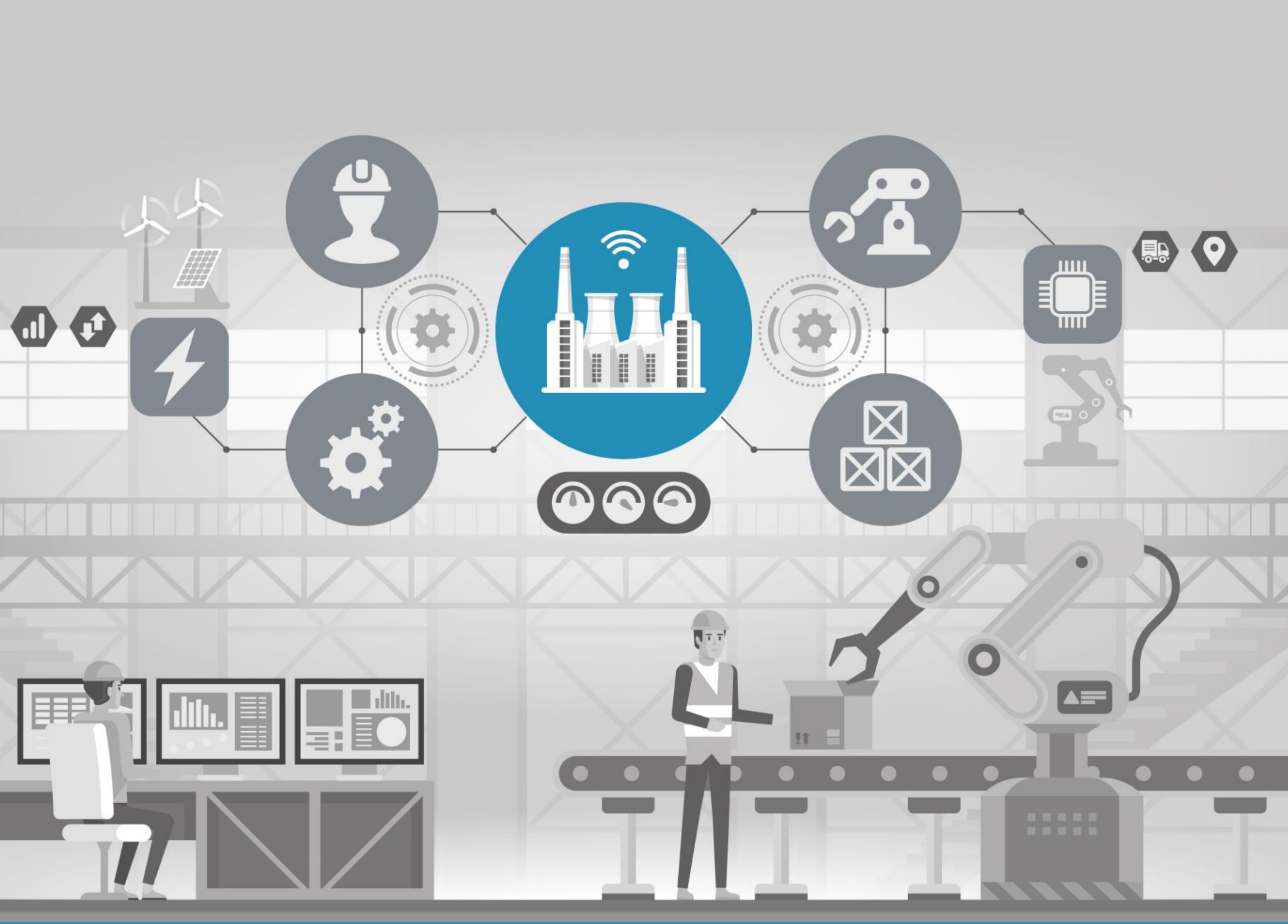


IIP-Ecosphere Whitepaper

IIP-Ecosphere Platform – Requirements (Functional and Quality View)

Version 1.0

Holger Eichelberger, Christian Sauer,
Amir Shayan Ahmadian, Michael Schicktanz,
Andreas Dewes, Gregory Palmer, Claudia Niederée

White Paper IIP-2021/002-en

IIP-Ecosphere
Next Level Ecosphere for Intelligent Industrial Production

Disclaimer

Publication: March, 2021 on https://www.iip-ecosphere.eu/

DOI: 10.5281/zenodo.4485774

Gefördert durch:

Bundesministerium
für Wirtschaft
und Energie

Executive Summary

This document elaborates the requirements for the IIP-Ecosphere platform. The requirements are based on discussions with the partners and the (representatives of the) work packages or subprojects of IIP-Ecosphere, such as the demonstrators. An overview of current Industry 4.0 platforms created in the project was included as a basis, as well as a requirements survey from a user perspective.

The document comprises 141 requirements with a total of 181 sub-requirements in 14 topic areas and now serves to derive a common architecture as well as the realization of the IIP-Ecosphere platform. It is planned to update this document as new requirements emerge in order to serve as a basis for the IIP-Ecosphere ecosystem as well as for further platform developments.

Inhaltsverzeichnis

1 Introduction

1.1 Motivation and Goals

The digitalization of industry and even higher interlinking of entities are increasing the performance and complexity of technical systems and the associated processes. In the industrial environment, intelligent production (Industry 4.0) is considered to have high innovation potential, with artificial intelligence (AI) as a key technology. The vision of the IIP-Ecosphere project, which is funded in the AI Innovation Competition of the German Federal Ministry for Economic Affairs and Energy (BMWi), is to achieve an innovative leap in the field of self-optimization of production based on networked, intelligent, autonomous systems to increase productivity, flexibility, robustness and efficiency. The goal is to build a novel ecosystem - the Next Level Ecosphere for Intelligent Industrial Production (IIP-Ecosphere) - that enables a "next level" in intelligent production.

One of the activities for implementing this vision is the realization of a cross-company virtual platform that connects existing devices and factory installations in a manufacturer-independent manner and also provides AI on these installations in a flexible manner. In addition to the low-threshold application of AI processes, AI processes should be provided as optimal as possible on manufacturing devices and be linkable with platform services, such as storage services or factory-wide data integration. The platform should also enable data sharing and provide the necessary security mechanisms.

This document is a step towards the realization of the IIP-Ecosphere platform. This document describes the vision of the platform services to be realized, their differentiation from existing platforms and, in particular, the requirements for the platform to be realized. Vision and requirements have been identified in intensive discussions with the partners of the project. The concepts and requirements will be discussed and evaluated with further stakeholders within the community dialog of the IIP-Ecosphere project. This may intentionally lead to an updated version of this document.

1.2 Interaction with other Initiatives

Existing platforms for industrial production as well as individual (innovative) technical activities may overlap with our work. However, overlaps also provide room for interaction, synergy and coordination with other initiatives and activities. In this document, we take this in account as follows:

- Industry 4.0 **standards and specifications** enable the interaction of Industry 4.0 platforms. The IIP-Ecosphere platform will consider, implement and support standards in this area such as management shells [26, 40], RAMI 4.0 [31], Industry 4.0 Language [27, 38] OPC UA [22] or OneM2M [21].
- **Other Industry 4.0 platforms** may provide similar or complementary functionalities. The IIP-Ecosphere platform is therefore designed in particular as an open and virtual platform that can integrate with other platforms and mutually utilize capabilities (where available). For this purpose, our planning takes into account the overview of industrial platforms [33] developed in IIP-Ecosphere.
- Important **functionalities** already exist, especially as **open source**, e.g., the realization of management shells in the BaSyS/BaSyX project [1, 6] or the implementation of protocols or services in the Eclipse IoT ecosystem [8]. These functionalities shall not to be developed again rather than adequately integrated.
- **Conceptual collaboration** with the goal of standardization, e.g., with the German Machine Tool Builders' Association (VDW) as a partner in IIP-Ecosphere or other initiatives or funded projects in the area of Industry 4.0, e.g., Plattform Industrie 4.0 [25], BaSyS [1] or OneM2M [21] or “VWS vernetzt” [41].

1.3 Structure of this Document

This document is structured as follows: In the next chapter, we provide an overview of (plans for) the IIP-Ecosphere platform. In Chapter 3, we describe the requirements for the platform along the concepts introduced in the overview in Chapter 2. In Chapter 4, we discuss how we aligned the requirements in this document to the activities in [36]. In Chapter 5 we elaborate on a prioritization of the requirements as well as identified requirement conflicts. Chapter 6 summarizes this document and in Chapter 7 we enumerate referenced work. Appendix 1 contains the original requirements request (slightly adjusted in language and or formatting) for the IIP-Ecosphere project parts, especially the demonstrators.

2 Overview of the IIP-Ecosphere Platform

In this chapter, we provide a brief overview of the IIP-Ecosphere platform, both in terms of the envisioned platform services and through an initial overview architecture. This overview aims at motivating and explaining the structure of the following requirements.

Figure 1 visualizes (as a kind of mind map) the services to be provided by the IIP-Ecosphere platform. This vision was developed by the cross-sectional working group "Architecture" in IIP-Ecosphere. This group is mainly composed of the partners of the Think Tank "Platforms"[1] (University of Hildesheim/Software Systems Engineering, University of Koblenz-Landau) and the "AI Accelerator" (Siemens, AI-Protect, Lenze, Bitmotec), but also includes interested or associated partners like Phoenix Contact.

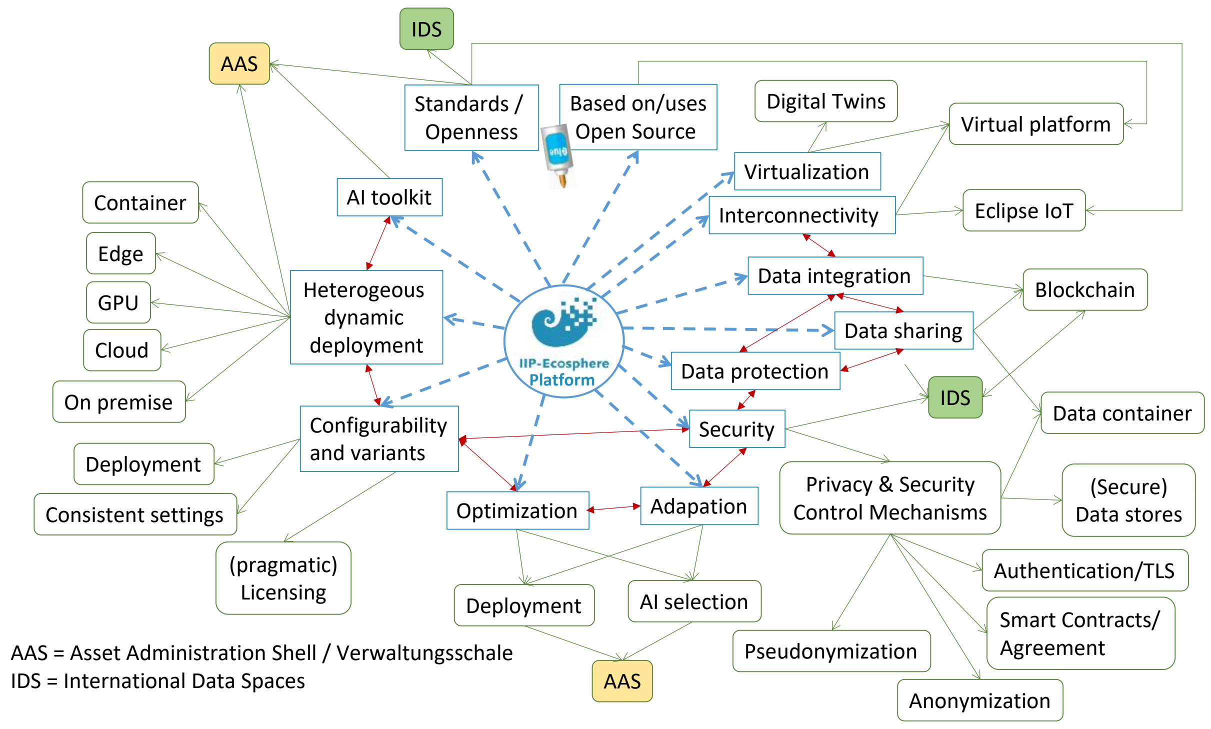


Figure 1: Mind map overview of the envisioned platform services and functionalities.

The IIP-Ecosphere platform consists of (counterclockwise in Figure 1)

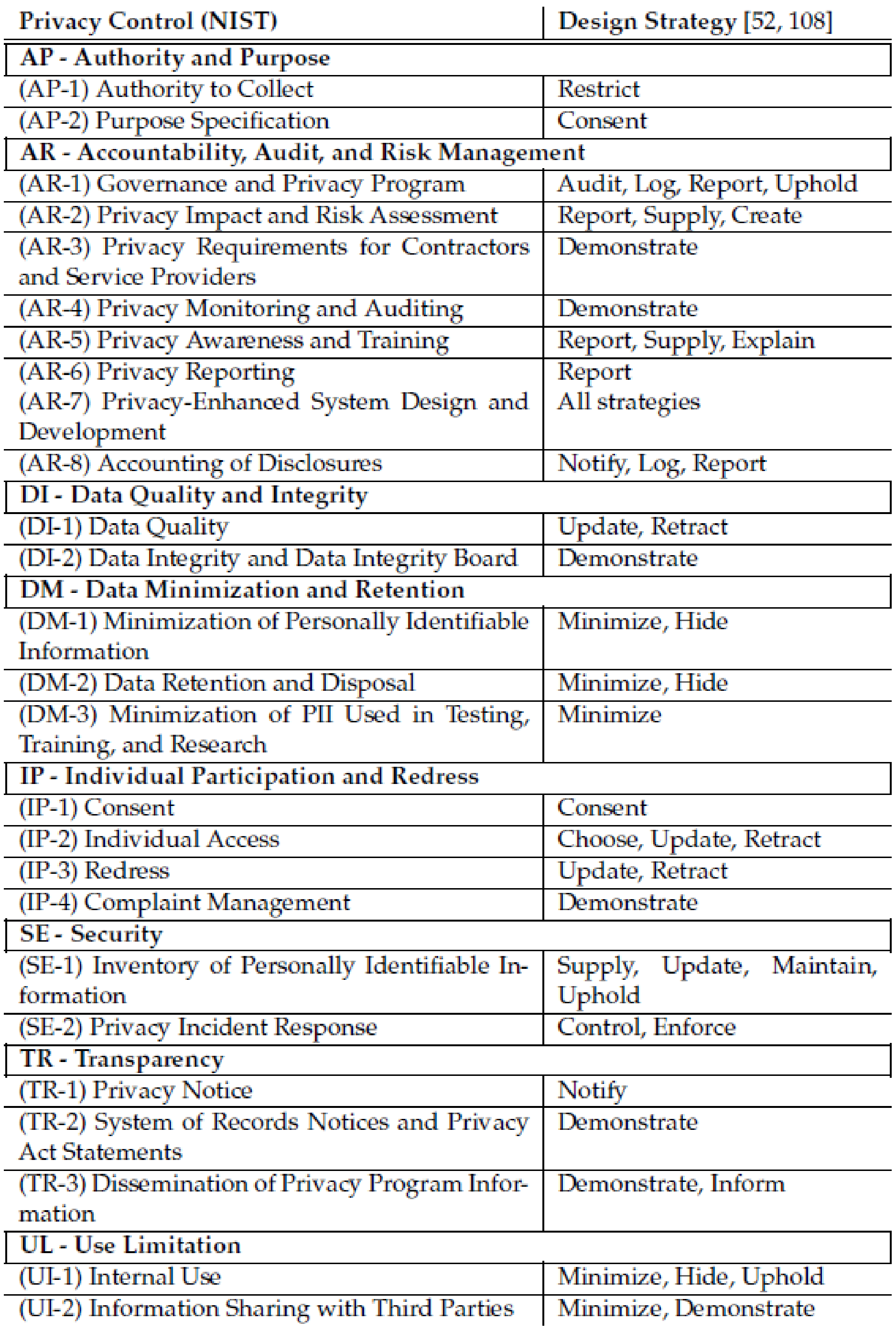
- An open and extensible **AI toolkit** that contains reusable AI building blocks in the form of services or components. Here, the platform will provide interfaces and mechanisms to distribute AI services to **resources** and to provide them with data or to make the data available to subsequent components - both for training and for runtime/prediction. The concrete building blocks and their specific interfaces (desirably structured according to AI processes), which are based on the platform interfaces for services and data, will be developed in particular in collaboration the demonstrators and the "AI Accelerator". The principles of the interfaces, which are represented by individual Asset Administration Shells (AAS) [26], will be defining the architecture of the platform.
- A mechanism for **heterogeneous, dynamic deployment** of selected platform functionalities such as AI services to different resources. Here, the platform decides on the services or components to be deployed based on information such as assurances and resource requirements. This information will be taken from the services/components, which describe

[1] For a description of the project structures, please refer to http://iip-ecosphere.eu/

themselves in form of AAS. Similarly, the potential target resources, such as edge devices or GPU servers, are described in form of AAS. Containers (e.g., Docker containers [5]) will be used as deployment units. Primarily, the IIP-Ecosphere platform targets **local/on-premise** installations. A **cloud-based** execution of the containers is optional and shall be considered as part of the architecture derivation. Even though we basically follow the LNI recommendations for edge deployment [20], IIP-Ecosphere goes beyond current proposals with these three deployment levels (edge, on-premise server and cloud) as detailed in [36].

