VDI/VDE-Gesellschaft
Mess- und Automatisierungstechnik

Status Report

Industrie 4.0 Service Architecture

Basic concepts for interoperability

November 2016

Preamble

Within the Technical Committee of the GMA-FA 7.21, it is the objective of the Working group "Service Architecture" to define the common characteristics of Industrie 4.0 Service Systems. The working group was established late 2015 and intends to deliver its final results by early 2017 in the form of a VDI Standard. This concepts presented in this VDI Status Report represent the results of initial discussions within the working group. Toward a complete and generally approved standard, the content of this status report is exclusively meant to be a basis for discussion among the various committees within the Plattform Industrie 4.0.

Düsseldorf, November 2016

Prof. Dr.-Ing. Ulrich Epple
Head of the Technical Committee 7.21 „Industrie 4.0" of the VDI/VDE Society Measurement and Automatic Control (GMA)

Dr. rer. nat. Dirk Schulz
Head of the Working Group „Service Architecture" within the Technical Committee 7.21

Authors

Thomas Bangemann, ifak e.V. Magdeburg

Christian Bauer, Siemens AG, Karlsruhe

Heinz Bedenbender, VDI, Düsseldorf

Annerose Braune, TU Dresden, Dresden

Christian Diedrich, ifak e.V., Magdeburg

Markus Diesner, MPDV Mikrolab GmbH, Mosbach

Ulrich Epple, RWTH Aachen

Filiz Elmas, DIN, Berlin

Jens Friedrich, ISW Uni Stuttgart, Stuttgart

Florian Göbe, RWTH Aachen, Aachen

Thomas Goldschmidt, ABB AG, Ladenburg

Sten Grüner, RWTH Aachen, Aachen

Martin Hankel, Bosch Rexroth AG, Lohr

Roland Heidel, Kommunikationslösungen e.K., Kandel

Klaus Hesselmann, Your Expert Cluster GmbH, Zirndorf

Guido Hüttemann, WZL, Aachen

Matthias Klein, Universität Stuttgart, IAS, Stuttgart

Ulrich Löwen, Siemens AG, Erlangen

Julius Pfrommer, Fraunhofer IOSB, Karlsruhe

Ursula Rauschecker, Fraunhofer IPA, Stuttgart

Miriam Schleipen, Fraunhofer IOSB, Karlsruhe

Dirk Schulz, ABB AG, Ladenburg

Timur Tasci, ISW Uni Stuttgart, Stuttgart

Mario Thron, ifak e.V., Magdeburg

Thomas Usländer, Fraunhofer IOSB, Karlsruhe

Clemens Westerkamp, HS Osnabrück, Osnabrück

Martin Wollschlaeger, TU Dresden, Dresden

Contents

1 Summary

This status report provides the foundation for a reference model of the Industrie 4.0 (I4.0) Service Architecture. This is the starting point for an implementation concept of the I4.0 Asset Administration Shell. This concept hinges on a technology-independent set of services to interact with I4.0 Components, a technology-independent way to describe information, and an approach to map the former onto specific M2M technologies.

It provides a classification of services according to the RAMI4.0 Layers that they relate to (data transport on the communication layer, information access on the information layer, platform administration services on the functional layer).

It provides and outlines the building blocks for the structured, distributed information models, and proposes means to express data semantics within these models (through semantic tags or references).

In conclusion, it provides a definition of the asset administration shell from the perspective of a service-oriented system architecture. It submits that it is possible to define the data and functionality of assets and how this information is accessed in a technology-independent manner. It de-couples the life-cycles of information and technology, leading to a sustainable overall system architecture. By then mapping these service and modeling concepts to technologies like OPC UA, it becomes possible to implement I4.0 Asset Administration Shells in a modular manner, and it helps to achieve semantic interoperability between the I4.0 Components.

