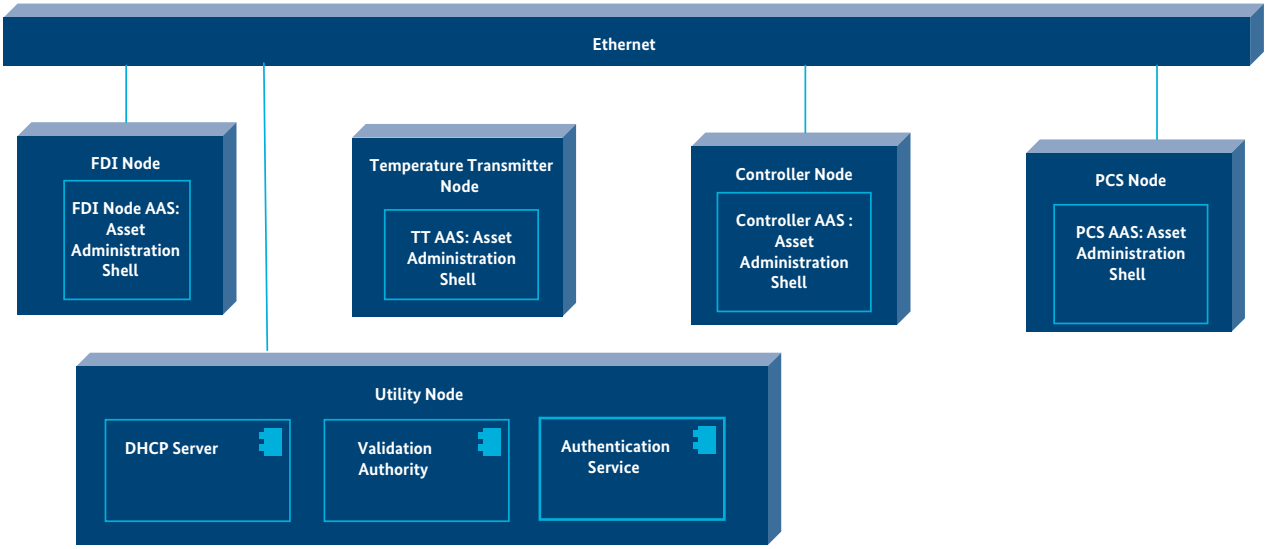


Figure 21: Deployment of the I4.0 Components

deployment PnP for Field Devices (Static Component / OPC UA)



- **Validation Authority:** verifies the validity of a digital certificate.

- **Authentication Service:** provides a central database to verify user tokens provided by clients.

Figure 21 depicts the deployment of the components to nodes and network connections. All components reside on a local area network using Ethernet. Each AAS resides on its own dedicated node. Other deployment (see Section 6.5) are possible as well, the plug-and-produce functionality is almost independent of the chosen deployment.

Figure 22 provides several high-level details on the involved AAS's contents. The FDI Node AAS includes an OPC UA Local Discovery Server (IEC 62541-12) as well as an OPC UA Server that can provide device information to interested parties, and an OPC UA client to manage the interaction with newly connected devices. These elements address the use case's requirement BasConn-1 (Automatic network connectivity).

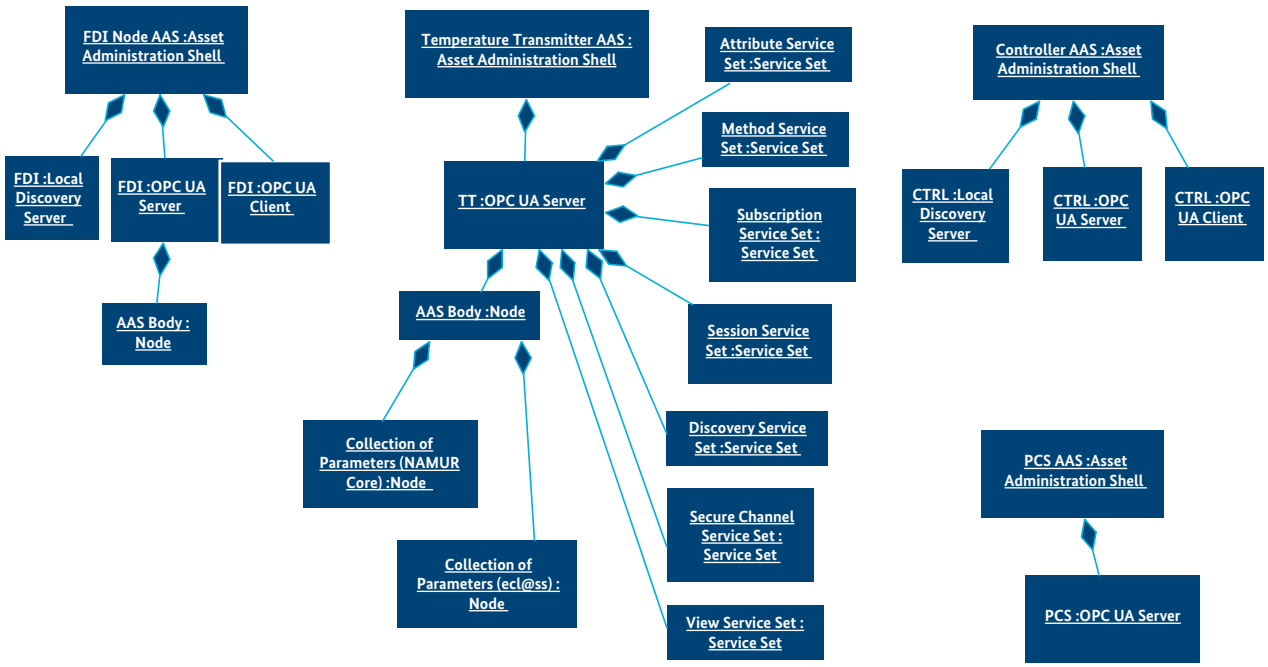
The Temperature Transmitter AAS contains an OPC UA Server to fulfil the use case's requirement NetInt-1 (Interoprable Modules). It provides the NAMUR NE 131 Core Parameters for temperature sensors as standardized I40 data elements and properties of eCl@ss class 27-20-02-06 for temperature transmitters. These data elements are vendor-neutral and can thus be interpreted by any party with an understanding of the standards. The data elements address the use case's requirement SemUnd-1 (Self-describing Modules)

As services, the Temperature Transmitter AAS provides the OPC UA standards services (IEC 62541-4), which allow connecting, browsing, reading, writing, and for example setting up subscriptions. This also contributes to requirement NetInt-1.