- For systematic and consistent **configurability** of the platform, the IIP-Ecosphere platform will use **systematic software variation techniques**. Such approaches allow determining whether the respective platform configuration is consistent and executable before the platform is executed, thus, simplifying installation and maintenance tasks. Since the configuration also describes the solutions/applications running on the platform and their required services/components, the configuration also enables application-driven construction of the deployment units. For this purpose, we will utilize EASy-Producer [9], a flexible software product line tool that can model such complex configurations, check their consistency/validity, and implement software customizations through generative methods including mechanisms for controlling runtime adaptation.
- Configuration modeling/configurability will be supplemented by (dynamic) information extracted from the AAS of the deployed components. This enables **optimization** of the heterogeneous deployment with respect to the available resources, but also of the concrete deployment of AI components in the containers. The optimization initially will focus on an assignment of components to resources during application startup based on respective (dynamic) information provided by the AAS of the involved resources.
- Moreover, an optimization of the deployment and the used AI services/components can also take place at runtime, i.e., lead to a form of **adaptivity** of the platform. For this purpose, suitable mechanisms for the derivation of adaptation decisions, for the runtime adaptation of
- **Data protection mechanisms:** Article 25 GDPR (data protection by design and by default, PbD - Privacy by Design) requires that appropriate technical and organizational measures are integrated into a system, both at the time of determining the means for processing and at the time of the actual processing. Examples for such mechanisms are security & privacy controls [17] designed to effectively implement data protection principles such as data minimization and to incorporate the necessary safeguards into the processing).

The technical measures (e.g., NIST Controls, see Figure 2) are too generic to be integrated directly into a system model.



Privacy Control (NIST)	Design Strategy [52, 108]
AP - Authority and Purpose	
(AP-1) Authority to Collect	Restrict
(AP-2) Purpose Specification	Consent
AR - Accountability, Audit, and Risk Management	
(AR-1) Governance and Privacy Program	Audit, Log, Report, Uphold
(AR-2) Privacy Impact and Risk Assessment	Report, Supply, Create
(AR-3) Privacy Requirements for Contractors and Service Providers	Demonstrate
(AR-4) Privacy Monitoring and Auditing	Demonstrate
(AR-5) Privacy Awareness and Training	Report, Supply, Explain
(AR-6) Privacy Reporting	Report
(AR-7) Privacy-Enhanced System Design and Development	All strategies
(AR-8) Accounting of Disclosures	Notify, Log, Report
DI - Data Quality and Integrity	
(DI-1) Data Quality	Update, Retract
(DI-2) Data Integrity and Data Integrity Board	Demonstrate
DM - Data Minimization and Retention	
(DM-1) Minimization of Personally Identifiable Information	Minimize, Hide
(DM-2) Data Retention and Disposal	Minimize, Hide
(DM-3) Minimization of PII Used in Testing, Training, and Research	Minimize
IP - Individual Participation and Redress	
(IP-1) Consent	Consent
(IP-2) Individual Access	Choose, Update, Retract
(IP-3) Redress	Update, Retract
(IP-4) Complaint Management	Demonstrate
SE - Security	
(SE-1) Inventory of Personally Identifiable Information	Supply, Update, Maintain, Uphold
(SE-2) Privacy Incident Response	Control, Enforce
TR - Transparency	
(TR-1) Privacy Notice	Notify
(TR-2) System of Records Notices and Privacy Act Statements	Demonstrate
(TR-3) Dissemination of Privacy Program Information	Demonstrate, Inform
UL - Use Limitation	
(UI-1) Internal Use	Minimize, Hide, Uphold
(UI-2) Information Sharing with Third Parties	Minimize, Demonstrate

Figure 2: Relations between data protection strategies and NIST privacy controls.

We use a selection of eight existing privacy design strategies (Minimise, Hide, Separate, Aggregate, Inform, Control, Enforce, Demonstrate), including their concrete specifications using sub-strategies, privacy design patterns, and privacy enhancing technologies (PETs), which simplifies the realization of the policies. We use feature modeling to show different mappings between strategies, patterns, and PETs. Feature modeling allows us to capture variability in a system in terms of features and relationships between them. An excerpt from the resulting feature model is presented in Figure 3.

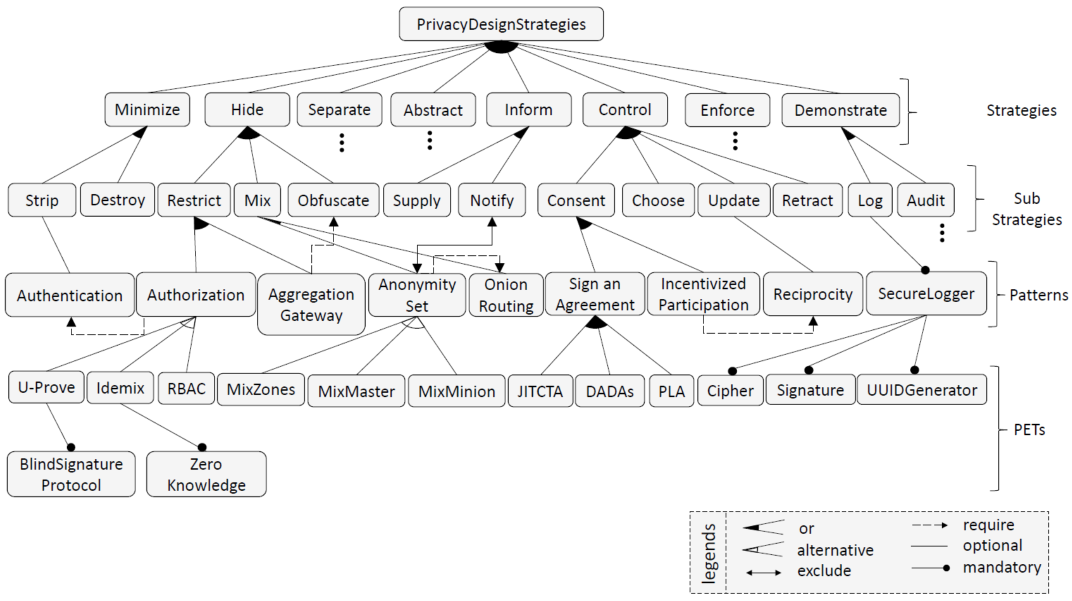


Figure 3: Excerpt from a feature model for privacy design strategies.

- Services and components for **data integration** from heterogeneous sources. The development of holistic models to increase the effectiveness and efficiency of production requires a systematic integration of data from different disciplines such as engineering, production planning or manufacturing. It is important to note that the nature of the data varies greatly based on the sources to be integrated. For example, streaming-heavy data from the OT domain must be combined with semantic data from manufacturing planning. After data integration, the data must be made available for solutions/applications and AI algorithms in a suitable structure, granularity and speed.
- IIP-Ecosphere aims to promote and support the exchange of data (**data sharing**) between partners in the ecosystem. To this end, the platform (in consultation with the Think Tank "Data") is to provide mechanisms that enable data sharing while taking data protection and data security into account, e.g., through smart contracts or data usage agreements.
- Services for the **integration of data** at runtime, i.e., the combination and integration of machine and production data with other production-relevant information such as order data or design plans. This data can in turn be made available to artificial intelligence to improve the decisions proposed there.
- Protocols for **interconnectivity** at various levels, whose data should in principle, be made available to all components of the platform. These include protocols or communication frameworks from the Industry 4.0 area that enable connection to machines and already installed platforms ("southbound" interconnectivity), such as OPC UA [22], protocols for connecting IIP-Ecosphere platforms (horizontal interconnectivity) as well as the connection of control systems (SCADA, MES) or visualizations that can benefit from AI processes through the functionality of the IIP-Ecosphere platform ("northbound" interconnectivity). Components from Eclipse IoT [8] in particular are intended for this purpose.
- **Virtualization** of the platform and its components. First, the IIP-Ecosphere platform is designed as a virtual platform, i.e., it builds on existing services, protocols, and platform installations and complements and extends them. The goal is not to replace existing platforms as studied in [33], but to complement them appropriately. Moreover, the platform itself is virtualized, from the containers of the dynamic and heterogeneous deployment to the central building blocks of the platform. Furthermore, the IIP-Ecosphere platform is intended to integrate **digital twins** [37], whether by providing suitable frameworks, by managing deployed containers, or

by connecting to installed (virtual) machines. In this context, the digital twins to be used must be made known as AAS to the platform.

The core functionality of the IIP-Ecosphere platform described above differs in many aspects from existing platforms [33], in particular through the combination of on-premise installation, heterogeneous deployment to a wide variety of resources including edge devices, an extensible AI toolkit, optimization and adaptation of services, stream-based data integration, and innovative security and privacy mechanisms. However, the basic functionality outlined is also very extensive. Therefore, partners will rely on existing **open source** components, so the platform will contribute in many places, especially the connections between these components (glue code) and other functionality. We expect that integration difficulties between different components can be avoided by use of container technology and that even additional flexibility can be achieved beyond that.

To structure the following requirements into thematic areas, Figure 4 presents an initial overview architecture covering the services and capabilities described above. This overview architecture will be refined into an implementable detailed architecture in the following work steps and discussions with the partners.

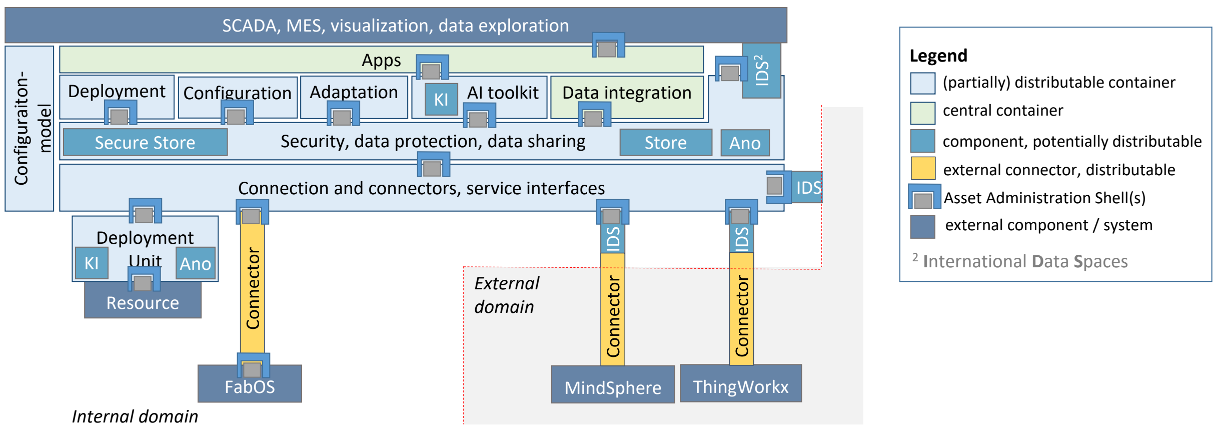

SCADA, MES, visualization, data exploration
Configuraiton-model
Apps
Deployment
Configuration
Adaptation
KI
AI toolkit
Data integration
IDS²
Secure Store
Security, data protection, data sharing
Store
Ano
Connection and connectors, service interfaces
IDS
Deployment Unit
KI
Ano
Resource
Connector
FabOS
Internal domain
External domain
IDS
Connector
MindSphere
IDS
Connector
ThingWorkx
Legend
(partially) distributable container
central container
component, potentially distributable
external connector, distributable
Asset Administration Shell(s)
external component / system
² International Data Spaces

Figure 4: Initial overview architecture of the IIP-Ecosphere platform.

Active resources are integrated into the IIP-Ecosphere platform through interfaces and **deployment units** (containers). The interfaces declare both resource properties and functionality in the form of management shells. The containers contain the functionality tailored to a resource. Machines or factories can be directly connected to the platform by means of (reusable) **connectors** – optionally using IDS[2] functionality, depending on the security domain. The **connection and connectors layer** manages the connectors and can also enable connections between multiple installed instances of the platform at this level. If necessary, parts of the connection and connector layer as well as connectors themselves can become part of a deployment unit, e.g., to communicate directly with the resource via protocols such as MQTT or information models/communication frameworks like OPC UA. Furthermore, this layer defines the basic service interfaces.

The **security, privacy and data sharing** layer[3] manages authorizations and provides additional components such as (secure) data stores or anonymization (**Ano** in Figure 4) or pseudonymization components. While data stores are more like central services, anonymization and pseudonymization

[2] International Data Spaces [13]

[3] Usually, security and data protection cross-cut multiple layers. We discuss here only the specific functions for IIP-Ecosphere rather than e.g., the relations to the connectors.

[3] Akin to security, a consistent configuration approach is crosscutting and the related model reaches across multiple levels of an architecture. We show here only the component a user may interact with to alter the platform configuration.

components are distributable and can be used even in the deployment units. An example from the IIP-Ecosphere demonstrators for anonymization is the creation of worker profiles in the manufacturing process to support machine parameterization. The question here is how to anonymize this personal data. This layer will also be able to control data accesses to AAS/the connection and connector level through appropriate mechanisms. Furthermore, this layer also provides secured external protocols, e.g. for the „northbound“ vertical Integration on platform side.

The **deployment** is responsible for creating and deploying customized deployment units. This is done using the **configuration model** / the **configuration** component. The **adaptation** (implicitly also the optimization) uses the runtime information, which is reflected from AAS into the configuration model and adapts the runtime part of the configuration in such a way that the Deployment and/or the used AI services are used optimally. To enforce this, appropriate AAS interfaces of the deployment units and resources are used.

The **data integration** links additional information such as blueprints with the production data. The extended data models are available to the platform, in particular to the AI services (and, through deployment, also to the associated resources, taking security requirements into account). The models created in the process can be stored in suitable data stores. In course of detailing the architecture, (parts of) the data integration may become distributable, i.e., they may become available on edge devices.

The **AI toolkit** provides interfaces for (distributable) **AI** services and implements or integrates example AI services (in particular via the "AI Accelerator" work package). AI services with increased resource requirements, e.g., for training, for example can be distributed and executed on GPU servers, while AI services with low resource requirements or high-speed capabilities can be executed on edge resources. An example from the IIP-Ecosphere is that forward propagation of a neural network may be done on an edge device and while backward propagation for training must be done on a suitable central resource, such as a GPU server. AI services can be added and are then distributable analogously to the services/components already supplied.