2 Motivation and context in RAMI 4.0

2.1 Motivation

With the Reference Architecture Model I4.0 (RAMI 4.0) ([1] and DIN SPEC 91345) and the I4.0 Administration Shell [2; 3], metaphors for the core concepts of I4.0 have been described. The task of the administration shell is no less than to expose the data and functionality of all assets – including products and entire production systems – in a standardised manner. The administration shell thus is the key feature of the RAMI 4.0 Information Layer. For brevity, the abbreviation AAS is used for the asset administration shell.

Toward actual implementations of the AAS, the interfaces to access its content must be defined on a technical level. This includes both the information models used to represent the data and functionality through which these models are accessed in a distributed system. This means topics like data semantics, integration of existing information standards, and seamless communication of data in a distributed system must be considered. For these topics, the GMA FA 7.21 closely cooperates with the respective Plattform I4.0 Working Groups.

It is the task of the service architecture to generally define the ground rules for interoperability. Besides offering services to access data, its main contribution is the definition of an I4.0 Interoperability Information Model, which allows all providers of assets and services to describe their semantics in a common way, paving the way for the concurrent, agile standardisation of data semantics based on this model. Eventually, this will allow for addressing data by their semantics instead of their (memory) addresses.

While M2M technologies such as OPC UA are very promising, still a service architecture should take on a technology-independent perspective. This allows for decoupling the service system largely from specific protocols – it is even conceivable to couple systems based on different M2M technology. It also addresses the challenge of system evolution, where technology is upgraded or replaced over time. In both cases, interoperability between the participants of the service system – the I4.0 Components and their AAS – is best supported by abstracting from specific technology and defining the services on a conceptual level. Mapping the conceptual services onto selected technology becomes a separate concern.

2.2 Scope

The AAS defined by previous I4.0 Publications [1; 5] is the main entry point for exposing data elements as well as services in an I4.0 System. Thus, it is also the most important member in the I4.0 Service Architecture.

The main objective of the service architecture is to enable the implementation of the I4.0 AAS for interoperability and seamless data access by specifying a reference model for the information layer:

To this end, the service architecture needs to address three layers of the “RAMI” as indicated in Figure 1:

- information layer

  Derived from the RAMI 4.0 [1], the AAS itself is implemented on the information layer of the RAMI (making available data and functionality of the adjoining layers).

- communication layer

  In an operational implementation of the AAS, a communication layer is required to provide data transport mechanisms with defined service quality for the needs of the information layer as transparently as possible.

- functional layer

  To ensure interoperability, all implementations of the functional layer must comply with the specifications of the service architecture.

While formally the AAS only exposes data and functionality of assets, sometimes services (to interact with the AAS) and RAMI4.0 Layers (containing the implementation of the exposed features) are used synonymously.

In consequence, Table 1 defines the scope of the I4.0 Service Architecture in relation to the RAMI 4.0 Layers and Security, also indicating where collaboration with other activities in the I4.0 Platform is needed to specify a full Industrie-4.0-detailed service architecture. This initial status report puts the main focus on the details of the information layer.

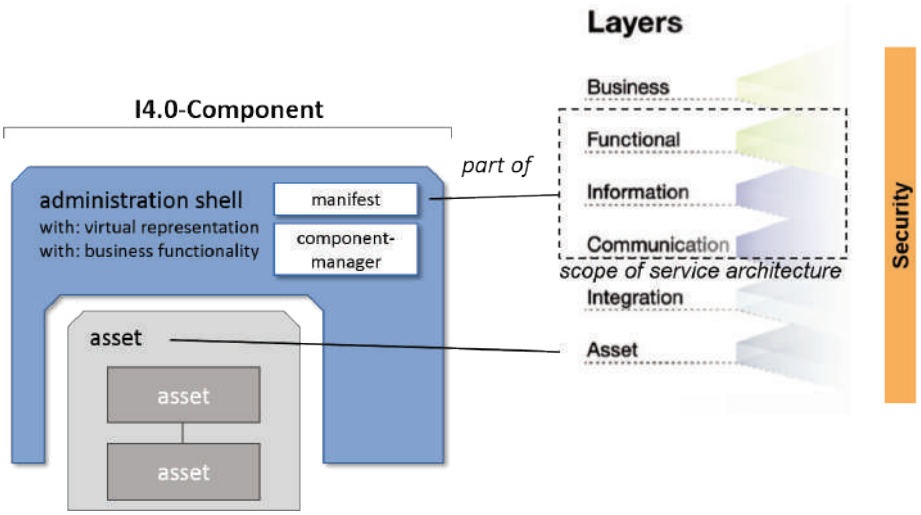

Figure 1. Relation of the asset administration shell and the corresponding RAMI layers