The Controller AAS and the PCS AAS again contain OPC UA Servers and Clients to interact with connected devices.

Figure 22: Details of the involved AAS

object PnP for Field Devices (Static / OPC UA)



7.2 Dynamic Views

The following figures depict how the components described in Section 8.1 interact to implement the use case described in Section 4. Figure 23 provides a high-level overview of the involved communication partners. As there are numerous interactions between the components, the following breaks down the overall interaction into six separate steps, each depicted as an individual interaction diagram.

After a commissioning engineer has physically plugged-in the temperature transmitter into an Ethernet switch in step 1, it contacts the DHCP server to retrieve an IP address. With the IP-address, the temperature transmitter calls the function RegisterService2 from IEC 62541-12 (OPC UA Discovery), which the Local Discovery Server on the FDI node can discover using a Multicast Probe request. After the temperature transmitter's OPC UA server has been registered with the Local Discovery Server, the FDI node's OPC UA client can discover the endpoint URL of the transmitter.

Figure 23: High-level Interaction Overview for the example implementation

sd PnP for Field Devices (Dynamic / OPC UA)

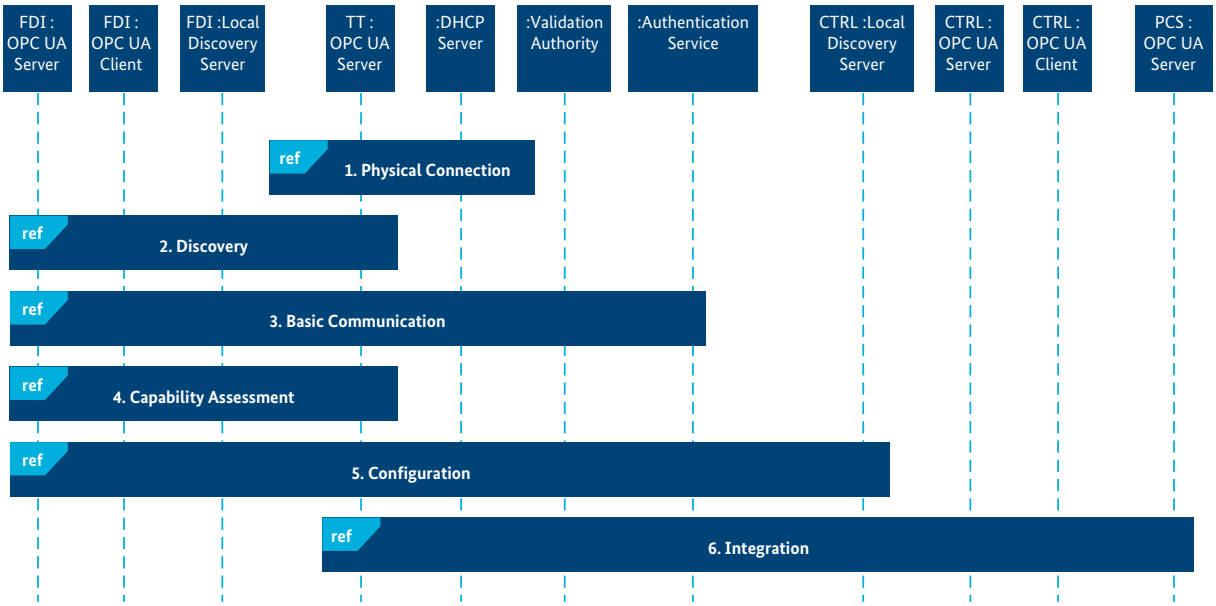


Figure 24: Discovery

sd Discovery

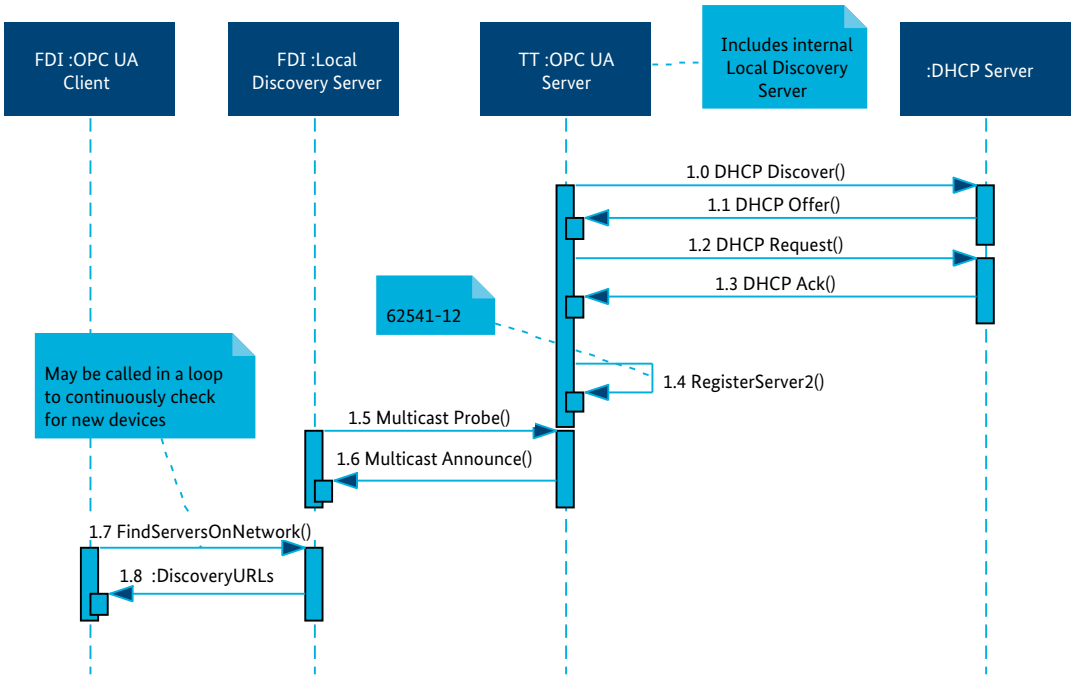


Figure 25 shows the regular OPC UA client/server interaction to start a session on the temperature transmitter's OPC UA server. The FDI node's OPC UA client involves the Validation Authority to validate the server certificate and the uses an Authentication Service to the desired access rights (IEC 62541-7).