Using the configuration model, the services and components of the platform can be assembled to solutions or applications (**Apps**) and, if necessary, supplemented with additional program code. An application can define its own application-specific services. The applications to be executed determine the required AI and platform components (such as the need for anonymization close to on manufacturing machines) and, thus, define the basis for deployment, optimization and adaptation.

External interfaces can be used to communicate IIP-Ecosphere platform instances, to ingest further data or to control the manufacturing process. The external components/systems can access AI services, platform services and data of an IIP-Ecosphere platform (taking into account the security requirements), modify them within limits (e.g. to start a production order) or visualize them.

3 Requirements

We formulate the requirements for the IIP-Ecosphere platform according to the idea of **controlled textual requirements** or **requirement templates** [32]. A requirement is described with a (preferably) simple sentence. We use (analogously) the following pattern

> *<condition>?*
>
> *<actor/component>*
>
> *<classification>*
>
> (provides | contains | has | is | …)
>
> *<whom>?*
>
> *<functionality or property > <source>*?

Elements in angle brackets like *<actor/component>* shall be substituted accordingly. Elements with a "?" at the end are optional, while the middle line indicates alternative sentence constructions. After the end of the sentence, the source of the requirement should be given as an abbreviation for traceability.

As a classification we use "must" (indicating highest priority), "shall" (for medium priority, desirable) and "can" (for low priority, nice-to-have). In addition to this classification, more detailed prioritization is required. We give an initial prioritization of the requirements in Chapter 4.

Requirements that logically consist of several parts are described by a main requirement as well as one or more sub-items (each again as a controlled textual requirement). If necessary, a requirement can be described in more detail by additional sentences (marked by "*Explanation:*").

We define necessary terms in Section 3.1, define sources of requirements and their abbreviations in Section 3.2, discuss general requirements in Section 3.3, and discuss then requirements for the concepts from Figure 4 in respective subsections (following the description order there). However, due to strange formatting behavior in Microsoft Word, not all references to sub-requirements might be correct, in particular cross-references with higher identifier number.

3.1 Terminology

We summarize now the terms used below and their meanings:

- In industrial production, the devices and networks used are often distinguished into two areas [15], which are
  - Operational Technology (**OT**) for production control and monitoring, especially consisting of production machines, control devices, edge devices, etc.
  - Information Technology (**IT**) for the "higher level" control of production using common devices, such as servers or cloud technology.
- Each asset in the Industry 4.0 context shall be represented by a so-called **Asset Administration Shell**, which is suitable for describing the asset minimally but sufficiently with respect to the Industry 4.0 use cases [40]. AAS can represent functional properties, interfaces as well as quality properties of the described assets.
- A **configuration model** allows for describing configuration possibilities and dependencies for software or hardware. A configuration model defines the space of all configuration possibilities and describes thereby the possible instances of a software and/or a hardware. A configuration

describes a single instance. A **configuration** is valid, if all dependencies are fulfilled. Such checks can be realized by model analyses such as constraint reasoning. A configuration can be **instantiated** (often automatically), i.e., the underlying software/hardware is customized, e.g., by omitting alternatives, so that the configured instance is realized. Adjustments can be possible **before run-time** (these are then partly irreversible) as well as **at run-time**. Both types may even be interlinked. Examples of configuration models from the area of **Software Product Lines** (SPL) are feature models [3, 18], see Figure 3 for an example, decision models [34] or also topological models [9].

- A **virtual platform** builds on existing services and platforms and integrates them to offer further, higher-value services. The IIP-Ecosphere platform shall be a virtual platform to link the existing services of existing Industry 4.0 platforms avoiding the need to develop them again.
- Below, we use the term "**platform**" as shortcut for the virtual IIP-Ecosphere platform.
- We use the term "**service**" synonymously for a "**component**" with an explicit, defined interface.
- A **platform service/component** generally refers to all mentioned components or component types of the IIP-Ecosphere platform, in particular (distributable) AI services, security components, connectors, etc.
- A pseudonymization replaces a name or another identifier by a pseudonym (usually a code consisting of multi-digit combinations of letters or numbers) to exclude or at least make it significantly more difficult to determine the identity of the data subject [39].

3.2 Sources for Requirements

The following stakeholders and documents were included as sources for requirements:

- The **funding grant** for IIP-Ecosphere, in particular the project concept with the initial reference architecture of the platform, as well as the individual sub-project descriptions.
- **Discussions** with various **sub-projects of IIP-Ecosphere** that can be understood as users of the platform. These include in particular the four IIP-Ecosphere demonstrators and the experimental field, but also the Think Tanks "Data" and "AI & Production", respectively. The cross-section group “Architecture” developed a questionnaire to query the needs (especially of the demonstrators). The results were integrated into the requirements and are marked accordingly. The questionnaire is displayed in Chapter 1.
- **Discussions** of the **cross-section group "Architecture"** in IIP-Ecosphere. The cross-section group brings together partners with a particular interest in platform topics, in particular the partners from the Think Tank "Platforms" and partners from the “AI Accelerator”. This group has developed the shared vision of platform services (see Section 2), the overview of on current Industry 4.0 platforms [33] as well as the precursor usage view requirements [36]. Furthermore, the basic requirements in this document were collected and discussed in the cross-section group.
- **Standards and legal requirements** can define or influence requirements.
- The in Section 1.2 mentioned **overview of current Industry 4.0 platforms** [33] serves in particular to substantiate, validate and delimit the requirements.
- The **usage view requirements** [36] are a direct basis for the requirements.
- **Discussions** of the requirements with **other stakeholders**, e.g. the Industrial Advisory Board of IIP-Ecosphere or external partners within the framework of the "Platforms" working group.

In the following, we mark the origin of individual requirements with:

Source	Marker
Grant agreement, contracts	[A]
Demonstrators	[D]
Think Tanks	[T]
Experimentation field	[E]
Cross-section group „Architecture“	[Q]
Standards, legal requirements	[S]
Plattform overview [33]	[P]
IIP-Ecosphere Industrial Advisory Board (IAB)	[I]
Usage View [36], see also chapter 4	[U]
IIP-Ecosphere partner (proactive, e.g., after meeting)	[V]
IIP-Ecosphere external work groups, project competition phase, external discussions	[X]

Multiple labeling is possible if a requirement originates from different sources.

3.3 General Requirements

In this section, we list general and overarching requirements for the platform.

R1. The platform **must** be vendor-independent or technology-neutral. [A]

R2. The platform **must** be standards-oriented. [A]

R3. The IIP Ecosphere platform **must** be realized as a virtual platform. [A]
Explanation: A virtual platform integrates existing services/platforms to provide higher value services, especially AI-based services.

R4. The platform **shall** be designed and implemented in a component/service-oriented manner. [Q, D]

a. Components/services **must** be described with their interface (input, output). [Q, U]
b. Components/services **must** be equipped with meta-information (version, categorization). [T, U]
c. Components/services **must** have a queryable state. [U]
d. The execution of services **must** be supervised (monitoring). [U]
e. Service monitoring **shall** be parameterizable. [U]
f. Service monitoring **shall** be realized by application-specific services. [U]
g. Monitored values **shall** be checked and reported by triggers. [U]

Explanation: A modular structure eases integration into existing systems. The state of a component/service signals, e.g., "in operation", "passive", "error" or "in adaptation". Monitoring can be implemented by different realization techniques like code injection, samples or specific services (R4.e). Values should be checked using standardized (R4.e) or application-specific (R4.f) mechanisms, and, if necessary, forwarded to interested parties so that action can be taken (see also R118.a as well as Sections 3.12 und 3.14)

R5. The platform **shall** be built up from open source components. [A]
Explanation: In particular, the licensing requirements of IIP-Ecosphere must be taken into account, especially that copy-left OpenSource licenses are considered problematic in the industrial environment and should therefore be avoided.

R6. The platform **shall** be able to integrate further optional components that do not correspond to R1 or R4. [Q]
Explanation: Examples are commercial components of the partners.

R7. Components/container interfaces as well as their capabilities **must** be described by AAS. [Q, U]

a. The description concepts for AAS **shall** be as uniform as possible. [D]

Explanation: The description concepts should be coordinated with the stakeholders, in particular the demonstrators. Use of standards or standardization for the description concepts is desirable.

Explanation: Where possible, AAS shall also be used for component/service communication. However, there might be frequency/volume reasons to resort to an uniform secondary approach (see e.g. R70, which then shall be based on Industry 4.0 communication protocols (see R14).

R8. In the platform, SPL approaches **shall** be used for variant management. [A]

a. The platform **must** contain an integrated configuration model for applications, services and platform properties. [A, U]

b. The SPL approaches **must** allow for automated validation of the configuration model. [A, U]

c. The SPL approaches **must** support automated derivation of platform instances. [A]

Explanation: SPL approaches shall be used where appropriate or scientifically challenging, e.g., to systematically realize the integration of optional components or to be able to completely remove them from a platform instance. The underlying configuration model (R8.a) shall be validated automatically (R8.b), usually taking into account application-specific conditions/constraints, and the model shall be instantiated automatically (R8.c).

R9. The platform **shall** ensure the availability of its services. [Q]

Explanation: This includes in particular the production-related (possibly autonomous) execution of complex programs (industrial PCs, edge devices).

R10. The production-critical platform operations **shall** meet the requirement of soft real-time (response time < 100ms). [T, Q]

Explanation: Safety-critical hard real-time requirements are out of scope, especially in the IT domain. However, the platform can generally only meet soft real-time requirements for the operations it provides/it is responsible for. For external extension functions, components or services that are based on interfaces of the platform, this is either not possible because they are external or meeting such requirements is only possible by adapting or, in extreme cases, terminating the respective component or service.

R11. The platform **must** provide documentation of its provided functions. [Q]

a. The platform **must** provide a developer documentation.

b. The platform **must** provide a user documentation.

c. The platform **must** provide a documentation of the execution.

Explanation: This includes documentation for administrators and operators (R11.b) such as a user manual, as well as documentation for programmers (e.g. code documentation, R11.a). AAS shall be suitably documented and, thus, represent (partially) machine-readable documentation. Furthermore, AAS shall reference the appropriate documentation. Various operations, in particular changes to resource properties, services or applications, must be documented at runtime of the platform (logging, R11.c).

R12. Data processing steps **shall** be documented in a uniform manner. [Q]

a. If sufficient information about the components used is available, the platform **can** automatically derive the documentation from the configuration model. [Q]

Explanation: Such a documentation is required for data protection reasons. The information could be derived automatically, e.g., from the AAS of the involved components/services.

3.4 Connectors and Connections/Interconnectivity

Connectors represent both external and internal connections. Connectors can provide access to data but also transport control information.

R13. The platform **must** allow for connections with other actors. [A]
   a. The platform **must** be able to connect directly to Industry 4.0 devices. [A]
   b. The platform **must** be able to connect with other Industry 4.0 platforms. [A]
   c. The platform **must** be able to connect to other instances of the IIP-Ecosphere platform. [A]
   d. The integration of the mentioned actors **must** be possible at runtime. [D]

   Explanation: R13.b and R13.c are the primary connections for a virtual platform (R3). This implies, that device information (R25.b), resource allocation (Section 3.12) or adaptation (Section 3.14) may occur depending on access permissions/approvals. R13.a reflects the specific focus on edge computing as decided by the partners in terms of the platform vision.

R14. The platform **must** offer connectors to enable the open and flexible integration of Industry 4.0/IoT protocols and (internal) data sources. [A, Q]
   a. The platform **must** support OPC UA (communication framework, information model) and MQTT (protocol). [D]
   b. The platform **can** support TCP/IP. [T]
   c. The platform **can** support Bluetooth LE. [D]

   Explanation: The platform shall not prescribe any fixed protocols, not even for internal transport. Protocols that are assessed as relevant and suitable in the respective application environment (for standardization and licensing reasons) should be made available, but also external protocols or protocols provided by third parties. This also applies to the transport formats if these are not defined by the protocol, e.g., the payload is not defined for MQTT. Furthermore, this also applies to (possibly stream-based) time-critical communication according to R10. R14.b and R14.c were explicitly mentioned by stakeholders, but are listed here as examples for optional/additional protocols.

R15. Connectors for one type of data, e.g., device connections **shall** be designed in uniform manner [Q]

R16. The integration of connectors into the platform **must** be open and extensible. [A]

R17. Connectors **shall** not be tied to other platform functionalities, so that connectors can be distributed as needed. [A]
   a. Connectors **shall** be described in the configuration model. [U]
   b. Connectors **can** be managed by the platform. [U]
   c. Connectors **can** be parameterizable. [U]

   Explanation: Akin to R20, the combination of configuration model and management by the platform results in the possibility of active reuse of connector information.

R18. The security of the connectors **can** be (optionally) ensured and thereby insert security or origin information into the data. [A, Q]

R19. The platform **shall** internally use a minimum of different formats for production data. [Q]
   a. The platform **shall** support (structured) input data from the southbound interface such as measurement results (Float32, Int16/32, Bool32, String, Arrays thereof), time (date, duration), test result, test location, serial number, codes/categories, events, alarms or manufacturing times. [D]

      Explanation: For example, a data set may contain 50 values of different measurement result types in a process step of 20-30 s

   b. The platform **shall** support (structured) input data from the northbound interface such as product, product number, product category, safety stock, average issue, lot size, processing time, setup time, delivery date (planned vs. actual), lead time (planned vs. actual), malfunctions and malfunction frequency with categorization (but without personal reference), machine availabilities (MTTR, percentage). [D]
   c. The platform **shall** offer other data formats such as JSON/XML via Restful API. [D]
   d. The platform **shall** offer (structured) output data on the northbound interface such as optimal sequence, optimal capacity utilization, optimal short-term staff scheduling, alternative processing stations in case of malfunction, error classification or recommended action. [D]
   e. The platform **shall** allow output data per processing cycle of 5 s. [D]
   f. The platform **shall** provide mechanisms for format adaptation or format conversion described in the configuration model. [Q, D]
      Explanation: Other approaches like [21] speak here of a "semantic model" or a "semantic mapping", often relying on ontologies from the field of Semantic Web. Since the formats must be described in the configuration model for validating the processing compatibility of the components, services, or deployment units (see also Section 3.5) and EASy-Producer [9] has very rich constraint and expression languages, we are convinced that efficient translation and adaptation mechanisms or connectors can be derived automatically with the available means and the configuration model.
   g. The platform **shall** provide mechanisms for customization or manipulation of metadata as specified in the configuration model. [T]
      Explanation: When adapting (R19.f) or integrating (Section 3.10) data, it is important to keep the related meta-information (R79) up to date.

   Explanation: A generic data format described by the configuration model (e.g., for individual applications) or derived meta-information would be desirable so that, e.g., security mechanisms can operate on the data. Here, usual modeling approaches for AAS as well as comparable approaches of OPC UA [22] or oneM2M [21] shall be considered. This also applies to the mechanisms for format adaptation in R19.f. Requirements for personal data such as R48 must also be taken into account.

R20. The platform **must** support application-specific data paths. [Q, D]
   a. The data paths **must** be defined in the configuration model. [Q]
   b. The data paths **can** have properties/parameters. [U]
   c. The data paths **shall** be managed by the platform. [U]

   Explanation: Data paths (called „relations" in [36], akin data flows) connect connectors, components and services. Data paths define through which format adaptation, AI, processing or storage services or components data is flowing on its way from sources, e.g. sensors, to actuators or to the application or visualization. Here, data paths define which inputs are prerequisites for processing and which outputs are delivered by processing. Since the applications, data formats, and components/services are described in the configuration model, it is important to describe also the data paths to achieve a consistent model. Inspired by other approaches such as [42], the active management of data paths and, thus, their reuse by the platform makes sense, even if the information should essentially be contained in the configuration model for consistency reasons (R20.a).