Table 1. Scope of the I4.0 Service Architecture

Layers and Aspect	In scope of service architecture	Out of scope of service architecture	Partners/Stakeholders
Functional layer	service composition and platform services related to information management	describing or implementing the actual functionality	suppliers of I4.0 Components (vendors)
Information layer	information models and semantic services (?), data access services, actual protocol selection/mapping	specifying domain- or application-specific information models, e.g. for MES, drilling, etc. Exceptions for information management itself?	any domain-specific user or supplier interest group such as FieldComm Group [9], NAMUR [10], etc.
Communication layer	negotiating on communication resources and service quality (I4.0 Networks, end-to-end connections, etc.)	implementation and specification of ISO/OSI layer 1 to layer 4	Plattform I4.0 AG1, UAG Network Communication
Security aspect	integrate security model into service calls	Authentication/trust = information layer security? Confidentiality/iIntegrity = communication layer security?	Plattform I4.0 AG3

Technically speaking, the service architecture aims at providing a common model layer between the meta-meta model of a middleware technology (e.g. OPC UA) and the domain/application-models (e.g. PackML (ISA-TR88), DriveCom [15]). The idea is to define a thin but strict foundation for the communication and information layers to ensure interoperability between components and their services.

The advantage in complying with this interoperability "sub-layer" is that it becomes possible for domain-specific interest groups to work independently from each other in an agile, massive parallel standardisation scheme without losing semantic interoperability when plugged into an I4.0-enabled system. This is the essence of Plug and Produce (PnP).

3 Asset administration shell and I4.0-compliant communication

3.1 Asset administration shell

The main objective of the AAS concept is to expose all data and functionality of an asset which are relevant in the life-cycles of this asset within an organizational scope e.g. an enterprise. From the perspective of the service architecture, the AAS of one asset as illustrated in Figure 2 is

1 the sum of all information on an asset represented by the information models (body)

2 which can be accessed through I4.0-compliant communication (services, see Section 2.2) and

3 which can be understood through an I4.0-defined semantics (manifest) or which follow a defined complementing data format (Industrie 3.0 Standards)

4 within a defined organizational scope, e.g. one enterprise

5 discoverable through a defined mechanism

6 based on common asset identification data (header)

7 regardless of the deployment of the individual views on (other) assets in that domain.

In this sense, the (sum of) information on an asset is provided by the participants of the service system in an organizational unit. While the original idea of the ASS focuses on the type and instance data which describe an asset, the focus of the service architecture is how to actually provide the illusion of a contiguous AAS from distributed service participants, when each is hosting just fragments of the entire AAS.

Conversely, this implies that information related to an asset which is exposed through I4.0-compliant communication by an (authenticated) service participant within an organizational domain may become part of the AAS simply by registering with the service system. When defining the service architecture in more detail, the focus needs to be on those service participants and their AAS fragments, not the I4.0 Component which is rather a resulting phenomenon.

While it is technically conceivable to deploy all fragments of an AAS on the represented asset itself, one can expect those fragments to be distributed across the organizational domain (actually, on other assets). For instance, an I4.0-compliant field device can implement an embedded information model, but still have a reflection of its parameters inside an FDI server [9], have its maintenance managed by a CMMS and its ordering information in an ERP system along with thousands of other assets. In fact, one can expect that the distributed deployment of ASS fragments illustrated in Figure 3 is the standard case. The assets a_1 and a_n indicate a native I4.0 Component (CP44), which exposes its data and functionality directly in an I4.0-compliant fashion. The a_2 represents a classic Fieldbus device (CP34), which is a managed entity and can actively communicate – just not in an I4.0-compliant manner.