Finally, the FDI node's OPC UA client can browse the header of the temperature transmitter's AAS and retrieve the device ID (Figure 26). If the FDI node can find the device ID in the engineering configuration, the client proceeds to read the default device parameters from the device. With this information, the FDI node can perform a device capability assessment and decide how to configure and integrate the device.

Figure 25: Basic Communication

sd Basic Communication

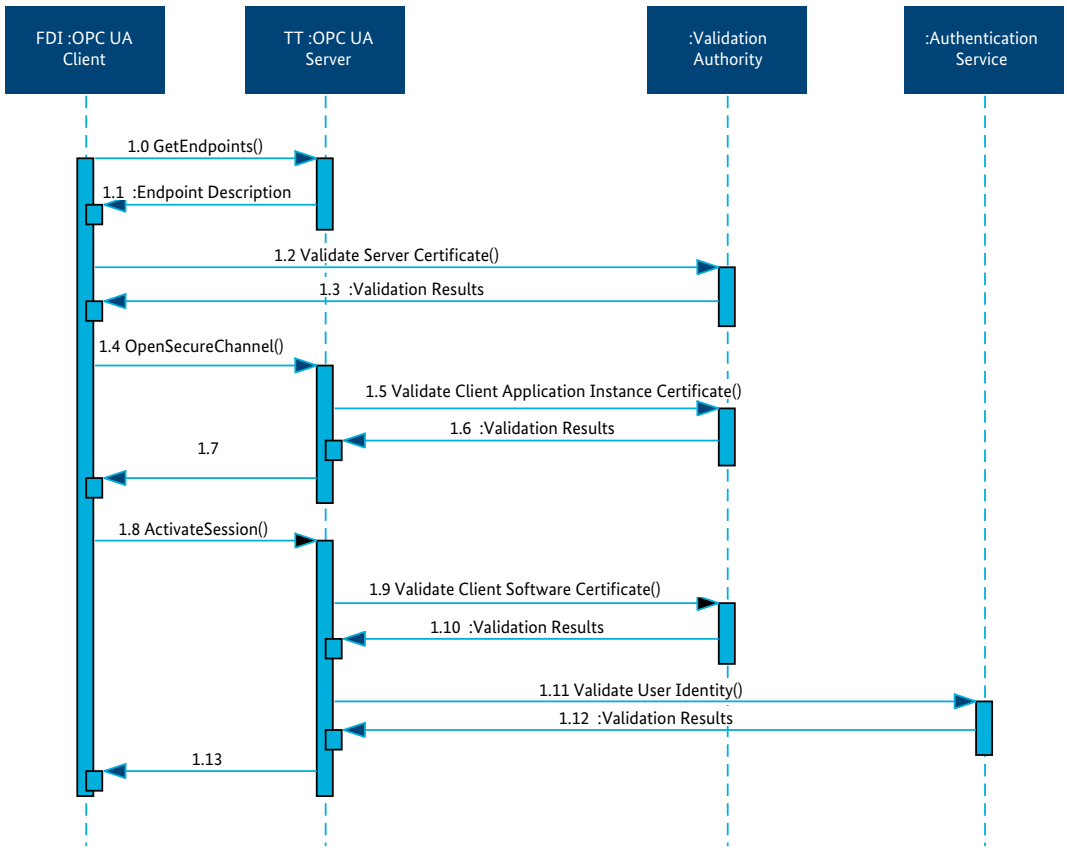
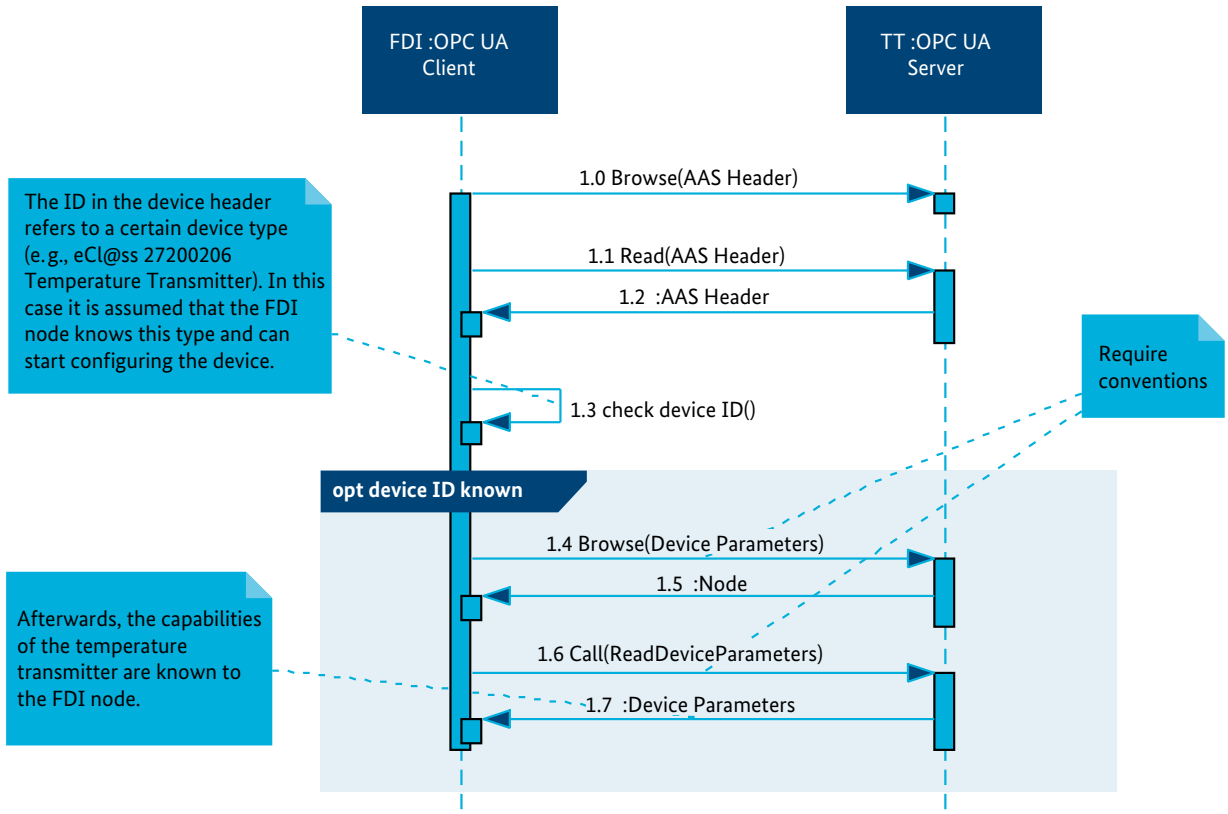