R21. The connectors or connections **shall** have low impact on the data throughput. [Q, D]

R22. The platform **shall** allow for a data throughput of 500 GByte per year. [D]
   Explanation: This requirement was mentioned without further details by a demonstrator in the context of connectors. We interpret this as the throughput at the data ingestion without aggregation or filtering. It is important to note that the data throughput for demonstrators

can be orders of magnitude higher, e.g., before and after data integration (R91) or as mentioned in a presentation at 50 ms cycle between 5 GByte and 600 GByte per day depending on the production step.

3.5 Heterogenous, dynamic Deployment

Deploying platform components allows for flexible use of existing and accessible resources (e.g., edge, server, GPU, cloud) by the IIP-Ecosphere platform. Since different resources with different technical characteristics are typically in use, a heterogeneous deployment with (as little as possible technical) dependency to vendor-specific characteristics is required. This leads to an abstraction of the operating system and the relevant functions, and, thus, unifies the execution environment for the deployment units. From a technical point of view, a deployment unit is a container (“ECS runtime” in [36], for edge devices "edge runtime" in [20]). In principle, multiple containers can be executed on a resource (the set of all containers or "edge runtimes" is called "edge infrastructure" in [20]). Deployment is dynamic, i.e., containers shall be created and deployed as automatically as possible by the platform based on the configuration model or application configurations, respectively. Automated deployment can minimize the number of (IIP-Ecosphere) containers per resource, although we can imagine that the abstraction of the execution environment is realized by additional vendor-specific containers that communicate with the IIP-Ecosphere containers. The container creation, deployment, and allocation functions (see also Section 3.12) form the deployment management system (called "deployment" for short in Figure 4, “ECS management system” in [36] or "edge management system" in [20]).

R23. The platform **must** support dynamic deployment units. [Q, A]
Explanation: A modular structure eases the integration into existing systems. Locally, modules should be as lean as possible without large platform overhead. [D]

R24. The platform **must** support deployment of platform components to different types of resources or hardware. [Q, A, U]
Explanation: Local procedures are necessary for process control (OT), as the latency to cloud systems, e.g., is otherwise too large. Production control, on the other hand, can be centralized (IT). [D]

R25. The properties or functionalities of a resource **must** be described as AAS. [Q, U]

   a. The AAS of a resource **shall** be realized by a resource abstraction layer (“ECS runtime” in [36]). [Q, U]
   b. The AAS of available resources **must** be announced to the platform. [Q, U]
   c. The platform **must** manage the available resources. [P, U]
   d. The platform **can** offer procedures to facilitate the management of available resources. [P, U]
   e. The AAS of a resource **must** describe static properties of the resource. [Q]
   f. The AAS of a resource **must** describe dynamic properties of the resource. [Q]
   g. The AAS of a resource **must** contain functions for deploying deployment units. [Q]
   h. The AAS of a resource **shall** contain functions for exchanging deployment units at runtime. [Q]
   Explanation: This allows an adaptation of the behavior. The functionality could already be implicit in R25.a or even not applicable in the production environment due to the overhead.

Explanation: This requirement refers in particular to the "Device Description Store", the "Device Configuration Tool" and the "ECS runtime" in [36], i.e., also to the abstraction of vendor dependencies (R25.a) as well as on/offboarding (R25.a) or device management (R25.b). Common management functions are not listed in this document, i.e., human interactions (acknowledgements), management techniques such as device templates or import functions for "asset data providers" [36] are desirable, but also well covered by existing platforms [33].

Access via connectors to underlying platforms are implied by R13. Static properties in R25.e are, e.g., the operating system and processor architecture (relevant for container selection) or the number of CPU/GPU/TPU cores. Dynamic properties in R25.f are, e.g., the resource utilization, the memory usage or the utilization of CPU/GPU/TPU cores. Updates of the platform/abstraction layer are described in R136.

R26. The platform must support the deployment (R24) to on-premise resources. [Q, A]

R27. The platform shall support the deployment (R24) to connected IIP-Ecosphere platform instances. [A]

Explanation: This allows companies that trust each other to share expensive resources such as GPU clusters. Appropriate security mechanisms must be in place and configured. A business model can regulate the compensation. Accordingly, device management (R25.c, R25.d) must be designed transparently and equipped with appropriate access mechanisms.

R28. The platform can support (optional) deployment (R24) to cloud-based resources. [E]

a. Cloud-based resources **must** be described in the configuration model (as optionalities). [Q, E, A]
b. The platform **can** enable cloud integration with Google Cloud. [D]
c. The platform **can** enable cloud integration with Gaia-X. [Q]

Explanation: A platform instantiation with cloud-based resources switched off must remove all cloud-related dependencies/program code from the platform code to increase user confidence in the platform. Depending on the cloud connection (private, hybrid, public), different performances may be possible, e.g., data transfer close to soft real-time could be possible with a private cloud, while this is rather unlikely with a public cloud. In principle, the millisecond performance requirements mentioned in this document apply (e.g., sample rate 0.2 ms, 8 ms machine clock, 5 s step clock, 25 s process clock), but trade-offs will need to be made when integrating external services/clouds. The goal is that based on the availability of runtime properties of resources and components (R25.h), the platform can make a trade-off among alternative execution resources and select/enforce the most appropriate for the application at hand. Resource abstraction shall also be consistent for cloud integrations ("ECS runtime" in [36], R25.a).

R29. A deployment unit **must** provide an explicit interface in the form of an AAS.

a. The (quality) properties of the deployment unit and its functional interfaces **must** be defined in the AAS. [Q]
b. The AAS of the target resource (R25) **shall** be made available in the AAS of the deployment unit. [Q]
c. The AAS of contained components/services **can** be made available in the management shell of the deployment unit. [Q]

R30. A deployment unit **must** be encapsulated as a container. [Q]

a. The containers at the IT level **must** be technologically uniform. [Q]
b. The containers at OT level **can** be technologically different. [Q]
c. The platform **can** support the integration of external container registries. [U]

Explanation: It may be necessary that, depending on the resource, e.g., a special edge device, a specific container technology must be used. This information shall be stored in the configuration model so that generation of the respective containers is enabled (see R32). Containers are usually created based on container template. These can be provided by the platform, but may also be obtained from external sources (R30.c).

R31. A container **shall** contain only the required components/services. [A]

a. The required components **must** be specified in the configuration model. [Q]

Explanation: These can be, for example, the components/services for an application.

b. Containers **can** contain optional components. [A]

Explanation: This allows for an adaptation of the behavior of the running application or the entire platform and, thus, more than the required components in R31 may be required.

c. Containers **can** dynamically exchange components/services. [A]
Explanation: This enables adaptation of the behavior of the running application or the platform and bypasses R31.b.

R32. A container **can** contain the data or models needed for execution. [T, D]
Explanation: AI models may only be trained and prepared on corresponding IT resources, e.g., with a sufficient number of GPUs. In this case, the executing container should contain the models or have access to the models via the platform and be able to download and update them if necessary (see also R118 for release). In [36] we view models as parameters of services and respective update functionality for the models behind such parameters may be provided. These models may also include PLC code or Matlab/Simulink models, but in particular standardized AI models such as ONNX [23] (see R115). Formats such as ONNX may in turn be executed directly by some devices, which requires appropriate consideration in the device abstraction (R25). Since not all procedures may require data or models, this requirement is categorized as nice-to-have, but for the respective components it should be considered as a must requirement.

R33. A container **can** contain a local data storage. [Q]
Explanation: It may be necessary that individual components require a local database, e.g., for performance reasons. Data can then additionally be written to central data storages. The requirements in Section 3.8 with the exception of R71 (access from all resources) and R75 (cloud-based storage services) apply here.

R34. The creation of containers by the platform **shall** be automated, based on the settings in the configuration model. [A]

a. A model validation **can** be performed before creation or execution to ensure executability. [Q]
b. The platform **can** support externally provided containers (e.g., for digital twins). [E]

Explanation: This requirement shall consistently enable R31. R34.a is based on R8.b.

R35. The edge devices and the containers installed on them **must** allow to read out production data at a sampling rate of approximately 2 ms. [T]
Explanation: At a minimum, the real-time parts of the OT edge devices must meet this requirement and the non-real-time/IT software on the edge devices must not interfere with this requirement. Further, R10 applies.

R36. The platform **shall** enable the configuration of resources. [U]

a. The platform **shall** allow writing configuration settings from resource management to resources. [U]
b. The platform **shall** allow reading configuration settings of resources into resource management. [U]

Explanation: Reading and writing (specific) configuration settings facilitates factory and platform management. The combination of both operations allows devices to be backed up. Examples of settings are network settings, security settings, (technical) settings for data connections, etc. Depending on the realization of the platform, functionality can also be realized by configurations embedded in code/services/applications, which can be efficiently and consistently transformed into code by model-based instantiation (R8.c).

R37. The platform **shall** enable remote maintenance of resources. [U]
Explanation: Remote maintenance is a standard functionality in device management of IIoT platforms [33].

3.6 Security

Security mechanisms as well as secured data sharing procedures are core functionalities of the IIP-Ecosphere platform.

R38. The platform **must** ensure the use of its services only for authorized persons. [Q, P]
Explanation: Authorization can be realized via the mechanisms mentioned in R40. Here, deeper functionality such as the security mechanisms of management shells should be integrated and used.

R39. The platform must **ensure** that (personal) data is only changed by authorized persons. [Q, P]

R40. The platform must provide the usual security mechanisms, such as Role-Based Access Control (RBAC) and Transport Level Security (TLS). [P]
   a. Roles **can** be specified in the configuration model. [Q]
   b. Certificates **can** be specified in the configuration model. [Q]

R41. The security mechanisms **shall** be integrated with common directory services. [P]
Explanation: One example is the Lightweight Directory Access Protocol (LDAP).
   a. Directory services **must** be configured in the configuration model. [Q]
   b. If no directory service is available, a mechanism for managing user accounts **must** be provided. [P]

R42. Further safety mechanisms **must** be configured uniformly via the configuration model. [P]

R43. Safety mechanisms **must** describe their (quality) properties and their callable functions as AAS. [Q]
   a. The AAS of a security mechanism **shall** describe the respective influence on the performance. [A, Q]
   Explanation: This allows the platform to check for (rough) compliance with configured performance targets based on the configuration.
   b. Selected security mechanisms **shall** be deployable as deployment units. [D]

R44. The configuration model **shall** offer IDS-based connectors as optionally configurable. [E]
Explanation: For platform instantiation without IDS connectors, all IDS dependencies must be removed from the platform code to increase confidence in the platform.

3.7 Data Protection

The platform should comply with or support compliance with current data protection requirements.

R45. The platform **must** provide fair and lawful processing of personal data. [Q, S]
Explanation: This requirement is based on the principle of transparency.

R46. In the platform, the collection of personal data **must** be for specified, clear and legitimate purposes. [Q, S]
Explanation: This can be supported by the documentation of the data processing steps (R12).

R47. The platform **shall** avoid the processing of personal data as much as possible. [Q, S]
   a. Applications running on the platform **shall** avoid processing personal data as much as possible. [Q]
   Explanation: This requirement points to the principle of data sparsity. While the platform mechanisms should work with generic data in a uniform format as far as possible (R19), the data of AI applications running on the platform is subject to the control and needs of the user. Therefore, this requirement is intended as an indication and may be governable by an operating agreement (see also R66) if this requirement is not enforceable by platform mechanisms.
   b. Models used on the platform **shall** be protected against privacy attacks. [T]
   Explanation: Research [35] has shown that such attacks can be carried out and applications must be protected against them.

R48. The platform **must** store personal data in a form that permits identification of data subjects for no longer than is necessary for the purposes for which the data is processed. [Q, S, D]
Explanation: This requirement is based on the principle of memory limitation and correlates with R66.

R49. The platform **must** process personal data adequate and relevant to the purpose and limited to what is necessary for the purposes of the processing. [Q, S]

a. Applications running on the Platform **shall** process Personal Data adequate and relevant to the purpose and limited to what is necessary for the purposes of the processing. [Q, S]
Explanation: This requirement points to the principle of data minimization. While the platform mechanisms should work with generic data in a uniform format as far as possible (R19), the data of AI applications running on the platform is subject to the control and need of the user. Therefore, this requirement is intended as an indication and may be governable by an operating agreement (see also R66), if this requirement is not enforceable by platform mechanisms.

R50. The platform **must** identify different categories of personal data processing. [Q, S]
Explanation: This can be done in connection with the documentation of the data processing steps (R9).

R51. The platform **must** ensure the authorization, deletion or blocking of personal data. [Q, S]

R52. The platform **must** store personal data in a structured, common and machine-readable format. [Q, S]
Explanation: This requirement is based on the right to data portability. The realization must be coordinated with R19.

R53. The platform **must** provide a mechanism for notifications regarding rectification, deletion, blocking, leakage. [Q, S]
Explanation: The mechanism, which may be replaceable or extendable, should be in line with the governance rules of the operator of the platform instance. If the platform is only used within a company, this can be regulated by a company agreement if necessary (see also R66). In particular, it must be ensured that external interfaces of the platform take these requirements into account, e.g., when transferring data or data sharing between companies.

R54. The platform **must** ensure possibilities to object to the processing of personal data. [Q, S]
Explanation: If the platform is only used within a company, this can be regulated by a company agreement if necessary (see also R66). In particular, it must be ensured that external interfaces of the platform take these requirements into account, e.g., when transferring data or data sharing between companies.

R55. The platform **must** provide ways to object to the direct marketing of personal data. [Q, S]
Explanation: If the platform is only used within a company, this can be regulated by a company agreement if necessary (see also R66). In particular, it must be ensured that external interfaces of the platform take these requirements into account, e.g., when transferring data or data sharing between companies.

R56. The platform **must** provide options for objecting to the transfer of personal data to third parties. [Q, S]
Explanation: If the platform is only used within a company, this can be regulated by a company agreement if necessary (see also R66). In particular, it must be ensured that external interfaces of the platform take these requirements into account, e.g., when transferring data or data sharing between companies.

R57. The platform **must** offer possibilities to object to the decision support based on the automated processing of personal data. [Q, S]

R58. The platform **must** support the detection of personal data breaches and their communication with data subjects. [Q, S]

R59. The platform **must** ensure authorization with regard to access to personal data. [Q, P]
Explanation: Authorization can be realized via the mechanisms mentioned in R40.

R60. The platform **must** provide privacy principles when consent is required for the processing of personal data. [Q]

R61. The platform **shall** enable users to control their personal data requirements. [Q]

R62. The platform **shall** facilitate the assessment of data protection impact assessment to identify threats and risks in the processing of personal data. [Q]

R63. The platform **shall** provide a mechanism to capture user privacy and security requirements. [Q]

R64. The privacy mechanisms **shall** anonymize the information in specified data fields. [D]
   a. The specification of the data fields **shall** be done via the configuration model. [Q]
   Explanation: In an application, e.g., anonymous IDs could be used for workers. Furthermore, machine data could be anonymized with the time stamp and the anonymous worker ID.

R65. The privacy mechanisms **shall** detect and anonymize personal data contained in free text. [D]
   a. The specification of the data fields **shall** be done via the configuration model. [Q]
   Explanation: Anonymizing all text fields can be time-critical. Enabling many text fields for anonymization can overload the configuration model. An appropriate compromise must be found and implemented.

R66. Pseudonymization **shall** keep pseudonyms in the system only for the time actually needed. [D]
   a. After the time has expired, a new pseudonym **shall** be assigned to the same person. [D]
   b. The purpose and scope **shall** be aligned with the operating agreement when used internally. [D]
   c. Pseudonymization **shall** be applied in case of external use only after the consent of the person concerned. [D]

R67. The platform **shall** capture and classify generated cookies or similar identifiers stored on end devices. [S]

R68. The platform **shall** provide the possibility of automatic deletion of such (R61) identifiers, as well as deletion at the request of the user. [S]

3.8 Central Data Storage Services

One important platform functionality for applications and demonstrators is the ability to store arbitrary data. Current Industry 4.0 platforms [33] offer various forms of "data lakes", some of which are based on cloud technology. Some edge device vendors also offer similar functionality. However, the usability, flexibility and data security across manufacturer boundaries is relatively limited so far. This also applies to the declarative and flexible selection of storage services depending on quality criteria such as data volume or data frequency. In addition to central data storage services, deployment units can contain local data storage services, e.g. as buffers (R33).