The AAS, which turns an asset into an I4.0 Component actually emerges by service participants providing standardised access to certain views/fragments/aspects of asset-related information and functionality, enabled and controlled by a set of administrative services.

To foster agile standardisation, it might be wise to split and parallelize the migration of existing standards among the different user/interest groups. Overall, there are three basic options how to proceed:

1 a centralized approach within the I4.0 Platform, having few working groups try and define the entire set of AAS;

2 split the task by asset providers, e.g. manufacturers of field devices, controllers, package units;

3 split the task by providers of automation (sub-)systems, e.g. have the FieldComm group [9] define the ASS for field devices, or have CMMS vendors define the maintenance management[1)].

The first option seems outright unrealistic, but in any case a process is needed to actively establish governance over the AAS body from the centralistic I4.0 Platform. The second option might be attractive to

[1)] In reference to Figure 3, this can also be considered a split “by deployment”.

asset providers, but to actually achieve compatibility, the third option is the best choice. However, it comes at the cost of aligning the redundancies among the standards indicated in Figure 8 to define the AAS manifest.

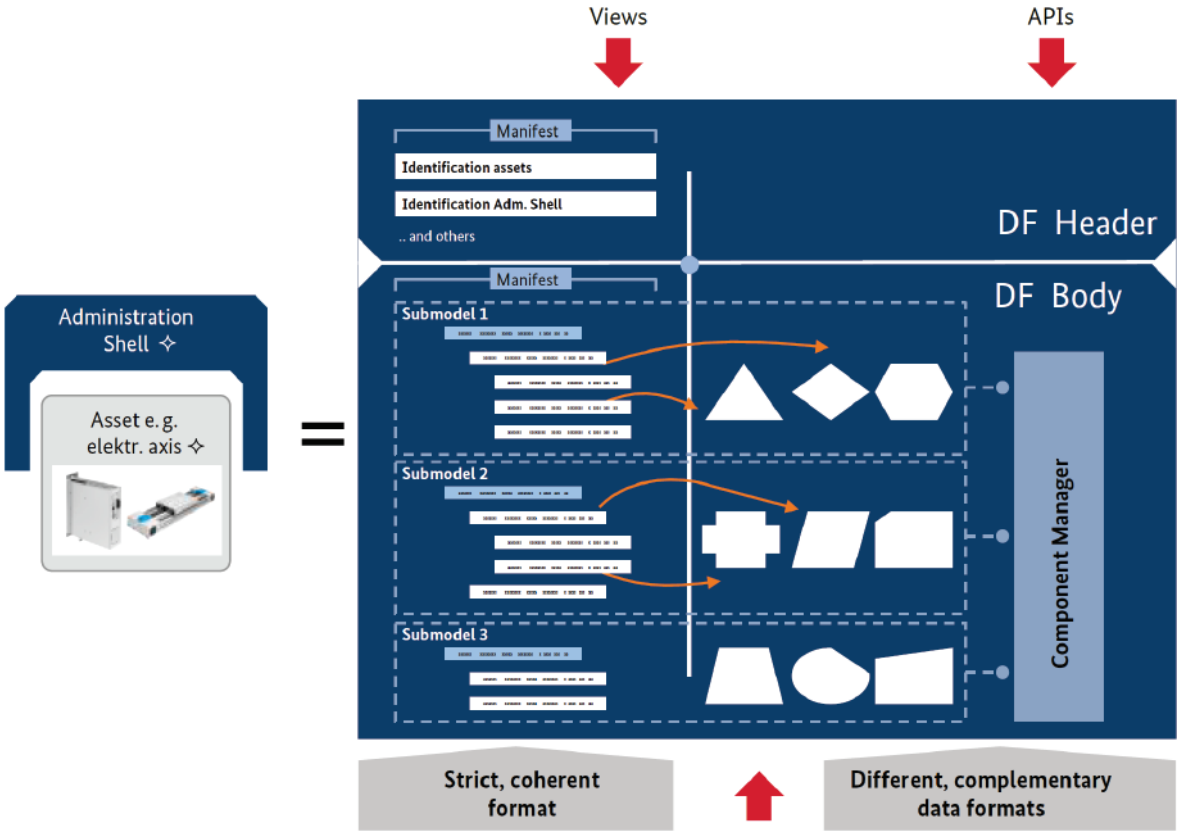

Figure 2. Standardised and complementing data in the AAS body (part of the information layer) [3]