Figure 26: Capability Assessment

sd Capability Assessment



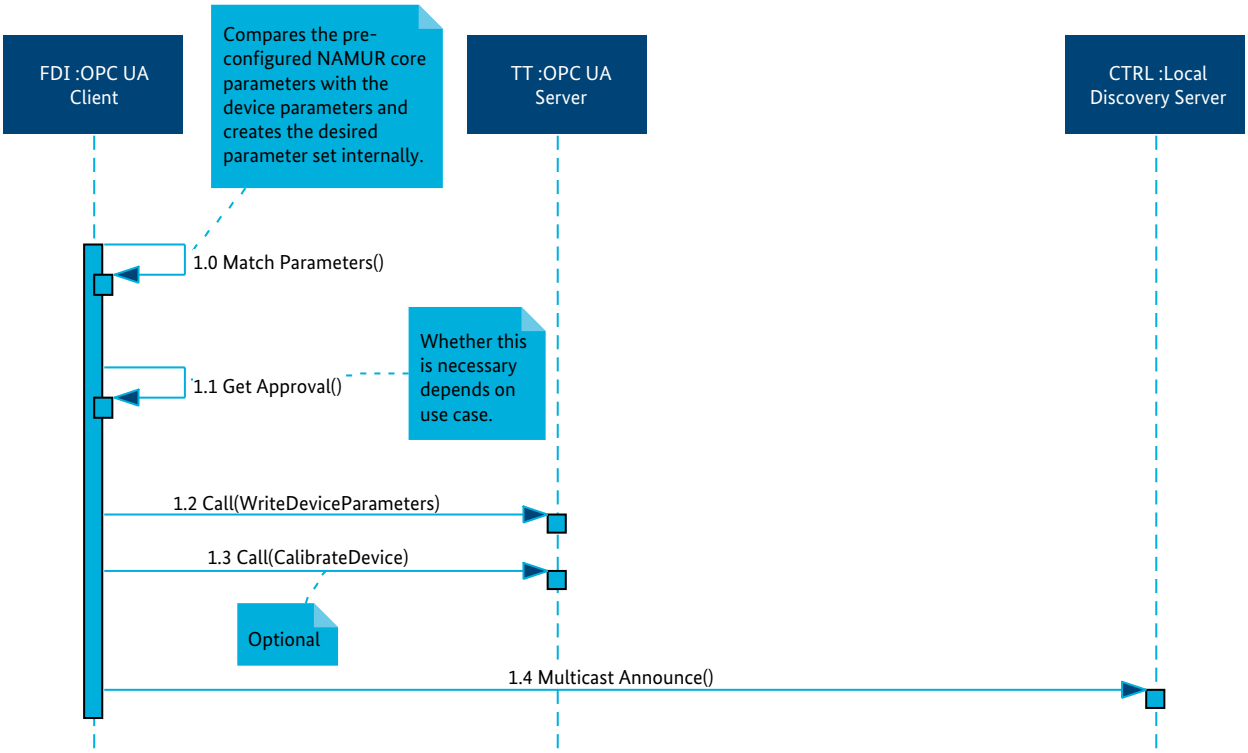
In this case, the FDI I40 component would look up the NAMUR core parameters that were planned for the device during engineering (Figure 27). Any other constraints that may prevent or limit the ability to configure the device (e.g., other high priority tasks) would need to be considered by the FDI I40 component at this point (requirement DisConf-1, autonomous constraints detection).

If modifications to the pre-specified configuration would be required due to the detected constraints, the FDI I40 component would calculate an updated set of configuration parameters (requirement DisConf-3). Otherwise the parameters from engineering are taken as-is. As an optional step, it shows these parameters to the commissioning engineer (e.g., on a mobile device) and asks for approval. If the commissioning engineer grants the approval, the FDI I40 component uploads the parameters to the device overwriting the default configuration.

Depending on the device, it also may invoke an automatic calibration routine, that the device offers a standardized I40 service. Once the device has calibrated itself, the FDI I40 component sends a multicast announce to the network, thereby informing the Controller's Local Discovery Server of the presence of the device. Most of these steps can be executed automatically, thus addressing the use case's requirement QoS-1 (Fast configuration).

Figure 27: Configuration

sd Configuration



In the following (Figure 28), the controller integrates the temperature transmitter into the overall production process. It may for example read out the device's properties after browsing for information relevant to the Controller. As for example the current temperature value measured by the device is stored in its AAS with a standardized NAMUR Core Parameter, the Controller can retrieve this value, and also subscribe to updates using the I40 services provides by the temperature transmitters AAS. The Controller can for example also inform the Process Control System of the presence of the new temperature transmitter and then forward the subscription updates, so that a human operator or engineer can utilize the new sensor information (addressing requirement DisConf-2 and ConnHmi1-1).