R69. The platform **must** offer the integration of alternative storage services with different quality characteristics [D]
Explanation: Conceivable quality characteristics are e.g., speed, capacity, safety.

R70. A storage service shall describe its (quality) properties and functions as AAS. [Q]
   a. The management shell **can** contain information for direct access to the storage mechanism. [X]

Explanation: Direct communication endpoints may be required to achieve the best performance. It would be desirable to use only a few protocols for database access. Standardization work in this direction is currently taking place in the "Platform Industrie 4.0" [25].

R71. The storage services **shall** be available to all resources. [D]

a. A concrete storage service **shall** be selected transparently by the platform based on predefined properties. [D]

R72. The platform **must** implement at least one concrete on-premise storage service. [D]

a. The platform **can** support connections to databases. [D]
Explanation: A connection to Oracle DB is desirable.

R73. The storage services **shall** support the following types of data [T].

a. The storage services **shall** be able to store structured data. [Q]
b. The storage services **shall** be able to store heterogeneous time series and process them with potent compression functions. [T]
c. The storage services **shall** be able to store unstructured data such as logs. [T, D]
d. The storage services **shall** be able to store labeled data. [U]
e. The data schema for storage services of structured data **shall** be defined in the configuration model. [Q, D]
f. The data schema **must** be saved with the data. [T]

Explanation: Changes in the configuration model lead to changes in the data schema of the storage services. Adding new information should be possible. Non-compatible changes can be rejected. The data schema is relevant for the meta-information in data sharing (R79) and must therefore be accessible.

R74. Storage services shall support write access in soft real-time. [D]

a. Real-time storage services **shall** be able to store three floating point values at 0.2ms sampling rate [D]
b. Real-time storage services **shall** be able to store 10 floating point values at 8ms sampling rate. [D]
c. Real-time storage services **shall** be able to store the integration of data from sub-requirements a and b. [D]
d. Real-time storage services **shall** be able to store data at 2GByte per hour. [D]

Explanation: R74.c was included so that (local) storage services (R33) can realize this requirement into account even without existing data integration. R74.a - R74.c may be met by local storage services if necessary. R74.d requires even higher throughput rates, which may require distributed storage and can no longer be met by local storage services.

R75. Storage services **shall** provide read access in soft real time. [T]

R76. The platform **must** provide functionality to query available data. [T]

a. The Platform **can** provide automated mechanisms for deleting data. [T]
b. The platform **shall** provide access to subsets of accessible data. [T]
c. The platform **shall** enable the further processing of accessible data. [T]

Explanation: Entire datasets but also parts can be made available for further processing, whereby the mechanism/specification for selecting the datasets is not to be defined here. Provided datasets can be made available as a stream or container, for example. Storage services can be made available for data exploration [36] under appropriate access rights, if necessary after prior copy of partial datasets.

R77. The platform **can** offer optional cloud-based storage services. [E, D]

a. The configuration model **must** offer the use of cloud-based storage services (R72) as an option. [E, A]

Explanation: For a platform instantiation without cloud-based storage services, all cloud dependencies and functionality must be removed from the platform code to increase trust in the platform.

b. The platform **can** offer AWS as cloud storage service. [D]
Explanation: AWS Frankfurt was mentioned as a specific location by a demonstrator.

c. The platform **can** offer Google as a cloud storage service. [D]

R78. The platform **must** provide a storage for services. [U]

a. The platform **must** provide a store for individual services by version. [U]

b. The platform **must** provide storage for deployment units. [U]

Explanation: This requirement refers to the "service store" in the platform usage view [36]. The binary code of the services must be made available in an appropriate and versioned manner (R78.a). Services and application mechanisms assembled into deployment units must be made available to resources for download/use (R78.b).

3.9 Data Sharing

Flexible and configurable data sharing between related companies is an innovative and novel functionality of the IIP-Ecosphere platform. Data Sharing allows the sharing of own data and the use of external data via external interfaces of the platform.

R79. Data **must** have meta-information. [T]

a. The meta-information **must** include a unique identifier. [T]

b. The meta-information **must** describe the data format. [T]

c. The meta-information **shall** describe the size of the data. [T]

d. The meta-information **must** include a timestamp of creation/modification. [T]

e. Meta-information **must** include access/usage rights. [T]

f. Meta-information **shall** describe the origin/path of the data. [T]

Explanation: This requirement refines R19.g. The meta-information should describe the exact origin, pre-processing by components/services, the structure (names, types, "semantics" of the fields) as well as the access and usage rights (e.g. by smart contracts). The meta-information can also contain further security information such as hash codes.

R80. Data **shall** be described in the configuration model. [Q]
Explanation: Since all data formats required for applications are described in the configuration model, "protection classes" can be defined for them. The protection classes determine the (IDS) connectors to be provided, the adaptation of the data formats, and the necessary anonymization or pseudonymization (see also Sections 3.4, 3.6 and 3.7). Further mechanisms are conceivable. Central meta-information can be derived directly from the configuration model description of the data and its processing (R79).

R81. Secure containers **shall** be used for data exchange beyond the boundaries of the platform instance. [A, T]
Explanation: When processing large amounts of data, e.g., an AI model or data to be learned on third-party resources (see R27, R28 and R71), it may be necessary to protect both the data as far as possible from access by the resource owner. Nevertheless, it should be possible to process the data on the external resources appropriately.

R82. The platform must provide functionality to query available data. [T]

a. The platform **must** provide the data accessible to an authenticated user based on access/usage rights. [T]

b. The platform **must** enable the deletion of accessible data based on access/usage rights. [T]

Explanation: This requirement extends R76 in the sense of manual access/data sharing. A user should be able to identify and (if necessary) delete the records available or ready for data sharing.

3.10 Data Integration

Data integration in the IIP-Ecosphere platform is intended to provide AI algorithms and applications with additional perspectives on production beyond pure production data. For this purpose, models such as design data are to be enriched with production data in the (stream-based) data integration.

R83. The data integration **shall** be provided as a platform container. [Q]
Explanation: Even if the data integration is distributed internally and will probably be installed permanently, this shall happen in a container-based manner. The installation may consist of more than just "one container" as long as one container provides the AAS.

R84. The data integration **shall** access further data sources such as internal data stores. [Q, A]
Explanation: Any necessary security mechanisms (see Section 3.6) must be defined, provided and configured for this purpose.

R85. The data integration **must** describe its (quality) properties and functions as a management shell. [Q, D]
Explanation: This applies to access "from top" as well as "from" the AI algorithms, and likewise access from edge devices.

R86. The functionality of the data integration **shall** be defined by the configuration model. [Q]
Explanation: This can be used to ensure that services or applications can operate on the necessary data.

R87. Data integration **must** be able to integrate both streaming-based OT and semantic IT data. [Q]

R88. Data integration **must** enable applications and AI services independent from programming language and deployment access to integrated data. [Q]

R89. The platform **must** allow the data integration write access to data. [Q]
Explanation: It should be possible to store data modified and generated by data integration in suitable data stores (see R33, Section 3.8). The data stores shall be defined in the configuration model (R86). Data integration shall add or adapt data meta-information where necessary (R19.g).

R90. Data integration **must** allow protecting access to data. [Q]
Explanation: This requirement emphasizes data protection (Section 3.7) in this context.

R91. Data integration **must** meet the soft real-time requirement. [Q]
Explanation: This requirement extends the soft real-time requirement for production-related platform operations mentioned in R10 to data integration. An example for the integration of different data into a data table could require to integrate, e.g., values in the machine clock speed every 8 ms and external sensors every 0.2 ms or an OT sampling frequency of 2 ms (R35). One application of data integration can be the aggregation of measured machine data, e.g., to reduce input data from 7 GByte per hour to relevant, storable data of 2 GByte per hour.

R92. Data integration **must** allow historical data to be retrievable. [Q]

3.11 Configurability

Systematic configurability of the platform facilitates installation, maintenance and definition of applications. Optional components such as cloud integrations or IDS functionality can be completely removed from a platform instance through appropriate instantiation measures, which should increase confidence in the platform. This could result in platform instances that lack required functionality for applications. Therefore, it is important to check configurations for validity. Configuration information can further be used at different (later) points in time, for example, configuration settings determined before runtime can usefully limit the scope of adaptation decisions.

Configuration properties usually affect several or even all functional areas. Therefore, requirements for the configuration model are sometimes mentioned in other sections without repeating the requirements here.

The following requirements do not enforce complete configurability, since the Think Tank "Platforms" focuses on researching platform concepts for Industry 4.0 rather than a seamless and complete implementation of a product.

R93. The platform **must** be systematically configurable in the form of a configuration model. [A]
Explanation: This requirement takes up R8.a.

R94. The platform **must** support the automatic validation of the configuration model for inconsistencies and errors. [A, Q]
   a. Testing of a configuration model with 50 resources and 5 applications **shall** be completed in less than 1 second. [Q]

   Explanation: This requirement concretizes R8.b.

R95. The platform **must** support automatic platform instantiation for a configuration. [A]
   a. The instantiation of a configuration model with 50 resources and 5 applications **shall** be completed in less than 15 minutes. [Q]

R96. The configuration model **must** represent optional and alternative platform components/services. [A]
   a. The configuration model **must** describe properties of the platform components/services.

   Explanation: Examples are license restrictions (R6) based on a highly simplified license model, security components (R40, R41, R44), cloud integration (R74), performance aspects, etc. Some of these properties should be suitably protected (e.g. read-only after initial definition).

R97. The configuration model **must** include the applications running on the platform. [Q]
   a. An application configuration **must** contain the configured services for the application. [Q, U]
   b. An application configuration **must** contain the configured connectors for the application. [Q, U]
   c. An application configuration **must** contain the data paths of the application. [Q, U]
   d. An application configuration **shall** contain alternative services. [Q, U]
   e. The configuration model **can** allow for application templates. [U]

R98. The configuration model **shall** support customizations at different times in the software lifecycle. [A, Q, U]
Explanation: The configuration model shall be used to customize the IIP-Ecosphere platform before runtime, to control optimizations at startup time, and to control customizations at runtime.

R99. Information from the configuration model **can** be made available to other components via internal connectors. [Q]
Explanation: Properties from the configuration model can in turn be reflected in its AAS.

R100. The configuration model **can** be a decentralized model. [Q]
Explanation: A central model may stand in the way of local optimization. However, this requires preliminary work from the BMBF-funded DevOpt project [4].

R101. Information provided in the AAS of components/services **shall** be mapped automatically into the configuration model. [Q]
   a. The transfer of information for a configuration model with 50 resources and 5 applications **shall** be completed in less than 1 second. [Q]

Explanation: To avoid inconsistencies between the configuration model and the components or their (runtime) properties, relevant information from the management shells should be transferred automatically.

3.12 Optimizing / Adaptive Deployment

The platform's automatic deployment optimization makes it easier for administrators to install and deploy the platform. However, relevant properties can change during runtime. Therefore, adaptive deployment is targeted, i.e., re-optimization and adaptation of the deployment at runtime. The optimization can support or realize R9 (availability).

R102. The platform **must** allocate deployment units to existing resources. [A, U]

a. The platform **must** validate the resource allocation. [U]
b. The platform **shall** provide resources for data exploration. [U]

Explanation: This is initially a (not necessarily optimal) allocation of containers to resources. The allocation must be checked for validity and the user must be informed if necessary. A special form of allocation is the provision of requested resources for data exploration (R104.b) in the context of a data science toolchain [36]. Since this may require central resources, e.g., GPUs, to be provided to a manually assembled container that competes with resources used by the applications, e.g., for retraining AI models, this form of allocation is the responsibility of the platform, i.e., adaptive deployment. The resource allocation may address resources in underlying platforms via connectors (R13).

R103. The platform **shall** optimize the allocation of deployment units to resources. [A]

a. The optimization **shall** consider resource usage (CPU, memory, etc.). [Q]
b. The optimization **shall** consider network properties. [Q]
c. The optimization **shall** consider component properties. [U]

Explanation: Component properties (c, R96.a) may influence/restrict the distribution of components, exclude resources if necessary, e.g., resources outside a factory due to legal reasons or licensing reasons.

R104. Resource allocation optimization **must** use the configuration model as the basis for optimization. [Q, A, U]

a. The resource optimization **must** use the configuration model as the basis for optimization. [Q, A]
b. The resource optimization **must** ensure the validity of the configuration model. [Q, A, U]

Explanation: The configuration model contains the information about the applications, the required components, the resources and their runtime information. R104.b is based on R94R8.b.

R105. Enforcement of resource allocation must be done via platform functions and management shells. [Q]

Explanation: For this purpose, R25 and R32 must be taken into account.

R106. The optimization **must** happen "statically" when starting the platform/application. [A]

R107. The optimization shall happen dynamically at runtime. [Q, A]

a. The optimization **must** regularly check the validity of the configuration model. [Q]
b. The resource optimization of a platform instance with 50 resources and 5 applications **shall** be completed within 15 minutes. [Q]

Explanation: This enables adaptation of the running applications or platform, e.g. to adapt the allocation when resources change, the configuration is changed, or over/under-used resources are detected, etc. The enforcement is then done at runtime via R105.

R108. The optimization **must** consider the executability of the applications. [Q]

Explanation: The optimization must not interfere with the execution of applications consisting of distributed components, i.e., the (dynamic) optimization must not have a negative impact on the execution of production processes or the required adaptation mechanisms must be designed accordingly.

R109. The optimization **shall** provide transparency and traceability about the decision making. [D]

3.13 AI (Service) Toolkit

The open and extensible AI construction kit is another innovative building block of the IIP-Ecosphere platform, especially since only a few Industry 4.0 platforms have so far recognized or realized the need for user-specific AI components [33]. The elements in the AI building block will be components/services that publish their interfaces and properties via AAS and (if possible and reasonable) are distributable in terms of deployment (Section 3.5), e.g., training services on GPU resources and associated prediction services on edge devices. The platform will provide interfaces for distributable services (Section 3.5), data paths (R20), data stores (Section 3.8), data sharing (Section 3.9), and data protection (Section 3.7) as foundations for this. Services are linked via data paths (R20) as part of the application configuration (Section 3.16).

The specification of the actual interfaces for AI services as well as the actual design of the AI building block based on current AI understanding and AI knowledge requires close and constructive collaboration between different partners. While requirements for the AI building block are mentioned below, e.g., also a wide variety of AI methods (R114) or AI frameworks (R113) that stakeholders mentioned during the requirements elicitation process and would like to see integrated into the platform are listed. For an implementation of the AI toolkit, the AI procedures must be suitably abstracted as services (based on platform interfaces and services) and supplemented with AI-specific interfaces. The AI methods to be integrated must be prioritized for implementation and integration, since it may not be possible to integrate all desired methods within the project time (shall-requirements). Here, it is more important for IIP-Ecosphere to demonstrate the approach and to work towards standardization/standardization of interfaces than to integrate as many AI methods as possible. This (and the related decisions) cannot be done exclusively by the Think Tank "Platforms" and, thus, collaboration and responsibility of several project parts, e.g., especially Think Tank "AI and Production", AI-Accelerator or even individual demonstrators must be involved. The Think Tank "Platforms" will contribute to this design (as described above) and incorporate the results appropriately into the platform so that the AI interfaces become an integral part of the platform and that the necessary implementations based on existing frameworks are available to the platform (and thus to the applications running on it) for configuration, deployment and execution.

In this context, it is important to refer to R9: AI services extend the platform functionality, i.e., if AI services are delivered with the platform, the responsibility for the quality of the services (in particular scheduling, memory usage, data type conformance and response times) lies initially with the creator but then implicitly with the IIP-Ecosphere platform. For external services, compliance with the required quality does not lie with the platform, i.e., as described for R9, the platform may have limited means to monitor or enforce quality compliance. This applies in particular to the composition of services into applications (Section 3.16), where the responsibility for overall quality lies accordingly with the user.