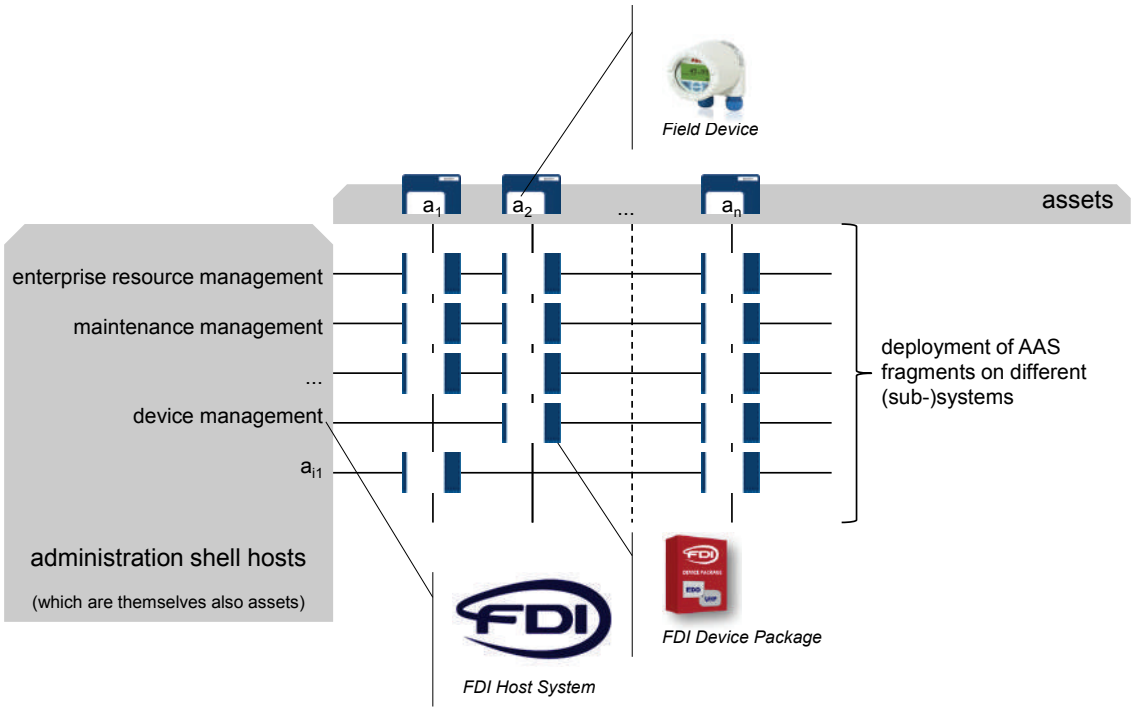

Figure 3. Distributed deployment of fragments of the AAS

3.2 I4.0-compliant communication

To participate in I4.0-compliant communication, an I4.0 Component needs to

- implement the information services defined in Section 3, based on common middleware/data exchange technologies (which have not officially been selected yet) and
- integrate with the communication layer (whose details are also still to be defined by the AG1, UAG Network-Based Communication) [5].

3.3 Service Architecture Reference Model for I4.0

A service-oriented interface to the I4.0 AAS is considered as one specification of a service architecture out of others in the context of I4.0. The basic concepts of I4.0 Service Architectures are specified in a so-called reference model for I4.0 Service Architectures (RM-SA) as proposed by the VDI Status Report “I4.0 – Auf dem Weg zu einem Referenzmodell “in April 2014 [14]. This section provides an overview about the motivation for such an RM-SA and the relationship of the AAS to it.