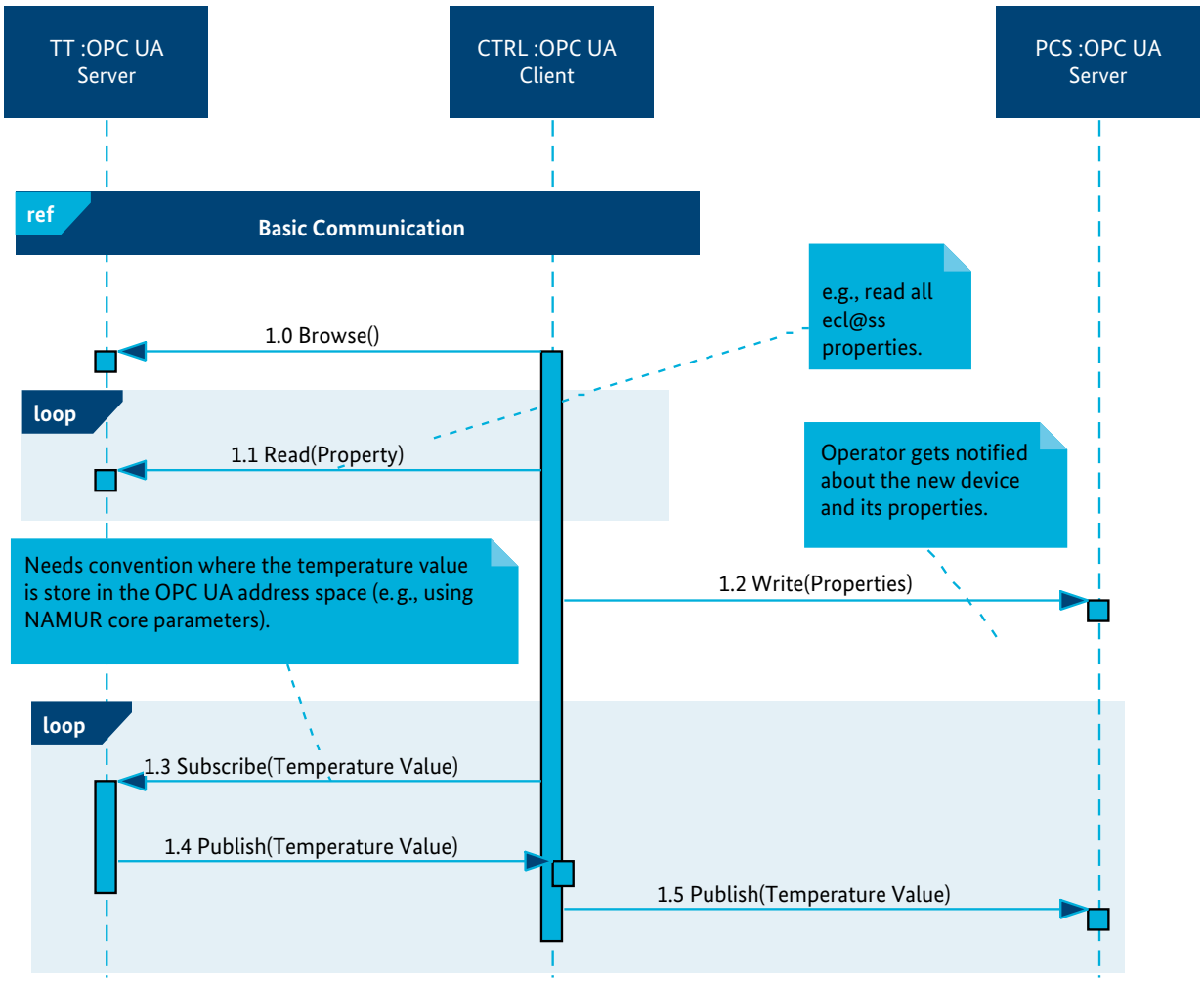
Depending on the particular system, other devices and application may need to be informed of the newly connected sensor as well. The procedure for such integration may be similar as for the controller.

Notice that the interaction described so far is only a single example and may require more or fewer steps in other contexts. A few requirements described in Section 4 have not be considered in this example, e.g., SemUnd-2 (Modular Engineering) and DisConf-4 (Auto-update system), because they were not required to get the device in this example operational.

The intention of the example is to illustrate the use of standards in the context of a Plug&Produce scenario, so that the limits of existing standards can be better understood. The next section briefly summarizes a prototypical implementation of the scenario, before Section 9 discusses the involved standards with more detail.

Figure 28: Integration

sd Integration



7.3 Prototype Implementation

A prototypical implementation shows how the example realization described in Section 8.1 and 8.2 can be put into practice.

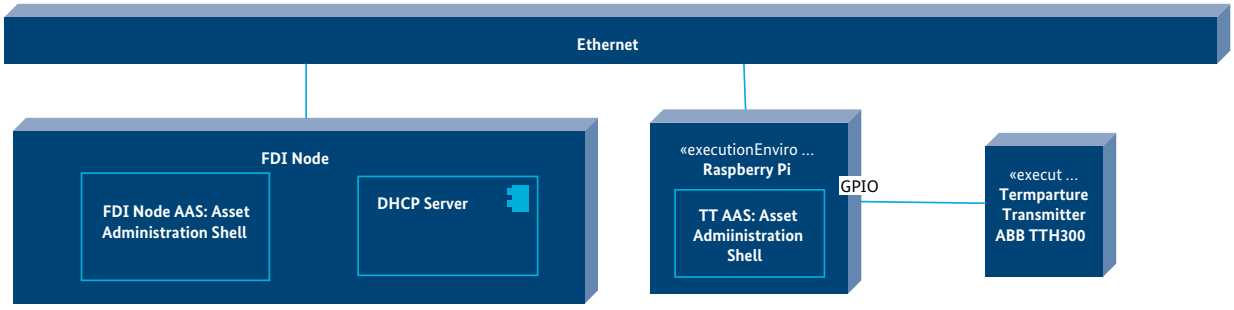
The prototype realizes the different asset administration shells as OPC UA Servers implemented using the ProSys OPC UA Java SDK. The temperature transmitter is an ABB TTH 300. It is connected to a Raspberry PI that hosts the OPC UA server representing its AAS. This OPC UA server holds the default parameterization of the temperature transmitter in its address space and also provides a unique ID that was assigned to the device during engineering and provided to the vendor, when ordering the specific device. This ID allows the FDI Node AAS, in this case a simple Java program, to retrieve the device parameters from an AutomationML XML file that provides the NAMUR NE 131 core parameters for the device embedded into a NAMUR NE 150 representation of the plant.