R110. The AI toolkit **must** define interfaces for relevant AI components in industrial production. [A, Q]

a. The interfaces of the AI toolkit/services must be realized as management shells. [Q]
b. The AAS of the AI components/services must describe both functional and quality properties [Q]

Explanation: These can be uniform/generic or heterogeneous interfaces. It is desirable that only necessary interfaces are defined (see also format economy in R19).

R111. The AI toolbox **must** be extensible so that users can add AI components to the platform or platform instance. [A, Q]
Explanation: For this, the AI components must comply with R110.

R112. The components/services in the AI toolkit **shall** be distributable to available resources in deployment units. [A]
a. Parameters of the components **shall** be described in the configuration model. [U]
b. Properties of the distribution **shall** be described in the configuration model. [A, Q]
c. Distribution **shall** be subject to restrictions for individual components. [D]
Explanation: The (dynamic) distribution of components based on their supply and resource consumption (R24) as well as their use within applications (R131) is a core functionality of the platform. For this purpose, it may be necessary to equip the components with different interfaces depending on their function, e.g., one for training and one for prediction. However, components may need to be deliberately executed centrally, which can be represented either by properties (R96.a), a restriction on the target resources, a corresponding resource requirement or - if necessary - by corresponding settings (R103.c, R96.a) in the configuration model or the underlying management shells (R101).

R113. The AI toolkit **shall** support AI components in common programming languages and programming environments. [D]
a. The AI toolkit **shall** support Python, Pandas/Numpy [24], Knime [19], Scikit-learn, Tensorflow and Rapidminer [30]. [T, D]
Explanation: Various programming languages (e.g., Python) and frameworks (e.g., Tensorflow) are used in the AI environment. These must also be supported in a systematic way in deployment, which may require the specification of suitable dependencies (in the configuration model). See also R6 and R96. Since Rapidminer is a project partner, priority shall be given to the integration of the Rapidminer approach compared to the Knime approach (both were mentioned in the same demonstrator context). R113.a names some examples but is not meant to be complete.
b. The AI toolkit **shall** provide a middleware for the agent management based on the Industry 4.0 Language [27, 38]. [T]
Explanation: This middleware should be able to handle a standardized creation and deletion of software agents, as well as use a standardized communication (extended Contract Net Protocol [27]) (similar to JADE- Java Agent Development Framework [16]). This could then also subsume FIPA [10].
Explanation: Depending on the AI method, it may be necessary to perform computations on appropriate IT resources such as GPUs, then export the solutions and deploy them with the procedures to the respective executing resources such as edge devices (see R32).

R114. The AI toolbox **shall** provide relevant AI methods that are common in production or suitable for production. [D, T]
a. AI methods in the AI toolkit **shall** be generic, parameterizable and adaptable. [D]
b. The AI toolkit **shall** support Transfer Learning. [D]
c. The AI toolkit **shall** support Reinforcement Learning [T, D].
Explanation: In particular Deep Q Learner as in Tensorflow shall be supported.
d. The AI toolkit **shall** support simple statistical procedures for decision making. [T]
e. The AI toolkit **shall** support neural networks. [T]
f. The AI toolkit **shall** support time series classification. [D]
Explanation: This requirement complements R114.e. In particular, Residual Neural Network and Feed Forward Neural Network as in Tensor Flow should be supported.
g. The AI toolkit **shall** support anomaly detection. [D]

Explanation: In particular, One Class SVM and AutoEncoder as in Scikit-learn should be supported.

h. The AI toolkit **shall** provide methods for state detection. [D]

Explanation: The procedures are enumerated here as a base set to support the design of the AI building block. R114.h could be implemented through R114.g. Provision and integration are the responsibility of the demonstrators using it, the Think Tank “AI & Production”, and, for use in a broader environment, the AI-Accelerator (especially for R114.a).

R115. The AI toolkit **shall** support AI models in standardized formats. [V, E]

Explanation: Models in standardized formats such as ONNX [23] support the application of different (implementations of) AI methods as well as their deployment. The AI toolbox is intended to support and promote the use of such formats.

R116. The AI toolkit **shall** provide procedures for the pre- and post-processing of data [T].

a. Pre- and post-processing procedures **shall** be generic, parameterizable and customizable. [D]

b. The AI toolbox **can** provide feature design / digital signal processing services (bandpass filters, FTF transforms, running RMS). [T]

Explanation: The methods are enumerated here as a basic set to support the design of the AI toolkit. Provision and integration are the responsibility of the using demonstrators, the Think Tank “AI & Production” and, for use in a wider environment, the “AI-Accelerator”.

R117. The AI toolbox and the AI methods included **shall** provide transparency and traceability about their decision-making. [D]

R118. The platform **must** provide central services to the AI construction kit.

a. The platform **shall** provide warning and alerting services to the AI toolkit. [D, U]

b. The platform **must** provide storage services to the AI toolkit. [D]

c. The platform **must** provide security services to the AI toolkit. [Q]

d. The platform **must** provide privacy services to the AI toolkit. [Q]

e. The platform **must** provide data integration services to the AI toolkit. [Q]

Explanation: Some sub-requirements only serve as clarification that central platform services in particular must be provided to the AI toolkit. Specifically named here are storage services (R118.b vs. Section 3.6), security services (R118.c vs. Section 3.7), data protection services (R118.d vs. Section 3.8) and data integration services (R116.e vs. Section 3.10). Services are linked via data paths (R20) as part of the application configuration (Section 3.16). Not mentioned, for example, is dynamic or adaptive deployment or adaptive exchange of AI procedures, since these platform services are to be implemented as transparently as possible for the AI toolkit.

R119. The training of AI methods **shall** happen automatically and in parallel/background. The trained models shall then be offered to the components of the platform. [D]

a. The release of the trained model **shall** be done manually by a user with appropriate rights. [D]

b. The release of the trained model **shall** be determined via settings in the configuration model. [D]

c. The release of the trained model **can** be automatic (if specified in the configuration model). [Q]

d. The changes triggered by a release of a trained model **shall** be traceable and traceable. [D]

e. Changes initiated by a release **shall** be reversible. [D]

Explanation: a corresponds to approval level 1 in [28], i.e., event-driven learning with human release. b softens approval level 1 in the direction of partially automated releases for configured components or applications. c corresponds to approval level 2, i.e., automatic release and learning [28], possibly based on configurable release or quality criteria.

3.14 Adaptive Service Selection

Nowadays, the choice of AI methods and AI components is relatively static. Nevertheless, the production environment can change rapidly, e.g., because new sensors or new edge devices are introduced into a production process, changing the data volumes, the data types, or even their quality. In such situations, it may be easier to adapt the used AI method or to change parameters of the AI method/AI component. This can be solved through dynamic deployment (see Section 3.5) and, potentially, more efficiently with an adaptation of the AI component in its deployment environment. The adaptation can support or realize R9 (availability).

R120. The configuration model **must** describe alternative AI components for an AI method. [Q, A]

R121. The platform **shall** monitor the use of AI methods and detect situations that require replacement of AI methods. [Q]

R122. Based on the configuration model, the platform should be able to decide on the optimal use of AI methods in the context of R120. [Q]

a. The adaptation **must** be based on the configuration model. [Q]
b. The adaptation **must** regularly check the validity of the configuration model. [Q]
c. The adaptation **must** store its decisions in the configuration model. [Q]
d. The adaptation **must** ensure the validity of the configuration model. [Q, U]
e. The adaptation of a platform instance with 50 resources and 5 applications **shall** be completed within 1 minute. [Q]
f. Adaptation **shall** consider component properties. [U]

Explanation: The decision can be made statically, but should (usually) be performed dynamically at runtime. R122.d is based on R8.b. Component properties (R122.f, R96.a) can influence the distribution of components, possibly excluding resources, e.g., resources outside a factory because of legal reasons or license reasons.

R123. The enactment of the adaptation **must** be done via platform functions and AAS. [Q]

a. The platform can offer mechanisms for the update of services. [U]

Explanation: The adaptation may address services in underlying platforms via connectors (R13).

R124. The adaptation **must** take into account the executability of the applications [Q]

Explanation: The adaptation must not interfere with the execution of applications with distributed services/components, i.e., the adaptation must not have a negative impact on the execution of production processes or the required adaptation mechanisms must be designed accordingly.

R125. The adaptation **shall** provide transparency and traceability about the decision-making process. [D]

R126. The platform **shall** independently search for possibilities of optimization (of AI deployment) and offer them centrally as a proposal. [D, Q]

Explanation: Akin to R119, the adaptation may be completely autonomous or subject to approvals depending on the configuration settings. R119 applies here similarly.

3.15 Virtualization

Digital twins are used in the field of industrial production for various purposes. Unfortunately, no standardization is currently foreseeable for digital twins [37]

R127. The platform **shall** enable the integration of digital twins. [Q]

R128. Digital twins **must** be described/interfaced as AAS. [Q]

R129. The platform **can** support and manage deployed containers with digital twin installed. [Q, D]
Explanation: Due to the lack of standardization [37], manual installation as a container is a potential option. If the containers meet the specifications of the platform, e.g., with regard to distribution (see R34.b), then the platform can take over the management of the containers. Other installation forms include virtual machines or direct installation on a resource. These installation forms can probably not be managed by the platform.

R130. The platform **can** integrate components to perform digital twins. [Q]
Explanation: For example, the platform can provide appropriate frameworks such as Eclipse Ditto [7] to facilitate the integration of digital twins.

3.16 Application Support

Industrial platforms often offer the possibility to run preconfigured applications or to load provided applications (as "apps") onto a platform [33]. To increase the applicability of and awareness for the IIP-Ecosphere platform, we name here the possibility of an "app store" knowing that resources for a realization might only be available after reallocation (possibly in the AI-Accelerator) due to the approved subproject grants. Similar considerations apply for resources for the realization of a user interface or for visualizations. As applications are composed from platform and AI services, the ultimate responsibility for the quality (e.g., timeliness, memory usage, termination) of the application lies with the application designer or user as discussed in Section 3.13.

R131. The platform **must** support the creation and deployment of applications for smart manufacturing. [Q]
   a. The configuration model **must** support the specification of applications, their required services, connectors, involved data paths and the needed resources. [Q]
   b. The configuration model **must** allow for the versioning of applications and services. [U]
   c. The configuration model **can** enable the parameterization of applications. [U]
   d. The configuration model **shall** support application templates [36] for simplified configuration of requirements. [U]
   e. The configuration model **must** describe dependent applications or services. [U]
   f. The configuration of applications and data paths **can** be done in a graphical way. [Q]
   g. The platform **must** support the deployment of applications. [U]
   h. The platform **shall** support interconnected platform instances. [U]
   i. The platform **shall** support interconnected applications. [U]

   Explanation: Applications consist in particular of required components or services offered by the platform (e.g., data protection services, AI services), but also application-specific services (R132), alternative components or services if necessary, the data paths (R20) between the components and data endpoints (e.g., sensors, actuators, storage services). Applications can declare dependencies to services or further applications (R131.e), which, e.g., must be started before the dependent application. The deployment of applications (R131.g) relies on the deployment of components/services, but must be listed separately, as it may also include/require configuration of data connectors [36]. Connections between applications (R131.i) and in particular between connecting platforms (R131.h) may have to be created manually.

R132. The platform **can** support application-specific components. [Q]
   a. The configuration model **must** support application-specific services. [Q]

R133. The platform **must** provide run-time support for applications. [U]
   a. The platform **must** know the status of the services. [U]
   b. The platform **must** know the status of the running applications. [U]

c. The platform **can** offer functions to adjust the status of the running applications. [U]
d. The platform **shall** provide functions to resolve error conditions in an application. [U]
Explanation: The status is a summary of the operational state of a service/application, i.e., "deployed", "starting", "running", "failed", "reconfiguring". The status can be determined based on the status of the services of the application. Within limits, it may be necessary to set the state explicitly. For some functions such as deployment, optimization (Section 3.12), and adaptation (Section 3.14), the platform must modify the state appropriately. Within limits, the platform shall monitor execution at this level and correct errors (R133.d), e.g., by restarting services.

R134. The platform **must** support the removal of applications/services. [U]
a. The platform **must** support the removal of deployed applications from resources. [U]
b. The platform **can** support the removal of applications from the configuration model. [U]
c. The platform **can** allow the removal/disposal of services. [U]
d. The platform **shall** enable the disconnection of connections between applications. [U]
e. The platform **shall** enable the disconnection of connections between platforms. [U]
Explanation: These are the counter function for deploying/defining applications (R131). Depending on the implementations, services can be replaced by newer versions, decommissioned (but kept in the platform) or even deleted if necessary (R134.c). Connections between applications and platforms (R134.d, R134.e) may have to be terminated manually [36].

R135. The platform **shall** support the update of applications. [U]
Explanation: Updating an application in a production environment is a critical activity. This can be done by shutting down the application (offline) or (partially) during runtime (online). If necessary, update steps can be denied as part of the validation of the configuration model (R94), e.g. if compatibility between interfaces is violated. Thus, possibilities for the evolution of applications are created. If necessary, this also includes an update of the connection between applications and platforms (R131.h, R131.i).

R136. The platform **can** contain functions to update the platform. [U]
a. The platform **shall** provide an update to the resource abstraction layer. [U]
Explanation: The resource abstraction layer ("ECS runtime" in [36]) is seen here as part of the platform rather than the applications/services. This abstraction view may need to be updated as the IIP-Ecosphere platform evolves, which can be considered as part of this requirement. Ideally, the abstraction layer can update itself, but interaction with the (store of the) device manufacturer may also be required.

R137. The platform **can** provide functions to secure the deployment of applications [U].
a. The platform **can** support the simulation of the integration of resources. [U]
b. The platform **can** support a pre-deployment simulation. [U]
c. The platform **can** support a test run of an application. [U]
Explanation: Deploying an application to a production environment is a critical activity. Therefore, mechanisms such as simulations (R137.a) are proposed in [36], also to estimate, validate and, if necessary, adjust the resource requirements.

R138. The platform **can** provide an app store. [Q]
Explanation: Due to the actual subproject descriptions, no app store is planned, although a corresponding functionality would be very helpful. Therefore, this requirement is prioritized as "can". Removing an application/service from the configuration model (R134.b) may result in removal from the store. Furthermore, if services/applications are installed in a platform instance, they should be included in its configuration model.

R139. The platform **can** provide a user interface. [Q. P]

a. The platform **can** provide a user interface for agent management and development. [T]

b. The platform **can** provide a user interface for the "tools" from [36]. [U]

Explanation: Industrial platforms typically provide a user interface, e.g., using web-based technologies. Since the goal of the Think Tank "Platforms" is the research and evaluation of platform concepts rather than the development of a fully-fledged Industry 4.0 platform, primarily no resources are reserved for the development of a user interface. Nevertheless, a (simple) user interface for an easy-to-use demonstration of the platform would be desirable.

R140. The platform shall support data analysts in the creation of AI services ("Data Science tool chain" in [36]). [U]

a. The platform shall provide appropriate northbound interfaces. [U]

b. The platform shall provide appropriate resources. [U]

Explanation: R140.a extends R19.d and R140.b provides access to the deployment (Section 3.5) and the resource assignment/optimization (Section 3.12).

R141. The platform **can** enable visualization of applications or data. [Q, D]

Explanation: This can be done in particular through external northbound connectors or via corresponding AAS. In general, visualization capabilities must be considered as outside the platform due to the subproject grants, although visualization capabilities are very useful in various environments such as for demonstration. Conceivably, external frameworks (some mentioned by the demonstrators) such as Grafana [11], BaSys4Dash [2], Rapidminer [30], or Pandas [24] could be utilized. The selection for support should be made in consultation with the demonstrators and the respective plans for public demonstrations.