4 Service architecture reference model for I4.0

4.1 RM-SA overview

The German Standardisation Roadmap for I4.0 [12] suggests defining the following reference models:

- SA – System Architecture
- RT – technical systems and processes
- RL – control functionality
- RB – technical-organizational processes
- human tasks and roles

The RM-SA (Reference Model for I4.0 Service Architectures) is currently being specified in the DIN SPEC working group 16593. It describes the basic concepts and an approach towards a Reference Model for the first category (SA), however, focusing on the appliance of service-oriented architecture (SOA) as one of the technological cornerstones of the I4.0 Vision.

The purpose of the RM-SA is, among others, to define a precise meaning of the concept "service" in the context of I4.0. Looking at the RAMI4.0 and especially the three RAMI4.0 Dimensions, this means that I4.0 Services shall be applicable

- to all architectural aspects ("layers" see Table 1),
- to all hierarchy levels and, hence, all types of assets (from "product", "field device", "control device" up to the "connected world"), and
- to the whole life cycle of both asset types and asset instances.

The RM-SA is meant to be a meta-model for service-oriented architecture (SOA) specifications. It is an abstract framework that encompasses, among others, the

- identification and terminological definition of basic concepts and their relationships within the scope of the I4.0 application domain),
- rules and notations to be applied,
- viewpoints to be described when defining I4.0 Service Architectures, and
- compliance statements that define what has to be fulfilled when claiming compliance to the RM-SA.

The RM-SA enables the specification of architectures, and this enablement encompasses both the conceptual and the technology-oriented implementation level as illustrated in Figure 4. If such architectural specifications are open, based upon standards, solve commonly agreed requirements in a given domain (possibly derived from use cases, including their non-functional requirements), and may be implemented in an efficient and cost-effective way, they may be classified as reference architectures.

The choice and combination of SOA design patterns (e.g. publish/subscribe or publish/find/bind) determine the architectural style that is supported, e.g. event-driven, request/reply or resource-oriented (representational state transfer (REST)). The RM-SA will define a list of SOA design patterns and their relationship to architectural styles in order to streamline the discussion of I4.0 Service Architectures.