After all nodes have IP addresses assigned by the DHCP server, the FDI Node AAS finds the newly connected Temperature Transmitter AAS using OPC UA Local Discovery Servers and mDNS. Its OPC UA client connects to the new devices and reads out the default parametrization and the ID. Afterwards the device configuration in the AutomationML file is retrieved. The FDI Node AAS shows the found configuration to a human user for approval. After approval, the parameters are downloaded to the device and the device is set operational.

To assess the quality of the implementation the use case's key performance indicators "setup rate" and "availability" from ISO 22400-2 would need to be computed (see Section 4.1.5). This would require a baseline KPI value for classical manual configuration and integration of a device, as well as certain assumptions about the intended product duration. Using this information, a financial calculation is possible to estimate the benefits of the whole approach. Because the prototype is still implemented in a simplistic fashion, such a calculation has not yet been attempted.

The prototype provides a rudimentary implementation of the example realization described before. Security features as well as controllers or process control systems have not yet been implemented. Notice that this is a proof-of-concept prototype, not a product sold in the market. All software and hardware is based on standards without any proprietary information models or network protocols. Thus, the devices in this prototype could be exchanged with devices from other vendors.

Figure 29: Deployment of the Prototype implementation



8. Evaluation of Standards

Multiple standards are required to implement the example realization of the use case “Plug and Produce for Field Devices” described in Section 8. This section attempts a first evaluation of the standards and tries to identify further standardization needs.

8.1 Communication

IEC 62541 (OPC UA): provides a rich and generic communication framework that permits an easy mapping of the AAS concept to OPC UA address spaces. It supports expressing data elements, services (methods), and views. Data elements in OPC UA can have attributes according to IEC 61360. OPC UA provides more than 30 basic services for example for reading, writing, subscriptions, authentications, and events. The device discovery needed for the Plug&Produce case is supported by the OPC UA Discovery Mechanism (IEC 62541-12).

OPC UA over regular TCP/IP is usually not used for real-time communication, but extensions in the area of Time-Sensitive Networking (TSN) are planned so that this could become possible. In the example realization described before, OPC UA is also used to communicate the process values to the controller, this would usually require a more deterministic real-time connection in typical plants. OPC UA provides a number of security features, which have been used in the example realization, but the standard and implementations are still being updated to fulfill requirements by security experts.

RFC 2131 (DHCP): helps in the basic auto-configuration of network devices. The functionality is sufficient for the use case under analysis. It requires additional network administration to setup the DHCP server.

mDNS (RFC 6762): the multicast Domain Name System resolves host names to IP addresses within small networks that do not include a local name server. When an mDNS client needs to resolve a host name, it sends an IP multicast query message that asks the host having that name to identify itself. That target machine then multicasts a message that includes its IP address. All machines in that subnet can then use that information to update their mDNS caches. The mechanism can be used in combination with IEC 62541-12 (OPC UA Discovery).

8.2 Information

IEC 62769-3 (FDI): includes the device description structure based on IEC 62541-100 (OPC UA Device Information Model for Field Devices) and thus provides means to express the structure of data elements of a device in a standardized way. Which data elements are included in this structure is to some extent up to the vendor providing the specification. FDI allows using both Electronic Device Descriptions (EDD) and Field Device Tool (FDT) specifications, thus providing an easy migration path for existing device descriptions. The device parameters in EDDs must be specified according to a standards as EDD only prescribes key/value pairs, but no standardized keys. Fieldbus profiles can be used for this. Type information (i.e., device properties) about a device can be also integrated into the IEC 62541-100 structure.

IEC 62543/ISA103 (FDT/DTM): Field Device Tool (FDT) is an open standard and was created to integrate user applications for field devices. Device vendors specify device-specific software components called Device Type Manager (DTM), which can be embedded into a single frame application. Device communication can be implemented in the frame application or the DTM itself. The intended use case is manual device parametrization using a Windows PC, but the use in a more automated Plug&Produce scenario is conceivable. FDT Frame applications can conduct network scans to discover connected devices. FDT provides an open architecture that is standardized independent of industrial automation network. It allows for a comprehensive network integration model allowing for seamless integration mapping to connect intelligent assets relaying device-specific diagnostics data enterprise-wide. The architecture promises to adapt to any field communication protocol, and the use of network tunneling allows an FDT/FRAME™ to seamlessly talk through any number of disparate network layers to the end device [11].

eCl@ss: provides a meta-model for properties based on a subset IEC 61360 and a catalog of 41000 product classes and 17000 properties. For field devices, classes and properties of IEC 61987 have been integrated in to eCl@ss. For example, the class for Temperature transmitters contains 77 different properties ranging from min/max operating temperature, dimensions, supported communication protocol to type of user interface. These properties relate to type information about a field device and are usually used in procurement processes to order devices according to specific required

features. In a simple Plug & Produce scenario they are not needed, assuming that a specific device type from a specific vendor has been chosen already in engineering. In more sophisticated cases, a Plug & Produce scenario could additionally support formerly unknown devices, i.e., devices that had not been considered during initial engineering. In this case, the device management server could search for devices according to the required eCl@ss properties and connect to matching devices in order to configure them.

eCl@ss has varying depths for different device types in terms of the number of properties specified. While measurement instruments for example are a specifically richly specified eCl@ss class, robot-related classes, which could be relevant in many manufacturing scenarios often only contain generic ordering properties, such as the brand and the manufacturer name. eCl@ss also specifies the semantics or definitions of properties as prose text, which is not directly machine-interpretable. It needs to be determined if the existing definitions are sufficient so that algorithms based on them can be correctly implemented by different vendors. Finally, there are also currently initiatives to map fieldbus profiles to eCl@ss, which would provide more means for configuration of communication related information.