4 Synchronization with the Usage View

In addition to this collection of functional requirements, partners of the IIP-Ecosphere project also collected a usage view (inspired by [12]) as part of the requirements elicitation. The result is presented in [36] and includes the description of the so-called System Under Consideration, 18 entities, 19 roles and 67 activities. The elicitation of the usage view was done in a series of workshops with IIP-Ecosphere partners. During these, the vision from Section 2 was presented as a framework (scope).

We integrated the findings from [36 into the functional requirements in this document. For this purpose, the requirements in this document were no longer changed until the creation of the user view was completed. Then, we performed an alignment at the end of the work on [36]. If the user view describes a requirement that was already known at the time of integration, we entered the user view as a source of the respective requirements using the marker "[U]". If the user view formulates a new requirement, we added the requirement to this document and recorded the addition in Table 1. Thus, this document can be understood as a more technical bridge between the user view and the functional view/architecture according to [12].

Table 1 describes each of the requirements added directly as results of the alignment with [36]. The structural changes made in the process lead in some cases to the addition of further requirements, or existing requirements have been detailed for reference purposes - these changes are not described here. We listed the affected requirements in this document, the triggering sections in [36] (we name in particular the first in content order, not necessarily all referencing sections) and the type of change, i.e., extension (addition of requirements or sub-requirements) or clarification of the description of existing requirements. No requirements were removed during the alignment.

We performed the along the order of the sections in [36], i.e., Table 1 was originally sorted according to the sections in [36]. Final revisions of [36] caused a disorder of the sorting. The entries "[Platform inter-connections]" refer to initial descriptions of activities for managing data connections between applications/platforms. These were already considered during the alignment, but were ultimately deferred to a later version of [36]. We kept these additional requirements in this document

Requirement	Section/Topic in [36]	Ext.	Clarif.
R4.d, R4.e	Section 3.8 „Services“ (monitoring)	X	
R4.c	Section 3.8 „Services” (state) Section 3.16 „Application” (state)	X	
R17.a, R17.b	Section 3.8 „Services“ (connectors)	X	X
R20.b, R20.c	Section 3.8 „Services“ (relations)		X
R78	Section 3.9 „Service store“	X	
R131.d	Section 3.9 „Service store“ (templates)	X	
R25	Section 3.13 “Device description store”, Section 3.14 “Device configuration tool”, Section 4.13 “Asset Data provider”, Section 5.1.1.2 “Adding an ECS device“		X
R123.a	Section 3.15 “Runtime application / service distribution tool” (updates)	X	
R102.a	Section 3.14 “Device configuration tool”	X	
R140	Section 3.18 “Data Science tool chain”	X	
R139.b	Section 3 all “Tools”	X	
R25.a, R25.c, R25.d	Section 4.2 “Edge device provider” (onboarding), Section 4.3 “Edge runtime provider”	X	
R25.e, R25.f	Section 4.2 Edge device provider (capabilities)		X
R27	Section 4.2 Edge device provider		X

Requirement	Section/Topic in [36]	Ext.	Clarif.
R28	Section 4.4 Cloud provider		X
R131.b	Section 5.1.6.1 “Deploying an (distributed) application” (versioning)	X	
R104.b, R122.d, R8.b	Section 5.1.6.1 “Deploying an (distributed) application” (validation, checking)	X	X
R131.e	Section 5.1.6.1 “Deploying an (distributed) application” (validation, checking)	X	
R136	Section 5.1.6.1 “Deploying an (distributed) application” (pre-deployment simulation), Section 5.1.6.2 “Updating an (distributed) application” (deployment testing), [Platform inter-connections]	X	
R17.c, R20.b, R112.a, R131.c, R131.g	Section 5.1.6.1 “Deploying an (distributed) application” (parameters)	X	
R135	Section 5.1.6.2 “Updating an (distributed) application”	X	
R133	Section 5.1.6.3 “Uninstalling an (distributed) application”, [Platform inter-connections]	X	
R136	Section 5.1.6.5 “Updating ECS runtime”	X	
R36	Section 5.1.6.7 “Replace an ESC device” Section 5.1.6.8 “Restore an ECS device”	X	
R37	Section 5.1.6.9 “Remote management of ECS devices”	X	
R136	Section 5.1.6.4 “Update of an ECS runtime”, Section 5.1.6.6 „Update of an ECS management system“	X	
R96.a, R103.c, R122.f	Section 5.1.4.1 „Provision of a service“	X	
R11.c	Section 5.1.4.1 „Provision of a service“ (logging)	X	
R134.c	Section 5.1.4.3 „Discontinuation of a service”	X	
R133.h	[Platform inter-connections]	X	
R4.e	Section 5.1.4.4 “Defining monitoring parameters for services”, Section 5.1.5.2 “Defining notification triggers (conditions) for monitoring parameters”	X	
R137.a	Section 5.1.7.2 “Simulating the Integration of ECS devices for an application”	X	
R30.c	Section 5.1.7.5 “Integrating external container registries”	X	
R133	Section 5.1.7.6 “Changing the state of an application”, Section 5.1.7.7 “Restoring an operational application”	X	
R102.b	Section 5.2.1.1 “Creating a data exploration process”	X	
R76	Section 5.2.1.1 “Creating a data exploration process”		X
R73.d	Section 5.2.2.10 “Backup of labeled data sets (for further learning/training”	X	

Table 1: Synchronization of the functional requirements with the usage view, Ext.=extension, Clarif.=clarification

In addition to mandatory platform services, the service concept of the IIP-Ecosphere platform can also be used to perform maintenance tasks, e.g., services for updating firmware on field or edge devices (Section 5.1.3.7 "Update firmware", [36]). Such optional services have not been captured in standalone requirements.

Not all activities from [36] are directly reflected in this document. This is due to the fact that various activities were already excluded from the scope of the IIP-Ecosphere platform in [36]. Furthermore, there are various activities that target the larger lifecycle of a system that we cannot map here.

Nevertheless, we have attempted to incorporate as many as possible activities and information, which translates into 10 new requirements (with 20 sub-requirements), a total of 53 new sub-requirements, and 9 modified/clarified requirements. The restructuring of existing requirements discussed above has added an additional 10 sub-requirements.

5 Prioritization and Requirements Conflicts

The requirements described in Section 3 are already roughly prioritized by the three categories "must", "should" and "can". In terms of all requirements (including sub-requirements), we prioritized 41% as "must", 43% as "should", and 16% as "can" requirements. The prioritization is based on the proposal in [36], although adjustments may have been necessary in the context of the requirements described here.

For the conversion into a platform architecture or realization as source code, we recommend the following sequence of topics:

1) Connectors and Interconnections (Section 3.4)
2) Distribution of Applications
    - Heterogeneous, dynamic Deployment (Section 3.5)
    - Configurability (Section 3.11)
    - Application Support (Section 3.16)
    - Virtualization (Section 3.15)
3) Services
    - AI (Service) Toolkit (Section 3.13)
    - Central Data Storage Services (Section 3.8)
    - Data Integration (Section 3.10)
4) Advanced Functionality
    - Data Sharing (Section 3.9)
    - Optimizing / Adaptive Deployment (Section 3.12)
    - Adaptive Service Selection (Section 3.14)

We take the probable dependencies as a basis and, starting from the basis of the platform, have the distribution of services and the development of applications for demonstrability or for initial experiments in mind as quickly as possible. Security (Section 3.6) and data protection (Section 3.7) are crosscutting issues (as are virtualization and configurability) that must be taken into account at the appropriate point (actually from the ground up). The "General Requirements" (Section 3.3) form the framework for all requirements and must be taken into account appropriately at every step.

Of course, during architecture and implementation work, trade-offs between requirements will have to be made or conflicts will be identified, which will then have to be resolved appropriately. Below, we list only a few examples of requirement conflicts:

- Additional data formats such as JSON/XML via REST (R19.c) could collide with the general requirement for uniform interfaces using AAS (R7). The desire for REST is quite understandable, since it is a common technology that is also used as an interface approach in various current platforms [33]. However, it must be taken into account that AAS implementations are often based on REST, i.e., in that case R19.c can be implicitly considered to be fulfilled, especially if the corresponding data can simply be represented in an AAS as strings in JSON/XML or the data can be converted into suitable internal formats (see transport format in R14).
- The requirement for manual releasing automated decisions (R119) may conflict with the basic goals of dynamic optimization and adaptation. For this reason, we have discussed the different release levels and provided possibilities with optional requirements to automate such releases via the configuration model. This should take into account the different needs.
- The AI service toolkit in Section 3.13 states many requirements that could trigger at least technical dependency conflicts, e.g., the various AI frameworks. This is also a reason why most

of the technical/methodological requirements in Section 3.13 are optional, and if necessary, unresolvable conflicts may lead to a permissible non-fulfillment of requirements.

- The requirements for user interface and visualizations in Section 3.16 collide with the work plans and the planned resources. Nevertheless, it is desirable and not excluded that under optimal conditions parts of a user interface or visualizations can be realized. Therefore, these requirements are classified as optional/nice-to-have.
- The ability to configure data formats can conflict with the integrity of the data. These conflicts can (partially) be described by global constraints and prevented when configuring a platform instance. In addition, application-specific constraints can be used. However, it is possible that not all potential conflicts can be described or have been specified. In addition, conflicts may exist that are fundamentally unavoidable.
- Non-functional requirements (e.g., R10, R19.a, R19.e, R22, R28, R35, R91) can trigger further conflicts, especially during realization/evaluation. Further, security and data protection requirements (Section 3.6 and 3.7) may affect quality aspects, e.g., the performance. Apparently, we do not expect serious conflicts due to the planned technologies to be used during implementation. Technically, these requirements may partially be solved through alternative components, or alternative components may introduce the actual conflicts.

6 Summary

This document describes the results of requirements elicitation from a functional perspective. For this purpose, both the user view described in [36] and an independent requirements elicitation through discussions, interviews and questionnaires were integrated. A total of 141 requirements with 181 sub-requirements in 14 subject areas were elicited, described, categorized and initially prioritized. Figure 5 graphically depicts the distribution of all requirements (including sub-requirements) across the 14 topic areas.

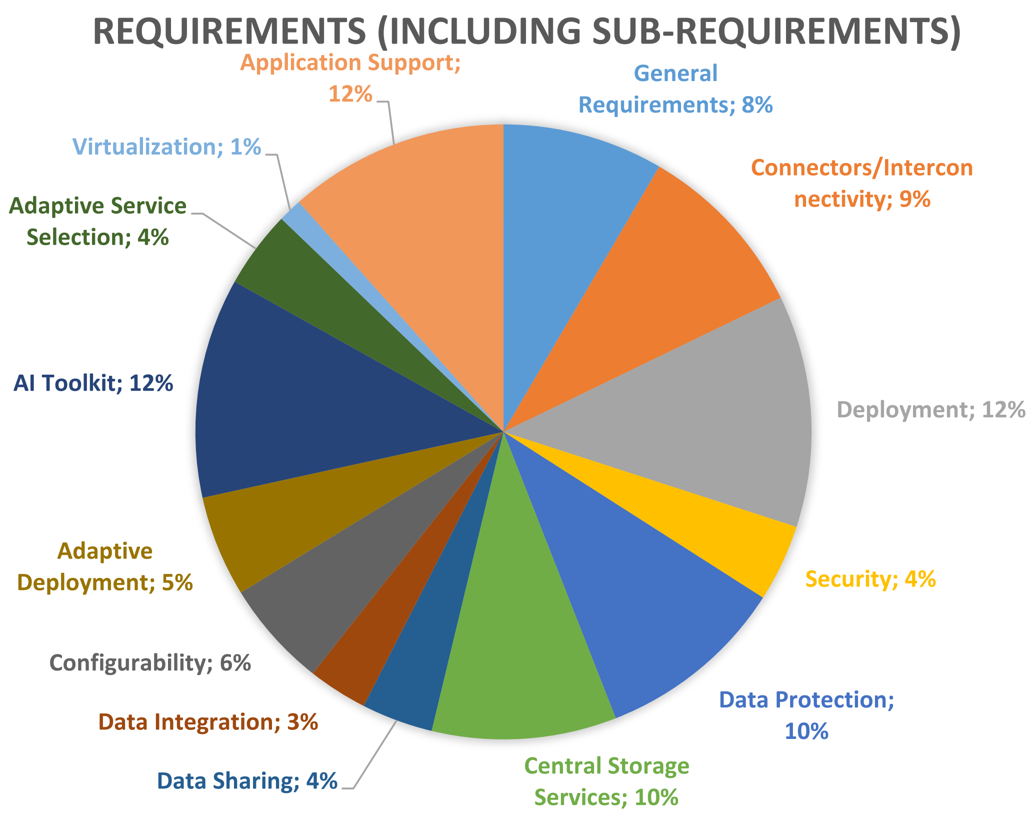


Figure 5: Distribution of requirements w.r.t. the topic areas.

Based on the IIP-Ecosphere platform overview [33], which laid the foundation for the development of a common platform vision, and based on the requirements from the user view [36], this document complements the requirements elicitation for the IIP-Ecosphere platform. As described earlier for the user view, we will update additional requirements that may emerge during the project lifecycle into incremental versions of this document. Requirements added later will probably not be able to (strongly) influence the development of the IIP-Ecosphere platform, but should serve future work as an even more complete requirements collection.

In IIP-Ecosphere, the next step is to specialize the requirements described here and in [36] in terms of an architecture and then into a realization of the platform.

We would like to thank all those who actively contributed to this requirements collection.

7 References

[1] BaSys 4.0, https://www.basys40.de/

[2] BaSyS Satellitenprojekte, BaSys4Dash, https://www.basys40.de/satellitenprojekte/

[3] K. Czarnecki and C. H. P. Kim. Cardinality-based feature modeling and constraints: a progress report. Intl. Workshop on Software Factories at OOPSLA'05, 2005.