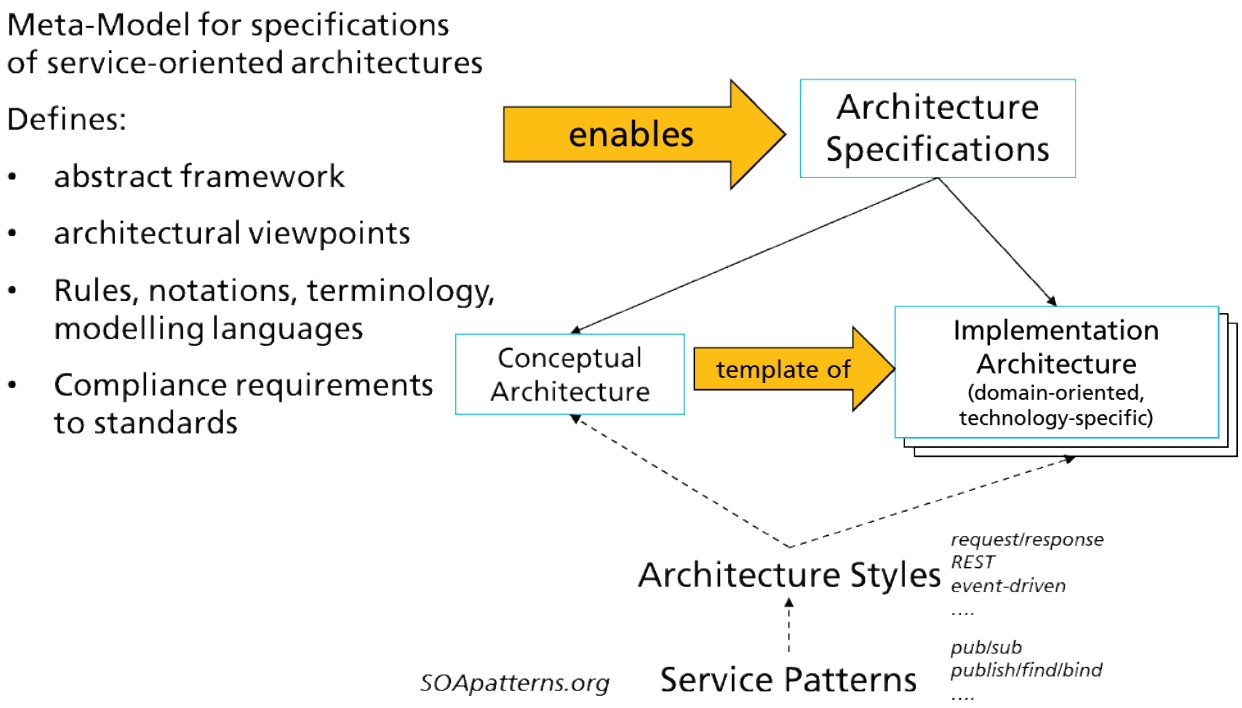

Figure 4. Reference models enable architecture specifications

On the one hand, the current draft of the RM-SA follows the service specification of the OASIS Reference Model for SOA [13] which defines a service considers to be a "mechanism to enable access to one or more capabilities where the access is provided by a prescribed interface and is exercised consistent with constraints and policies as specified by the service description".

Applied to the AAS, this means that an AAS service is a "mechanism to enable access to one or more AAS capabilities where the access is provided by a prescribed AAS interface and is exercised consistent with constraints and policies as specified by the service description in the AAS manifest".

On the other hand, the RM-SA broadens the view towards protocols as used in peer-to-peer architectures. It tries to define common conceptual elements (e.g. interactions) supporting both the "service" as well as the "protocol" architectural approach.

4.2 Service Hierarchy

The I4.0 Service Architecture distinguishes between four different types of services that are built on top of the actual communication infrastructure (cf. Figure 5) and aligned to the three layers communication, information and functional as introduced see Table 1. The first set of services – **the communication services** – define the primitive services required to perform data transfers such as (connect, transmit, etc.). Additionally, these services define how quality of service (QoS) between the communication participants can be negotiated and ensured. The communications services themselves are defined in a technology independent way in order to allow mappings to concrete communication technologies such as OPC UA or DDS. This layer has not been specified in detail, yet. The services dealing with QoS will be handled in future work. The communication services represent the functionality of the communication layer (see Table 1).

On top of the communication services the "service atoms" for information access – **the information services** - are located. These services define the basic functionality required to work with information models, such as read, write, create, or delete. This status report focuses on these services which are detailed in Section 5. However, it is again important to keep in mind that the information service also do not define higher-level semantics but only a common vocabulary for interaction with data. Even though they are also technology-independent, they were defined to be easily mappable to major technology candidates such as OPC UA. The information services represent the functionality of the information layer (see Table 1).

Using these service atoms, higher-level services are built. This includes **the platform services** that define the self-management functionality of an I4.0 system and **the application services** which provide the actual functionality for building productions systems (e.g., a welding service).

Whereas the platform services are designed to be domain-agnostic and provide generic functionality such as discovery or localization, the application services are different for each (sub-)domain participating in I4.0.

As depicted in Figure 5, the service architecture is part of the RAMI 4.0 information and communication layers, but it exposes the features of the communication, information, functional, and business layers. The service system is technology-independent (abstract), it defines the mechanisms for providing the semantics of data, but it does not define the semantics of specific data content itself. Thus, it does not reside on the higher, function and semantic bearing layers whereas the functionality and services built with it would reside on the upper layers (functional and business) of RAMI.