NAMUR NE 131 (Core parameters): To support fast and automatic field device parameterization, the NAMUR is working on a set of standard device parameters, called Device-Core-Parameters. Similar to the approach in the automobile industry, where standard functions such as brakes and wipers are standardized, these parameters shall allow basic startup of a device in a vendor-neutral way. The set contains 38 standards parameters, of which 18 are independent of the measurement approach. They shall be integrated into a revision of NE 131. For the Plug&Produce use case, these parameters allow an initial high-level and vendor-neutral configuration of a sensor and have consequently been used in the example realization. Additional parameters are vendor-specific, so that tuning a device with specialized parameters is not possible based on the NE 131 parameters.

IEC 62714 (AutomationML): the Automation Markup Language is a neutral XML data format for the storage and exchange of plant engineering information. Goal of AutomationML is to interconnect the heterogeneous tool landscape of modern engineering tools in their different disciplines, e.g. mechanical plant engineering, electrical design, HMI development, PLC, robot control. It incorporates different standards, i.e., CAEX (IEC 62424) for topology modeling, COLLADA for geometry modeling and kinematics, and PLCopen for logic modeling. AutomationML also provides a standard way of referencing eCl@ss properties. There is also a companion standard describing how AutomationML models can be communicated via OPC UA, therefore allowing easy integration into runtime AAS. For PnP approaches, CAEX models can hold information about devices (e.g., structured using NAMUR NE 150), this was used in the example realization described before. It would also be possible to generate PLCopen logic for a re-configuration process in a PnP scenario.

Fieldbus profiles: application specific fieldbus profiles can potentially be used for PnP functionality when dealing with fieldbus communication. For example for PROFIBUS, HART, and Foundation Fieldbus, there are such profiles for process control devices. They provide for example parameter lists for certain type of function blocks and thus allow a more fine-grained configuration of a devices that goes beyond the NAMUR core parameters. It needs to be determine how these profiles could be used by I40 components.

8.3 Standardization Gaps

Regarding communication protocols, existing standards such as OPC UA are much more powerful than needed for a basic PnP scenario. There are also multiple standards to implement the requirement for an automatic network discovery, to prevent the need for manual network configuration in a PnP scenario. OPC UA communication is currently not available in a deterministic way, which would enable real-time communication, but this shall be addressed by communicating OPC UA via Time-Sensitive Networking soon.

The possibilities of PnP are rather limited by the availability of rich property catalogs and potentially higher-level services. eCl@ss properties for sensors seem adequate to support searching for required sensors in a flexible I40 scenario, because they provide comparably fine-granular type information for such devices. Standardized configuration parameters are currently restricted to the NAMUR NE 131 Core Parameters, which allow basic configuration, but do not support more sophisticated, possibly vendor-differentiating fine-tuning of a device. Also maintenance-related data elements currently lack standardization.

For a PnP scenario, it may also be useful to have higher-level services in the devices that allow manipulating the state of the device (e.g., un-configured, ready to operate, malfunctioning) or even negotiate Quality-of-Service related properties (e.g., resolution of the sensor value).

9. Conclusions and Next Steps

This whitepaper described a use case definition, a conceptual modeling, and prototype implementation for the use case “Plug and Produce for Field Devices” derived from the Plattform Industrie 4.0 application scenario “Adaptable Factory”. The paper provides a first step for a deeper analysis of standardization gaps and scoping the required work to close these gaps.

In order to substantiate the evaluation of standards, the existing prototypes need to be extended and refined. More domain experts need to express their views on the use case. Multiple variants of the use case should be worked out, and more example implementations with different technologies can be created. Based on the learning, standardization committees should augment existing standards (e.g. richer property catalogs), so that a more sophisticated PnP becomes possible.

In the long term, a true vendor-agnostic PnP for industrial devices fully based on Industrie 4.0 standards is envisioned. This could help plant and factory owners in reducing commissioning times and making them much more flexible than today. End-customers could benefit from the possibilities of more individualized products that may result from this flexibility. The faster commissioning times could decrease also decrease time-to market for new products, thus providing end-customers faster access to products based on novel designs.

10. Annex

Annex A

References

1. **Plattform Industrie 4.0:** *"Aspects of the Research Roadmap in Application Scenarios"*, April 2016, http://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/aspects-of-the-research-roadmap.html

2. **Acatech Working Group Industrie 4.0:** *"Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0"*, April 2013, https://www.bmbf.de/files/Umsetzungsempfehlungen_Industrie4_0.pdf

3. **VDI/VDE Status Report, GMA 7.21:** *"Industrie 4.0 Components – Modeling Examples"*, November 2016

4. **Plattform Industrie 4.0:** *"Implementation Strategy Industrie 4.0"*, January 2016, http://www.zvei.org/Publikationen/Implementation-Strategy-Industrie-40-ENG.pdf

5. **Plattform Industrie 4.0:** *"Struktur der Verwaltungsschale: Fortentwicklung des Referenzmodells für die Industrie 4.0-Komponente"*, April 2016, https://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/struktur-der-verwaltungsschale.pdf

6. **Plattform Industrie 4.0:** *"Fortschreibung der Anwendungsszenarien"*, October 2016, http://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/fortschreibung-anwendungsszenarien.html

7. **Plattform Industrie 4.0:** *"Interaktionsmodell für Industrie 4.0-Komponenten"*, November 2016, http://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/interaktionsmodell-i40-komponenten-it-gipfel.html

8. **Fraunhofer IOSB:** *„Begrifflichkeiten rund um Industrie 4.0"*, https://www.iosb.fraunhofer.de/servlet/is/48960/

9. **VDI/VDE Status Report, GMA 7.21:** *„Industrie 4.0 Service Architecture: Basic concepts for interoperability"*, https://www.vdi.de/fileadmin/vdi_de/redakteur_dateien/gma_dateien/END_3_Status_Report_Industrie_4_0_-Service_Architecture.pdf

10. **Großmann, D., Braun, M., Danzer, B., & Riedl, M.** (2013). *FDI Field Device Integration*.