[4] DevOpt – DevOps für Selbst-Optimierende Emergente Systeme, https://sse.uni-hildesheim.de/forschung/projekte/devopt/

[5] Docker, https://www.docker.com/

[6] Eclipse BaSyX, https://www.eclipse.org/basyx/

[7] Eclipse Ditto, https://projects.eclipse.org/projects/iot.ditto

[8] Eclipse IoT, https://iot.eclipse.org/

[9] H. Eichelberger, C. Qin, R. Sizonenko, K. Schmid, Using IVML to model the topology of big data processing pipelines, International Systems and Software Product Line Conference, 204-208, 2016

[10] FIPA http://www.fipa.org/index.html; http://www.fipa.org/specs/fipa00061/SC00061G.pdf

[11] Grafana, https://grafana.com/

[12] The Industrial Internet Reference Architecture Technical Report, https://www.iiconsortium.org/pdf/IIRA-v1.9.pdf

[13] International Data Spaces Association, https://www.internationaldataspaces.org/

[14] International Data Spaces Association, Reference Architecture Model, Version 3.0, April 2019, https://www.internationaldataspaces.org/wp-content/uploads/2019/03/IDS-Reference-Architecture-Model-3.0.pdf

[15] ITOperations, IT/OT convergence, https://searchitoperations.techtarget.com/definition/IT-OT-convergence

[16] JADE, http://jade.tilab.com/

[17] Joint Task Force, Security and Privacy Controls for Information Systems and Organizations, Draft NIST Special Publication 800-53, rev. 5, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5-draft.pdf

[18] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, A. S. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study, CMU/SEI-90-TR-21 ESD-90-TR-222, 1990

[19] Knime, https://www.knime.com/

[20] LNI 4.0 Testbed Edge Configuration – Usage View, https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/LNI4.0-Testbed-Edge-Configuration_UsageViewEN.pdf

[21] One2M, https://www.onem2m.org/

[22] OPC foundation, https://opcfoundation.org

[23] Open Neural Network Exchange, https://onnx.ai/

[24] Pandas, https://pandas.pydata.org/

[25] Plattform Industrie 4.0, https://www.plattform-i40.de

[26] Plattform Industrie 4.0, Die Verwaltungsschale im Detail, 2019, https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/verwaltungsschale-im-detail-pr%C3%A4sentation.html

[27] Plattform Industrie 4.0, Industrie 4.0 Sprache, https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/hm-2018-sprache.html

[28] Plattform Industrie 4.0, Wegweiser KI In der Industrie 4.0: Orientierung, Anwendungsbeispiele, Handlungsempfehlungen, 2020, https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/ki-in-der-industrie-4-0-orientierung-anwendungsbeispiele-handlungsempfehlungen.html

[30] Rapidminer, https://rapidminer.com/

[31] Reference Architecture Model Industrie 4.0, https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.html

[32] C. Rupp und die Sophisten, Requirements-Engineering und Management – Professionelle, iterative Anforderungsanalyse für die Praxis, 4. Auflage, 2006

[33] C. Sauer, H. Eichelberger, A. Ahmadian, A. Dewes, J. Jürjens, Aktuelle Industrie 4.0 Plattformen – Eine Übersicht, IIP-Ecosphere Whitepaper IIP-2020/001, 2020

[34] K. Schmid and I. John, A Customizable Approach To Full-Life Cycle Variability Management, Science of Computer Programming 53(3), 259-284, 2004

[35] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership Inference Attacks against Machine Learning Models, IEEE Symposium on Security and Privacy, 2017

[36] H. Stichweh, C. Sauer, H. Eichelberger, IIP-Ecosphere Platform Requirements (Usage View), Version 1.0, Januar 2021, IIP-2021/001

[37] VDI Statusreport, Simulation und digitaler Zwilling im Anlagenlebenszyklus – Standpunkte und Thesen, Februar 2020

[38] VDI/VDE 2193 Blatt 1/2, Sprache für I4.0-Komponenten, https://www.vdi.de/richtlinien/details/vdivde-2193-blatt-1-sprache-fuer-i40-komponenten-struktur-von-nachrichten, https://www.vdi.de/richtlinien/details/vdivde-2193-blatt-2-sprache-fuer-i40-komponenten-interaktionsprotokoll-fuer-ausschreibungsverfahren

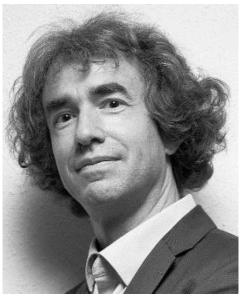
[39] Wikipedia, Anonymisierung und Pseudonymisierung, https://de.wikipedia.org/wiki/Anonymisierung_und_Pseudonymisierung

[40] ZVEI, Beispiele zur Verwaltungsschale der Industrie 4.0-Komponente – Basisteil, 2016, https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2016/November/Beispiele_zur_Verwaltungsschale_der_Industrie_4.0-Komponente_-_Basisteil/Beispiele-Verwaltungsschale-Industrie-40-Komponente-White-Paper-Final.pdf

[41] VWS vernetzt, https://vwsvernetzt.de/

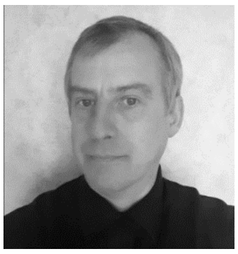
[42] R. Wöstmann, P. Schlunder, F. Temme, R. Klinkenberg, J. Kimberger, A. Spichtinger, M. Goldhacker, J. Deuse, Conception of a Reference Architecture for Machine Learning in the Process Industry, in Intl. Conference on Big Data, 2020.

About the Authors

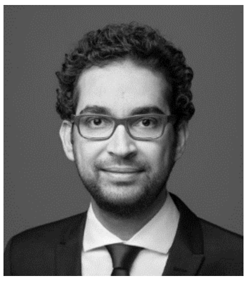


Dr. Holger Eichelberger is deputy head of the Software Systems Engineering group at the Institute of Computer Science at the University of Hildesheim. He conducts research in the areas of software product lines, model-based engineering, performance monitoring, and performance analysis. In particular, he is interested in the integration of these areas to create adaptive software systems. In IIP-Ecosphere he leads the think tank "Platforms" as well as the AI Accelerator. He studied computer science at the University of Würzburg, where he received his PhD on the automatic layout of UML diagrams.

Fotograf: Daniel Kunzfeld



Dr. Christian Severin Sauer is a postdoctoral researcher in the Software Systems Engineering group at the Institute of Computer Science, University of Hildesheim. His research interests focus on domain knowledge elicitation and modeling for explanatory and context-sensitive AI applications. He studied at the University of Hildesheim and received his PhD in Computer Science from the University of West London. During his PhD, he investigated and developed methods for knowledge elicitation and knowledge modeling for explanatory and context-sensitive AI applications.



Dr. Amir Shayan Ahmadian is a postdoctoral researcher at the Faculty of Computer Science at the University of Koblenz-Landau. His research interests focus on the challenges of designing and implementing secure and privacy-friendly software systems as well as on the current developments in Industry 4.0. He studied computer science at the University of Paderborn and received his PhD in computer science from the University of Koblenz-Landau. During his doctorate, he developed a methodology to operationalize the principle of "privacy by design".

Michael Schicktanz graduated at the Technical University of Nuremberg: with a B. Sc. in Business Information Systems and at the Otto Friedrich University of Bamberg: with a M. Sc. in Business Information Systems. From 2015 to 2017, he was at Allianz SE & Allianz Life, US in the post-graduate program focusing on digitalization and modern software technologies. Since 2017 is with Siemens AG (Erlangen) where he is Project Portfolio Manager and Solution Architect.

Dr. Andreas Dewes holds a PhD in experimental quantum computing from the Sorbonne University of Paris and the French Nuclear Energy Agency (CEA). He has founded several software companies and is the CEO of KIProtect GmbH, which develops advanced technical software solutions for data protection and data security. Within IIP-Ecosphere, KIProtect GmbH is developing a solution for the secure and privacy-compliant use of industrial & IoT data together with the consortium project partners and associated companies.

Dr. Gregory Palmer is a PostDoc at the L3S Research Center. He completed his PhD in 2019 in the Department of Computer Science at the University of Liverpool under the supervision of Prof. Karl Tuyls and Prof. Rahul Savani. Throughout his PhD, Gregory developed a number of approaches that enable independent learners to overcome multi-agent learning pathologies within fully collaborative team games. He also worked with the HAL allergy group on automating the inspection of opaque liquid vaccines. Within IIP-Ecosphere, Gregory is responsible for leading the Data think tank, and works on robust Reinforcement Learning for industrial environments within the Think Tank AI and Manufacturing.

Dr. Claudia Niederée works as a research group leader at the L3S Research Center in Hannover. Before she joined L3S, she worked at Fraunhofer. She holds a Master in Computer Science and was awarded a doctoral degree in Computer Science by the TU Hamburg-Harburg. Her main research interests are AI, diversity in knowledge, digital forgetting, social media analysis and intelligent systems e.g. for intelligent production. She has published more than 100 scientific articles and papers.

1 Appendix: Requirements Request Form

This appendix contains original requirements request (not translated, slightly adjusted in language and or formatting) for the IIP-Ecosphere project parts, especially the demonstrators.

Szenario/AP: ________________________

Die Think Tanks (TT) und der KI-Accelerator benötigen Ihre Hilfe für die gerade anstehenden Planungsschritte. Hierfür haben wir einige Fragen zusammengestellt, deren Beantwortung uns bei unserer Arbeit sehr helfen würden. Dabei handelt es sich um Fragen, die vielleicht bei Ihnen bereits diskutiert wurden bzw. deren Antwort sogar ihrer Arbeit helfen könnte. Uns ist durchaus bewusst, dass es sich zurzeit hierbei um vorläufige Planungen handelt und sich die Details während der Umsetzung ändern können.

Umgekehrt hoffen sowohl der TT Plattformen als auch der KI-Accelerator, dass die Ideen und später die Verfahren und Technologien, die wir entwickeln, wiederum den Szenarien/Demonstratoren helfen werden, z.B., durch Wiederverwendung von KI-Bausteinen oder Diensten, die Sie nicht entwickeln können/werden. An dieser Stelle ist vielleicht etwas Fantasie gefragt, denn einige Fragen zielen auf ihre (zukünftigen) Wünsche, die wir gerne in unsere Planung mit aufnehmen (und falls möglich auch umsetzen) werden.

Wir bedanken uns für Ihre Unterstützung. Rückfragen bzw. Antworten auf die Fragen bitte einfach an Holger Eichelberger. Natürlich stellen sich der Think Tank „Plattformen“ bzw. der KI-Accelerator auch gerne bei Ihnen vor, auf Absprache, bei einem IIP-Ecosphere Event,…

Kurzbeschreibung des Szenarios, insbesondere mit Übersichtsbild (z.B., Komponenten), ggf. schrittweisem Ablauf

Bitte geben Sie uns eine Kurzbeschreibung des Szenarios/Demonstrators basierend auf dem aktuellen Planungsstand, sodass wir das geplante Ziel und die Einsatzumgebung besser verstehen und einordnen können. Dabei wäre für uns ein Übersichtsbild oder sogar ein schrittweiser Ablauf sehr hilfreich, auch um die Interaktionen (besser) zu verstehen. Ein Übersichtsbild könnte beispielsweise eine Art Entwurfsdiagramm z.B., basierend auf Komponenten oder Services sein.

- …

Welche Inputs werden (basierend auf Übersichtsbild, z.B., mit Daten, Volumen, Frequenz, Typ/Protokoll) auftreten?

Basierend auf der Kurzbeschreibung bzw. dem Übersichtsbild, welche Eingaben sind im Szenario/Demonstrator angedacht. Dabei wäre es für uns hilfreich zu verstehen, welche Arten von Daten in welcher Geschwindigkeit bzw. Größe auftreten könnten. Uns ist durchaus bewusst, dass es sich hierbei zurzeit um grobe Schätzungen handeln könnte…

- …

Welche Outputs (basierend auf Übersichtsbild, z.B., mit Daten, Volumen, Frequenz, Typ/Protokoll) werden auftreten?

Basierend auf der Kurzbeschreibung bzw. dem Übersichtsbild, welche Ausgaben sind im Szenario/Demonstrator angedacht. Dabei wäre es für uns hilfreich zu verstehen, welche Arten von Daten in welcher Geschwindigkeit bzw. Größe auftreten könnten. Uns ist durchaus bewusst, dass es sich hierbei zurzeit um grobe Schätzungen handeln könnte…

- …

Verwendete KI-Verfahren/Komponenten/Frameworks

Ein Ziel der IIP-Ecosphere Plattform und des KI-Accelerators ist es, einen KI-Baukasten (easy-to-use AI) für die intelligente Produktion zu entwickeln, der die Umsetzung zukünftiger KI-Anwendungen erleichtert. Hierzu wäre es für uns bereits jetzt hilfreich, zu verstehen, welche KI-Verfahren bei Ihrem Szenario möglicherweise zum Einsatz kommen. Gibt es für diese KI-Verfahren vielleicht bereits Ideen, ob existierende Komponenten bzw. Frameworks hierbei als Grundlage verwendet werden sollen?

- ….

Welche IT-Ressourcen (GPU, Cloud, Edge, PLC-Edge, … mit grober Beschreibung) sind geplant?

Welche IT-Ressourcen (mit Rechenkapazitäten) werden in Ihrem Szenario/Demonstrator wahrscheinlich vorkommen. Die Ressourcen führen beispielsweise KI-Verfahren aus oder speichern Daten. Dabei darf es sich um lokale Ressourcen nahe an der Produktion handeln (Steuergerät mit Edge-Funktionalität, Edge, oder auch Maschine) oder auch um eher zentrale Ressourcen (GPU-Cluster, private/hybride/öffentliche Cloud) handeln. Für die geplanten Ressourcen wäre für uns die Anzahl und eine grobe Beschreibung hilfreich.

- …

Sind lokale KI-Verfahren/Vorverarbeitung (nahe an der Produktionstechnik) hilfreich oder sind nur zentralisierte Verfahren geplant?

Ein Dienst, den die IIP-Plattform wahrscheinlich anbieten wird, ist es Komponenten auf (nahezu) beliebigen Ressourcen mit Rechenkapazitäten zu verteilen. Dabei versuchen wir sowohl verschiedene Hersteller zu unterstützen, relevante Standards einzusetzen (z.B. die Verwaltungsschale) als auch flexibel zu sein, d.h., KI-Verfahren sollen nur dort eingesetzt werden, wo sie auch Sinn machen. Angenommen, ein derartiger Dienst würde zur Verfügung stehen. Wäre dieser Dienst für Sie (Ihr Szenario/Ihren Demonstrator oder auch allgemein) hilfreich? Falls ja, was würden Sie Sich von einem derartigen Dienst wünschen?

- ….

Welche bereitgestellten Dienste der IIP-Ecosphere Plattform würden helfen, dass KI-Anwendungen schneller, einfacher, besser umzusetzen?

Jetzt ist etwas Fantasie erforderlich… Die IIP-Ecosphere Plattform soll verschiedene hilfreiche Software-Dienste zur Verfügung stellen, die Ihre Anwendung unterstützen, verbessern, sicherer oder auch flexibler machen. Initiale Beispiele sind bereits zum Kickoff angeklungen: Datenintegration, Sicherheit oder Anonymisierung/Pseudonymisierung von Daten. Welche Dienste wären für Sie (im Szenario oder allgemein) sonst noch hilfreich? Welche Funktionalitäten bzw. Eigenschaften sollten derartige Dienste (auch die oben genannten) haben, damit diese für Sie hilfreich werden?

- ….

Wenn Dienste/Eigenschaften zur Laufzeit veränderbar wären, welche wären für Sie hilfreich/profitabel/wünschenswert?

Und jetzt wird es noch etwas wilder… Die Produktion ändert sich üblicherweise über die Zeit. Hierbei könnte Ihnen die Plattform helfen, wenn sie hinreichend Flexibilität erlauben würde. Ein Beispiel für den oben genannten Dienst, der KI-Komponenten auf die verfügbaren Ressourcen verteilt, könnte sein, dass *die Verteilung zur Laufzeit überprüft und optimiert wird, natürlich möglichst ohne negative Effekte zu verursachen. Generell kann diese Flexibilität durch manuellen Eingriff angefordert werden oder die Plattform könnte sie auch selbstständig auslösen. Wäre das für Sie hilfreich/wünschenswert? Auf welche Eigenschaften oder Dienste würde das aus Ihrer Sicht zutreffen (gerne mit Beschreibung)?*

- ….

Werden personenbezogene Daten verarbeitet? Welche Arten von personenbezogene Daten und in welchem Verarbeitungsprozess werden diese verarbeitet?

Mögliche Hinweise aus der DGSVO um Datenpfade in der Plattform zu planen.

DSGVO: „personenbezogene Daten“ alle Informationen, die sich auf eine identifizierte oder identifizierbare natürliche Person (im Folgenden „betroffene Person“) beziehen…

DSGVO: „Verarbeitung“ jeden mit oder ohne Hilfe automatisierter Verfahren ausgeführten Vorgang oder jede solche Vorgangsreihe im Zusammenhang mit personenbezogenen Daten wie das Erheben, das Erfassen, die Organisation, das Ordnen, die Speicherung, die Anpassung oder Veränderung, das Auslesen, das Abfragen, die Verwendung, die Offenlegung durch Übermittlung, Verbreitung oder eine andere Form der Bereitstellung, den Abgleich oder die Verknüpfung, die Einschränkung, das Löschen oder die Vernichtung.

- ….

Welche technische oder organisatorische Maßnahmen (wie z. B. Pseudonymisierung) sind vorhanden bzw. denkbar, um Datenschutzgrundsätze (DSGVO Artikel 5 – wie z. B. Datenminimierung) wirksam umzusetzen?

Um Datenpfade sicherer zu gestalten, können (erweiterte, innovative) Mechanismen zum Datenschutz eingesetzt werden. Welche Bedarfe und Möglichkeiten sehen Sie im Rahmen Ihres Demonstrators/Szenarios bzw. Ihrer Arbeit.

….