11. http://fdtgroup.org

12. http://www.zvei.org/Publikationen/ZVEI-White-Paper-Modulare-Automation.pdf

13. **Wassilew, S., Urbas, L., Ladiges, J., Fay, A., & Holm, T.** (2016). *Transformation of the NAMUR MTP to OPC UA to allow plug and produce for modular process automation*. Proceedings 21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2016)

14. **Weyer, S., Schmitt, M., Ohmer, M., & Gorecky, D.** (2015). *Towards Industry 4.0-Standardization as the crucial challenge for highly modular, multi-vendor production systems.* IFAC-PapersOnLine, 48(3), 579–584.

15. **Krüning, K., & Epple, U.** (2013). *Plug-and-produce von Feldbuskomponenten*. atp edition, 55(11), 50–56.

16. **Dürkop, L., Imtiaz, J., Trsek, H., Wisniewski, L., & Jasperneite, J.** (2013, July). *Using OPC-UA for the autoconfiguration of real-time ethernet systems.* In 2013 11th IEEE International Conference on Industrial Informatics (INDIN) (pp. 248–253). IEEE.

17. **Reinhart, G., Krug, S., Hüttner, S., Mari, Z., Riedelbauch, F., & Schlögel, M.** (2010, November). *Automatic configuration (plug & produce) of industrial ethernet networks*. In Industry Applications (INDUSCON), 2010 9th IEEE/IAS International Conference on (pp. 1-6). IEEE.

18. **Hammerstingl, V., & Reinhart, G.** (2015, March). *Unified Plug&Produce architecture for automatic integration of field devices in industrial environments.* In Industrial Technology (ICIT), 2015 IEEE International Conference on (pp. 1956–1963). IEEE.

19. **Jasperneite, J., Hinrichsen, S., & Niggemann, O.** (2015). *Plug-and-Produce "für Fertigungssysteme.* Informatik-Spektrum, 38(3), 183–190.

20. https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.html

21. https://github.com/acplt/openAAS/blob/master/UML/AAS_Structure.pdf

Annex B

Adaptable Factory Requirements

Figure 30: Adaptable Factory Initial Requirements

ID	Description	RAMI			Candidate Standards	Standardization Gap
		RAMI Layers	RAMI Value Stream	RAMI Hier-achy		
#1	The system shall make devices made operational "in a short time span".	Int.,C	I-P	FD, CD, S	Not needed	None
#2	The system shall allow easy reaction towards improved manufacturing skills and process changes.	Int, C, Inf, F	I-P	S	DIN 8580	Manufacturing skill models (ontologies?), process models, negotiation services
#3	The system shall enable workpieces to specify the pro-duction procedure applied to them.	Int, C, Inf	I-P, T-D	P	RFID, VDI/ VDE 3682	Unknown
#4	The system shall enable production resources to determine constraints to their production procedures.	Inf, F	I-P	FD, CD, S	?	Constraints to production pro-cedures, QoS properties
#6	The system shall allow to scale a manufacturing line largely automated regarding production output.	A, Int	I-P	S	?	Standardized physical connec-tors for production modules
#9	The system shall be composed out of modules that are intelligent.	C, Inf, F	I-P	FD, CD, S	?	Standardized belief-de-sire-intention models?
#10	The system shall be composed out of modules that are interoperable.	C, Inf, F	I-P	FD, CD, S	IEC 62541 (OPC	None, OPC UA sufficiently expressive.
#11	Each production module shall include a self-descrip-tion that allows fast and robust re-configuration of the production line.	Inf	I-P	FD, CD, S	UA), IEC 62769 (FDI), NAMUR MTP	Manufacturing skill models, more comprehensive prop-erty catalogs

Figure 30: Adaptable Factory Initial Requirements (continued)

			RAMI			
#12	Each newly connected field device shall receive network connectivity without manual intervention.	Int, C	I-P	FD, CD, S	eCl@ss, IEC 62541-1	None
#13	Each newly connected field device shall be advertised to all connected system parts.	Int, C, Inf	I-P	FD, CD, S	RFC 3927	May require extentions to fieldbus
#15	The system shall detect the necessary control-required and software- required modifications upon connecting a new field device.	Int, C, Inf	I-P	FD, CD, S	(Zeroconf)? IEC 62541-12	standards?
#16	The system shall propagate the necessary control-required and software- required modifications upon connecting a new field device to all relevant system parts.	Int, C, Inf	I-P	FD, CD, S	IEC 62541, IEC 62769	?
#17	The system shall allow moving software components for process control between decentralized control units, while adhering to constraints such as production output and availability.	Int, C, Inf	I-P	FD, CD, S	IEC 61131, IEC 61499	?
#18	The system shall allow automatic creation of visualizations for device parameters.	Inf	I-P	FD, CD, S	IEC 62769, NAMUR MTP	FDI sufficient?
#19	The system shall allow engineers to execute a modular engineering, where libraries of reusable modules are used.	Inf	T-D	FD, CD, S	IEC 62714 (AutomationML), NAMUR MTP	Standardized control applications?
#20	The system shall provide vendor-independent services for archiving.	F	I-P	FD, CD, S	IEC 62541-11	None
#21	The system shall provide vendor-independent services for alarm management.	F	I-P	FD, CD, S	IEC 62541-9	None
#22	The system shall provide vendor-independent services for visualization.	F	I-P	FD, CD, S	IEC NAMUR MTP	?
#23	The system shall provide vendor-independent services for integrating MES functions.	F	I-P	FD, CD, S	IEC 62264, ISO 22400?	?

AUTHORS

Dr. Heinz Bedenbender | Alexander Bentkus | Prof. Dr. Ulrich Epple | Dr. Thomas Hadlich | Roland Heidel | Oliver Hillermeier | Dr. Michael Hoffmeister | Haimo Huhle | Markus Kiele-Dunsche | Dr. Heiko Koziolek | Dr. Steffen Lohmann | Marco Mendes | Dr. Jörg Neidig | Florian Palm | Stefan Pollmeier | Benedikt Rauscher | Frank Schewe | Bernd Waser | Ingo Weber | Prof. Dr. Martin Wollschlaeger

This working paper has been elaborated in the working group “Modelle und Standards” of the ZVEI together with the working group on “Reference architectures, standards and norms” (Plattform Industrie 4.0).

www.plattform-i40.de