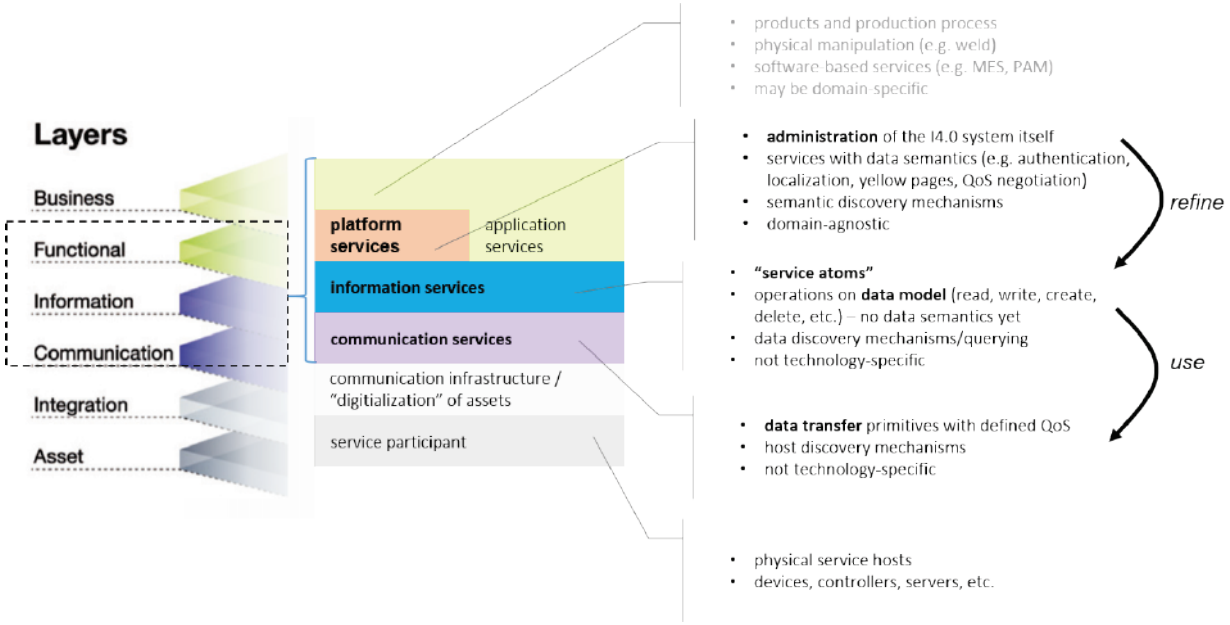

Figure 5. The I4.0 Service Hierarchy

4.3 Description of the I4.0 Service System

Previous publications on I4.0 either focused on the high-level reference architecture (RAMI 4.0) [1] or the description of particular parts of the I4.0 ecosystems such as the I4.0 Component and the ASS [4]. However, in order to build a service architecture a more concrete specification on how these things relate to each other is required. Thus, Figure 6 depicts a more formal overview model of the already defined artifacts of I4.0 and how they relate to each other. The elements described in this model are based on the official articles on I4.0 mentioned above as well as the glossary defined by GMA FA 7.21 that is extracted from these publications and agreed on in the consortium [7].

As depicted in Figure 6 the top level elements of the I4.0 Service Architecture are defined by the I4.0 Platform, I4.0 System(s) and the actual I4.0 Service System. An I4.0 Platform defines the basic and standardised communications and system infrastructure that enables the efficient creation of I4.0 Systems in a particular domain. Thus, actual I4.0 Systems are based on such a platform. The system itself is comprised of I4.0 Components which in turn can be systems themselves. An I4.0 Component is defined as the combination of one or more assets together with their ASS(s). There can be multiple administration shells per asset as e.g., one might reside at the manufacturer's side and one in the organization where the asset is actually used.

The I4.0 Service System now flanks this hierarchy of elements and brings in the service orientation side of the I4.0 System. There might be different notions of a service system depending on the scope to which services need to be exposed. For example, a site-internal service system would probably involve a different set of services than a global I4.0 Service System. An I4.0 Service System comprises of a set of I4.0-compliant services that have a defined quality of service (QoS) specification. These services are provided and/or used by I4.0 Service System Participants. The most prominent of these participants are naturally the services offered by the ASS of the I4.0 Components that take part in the system. However, also human actors or tools that are no I4.0 Components (e.g., for engineering systems or tools) can act as I4.0 Service System Participants by interacting with these services.

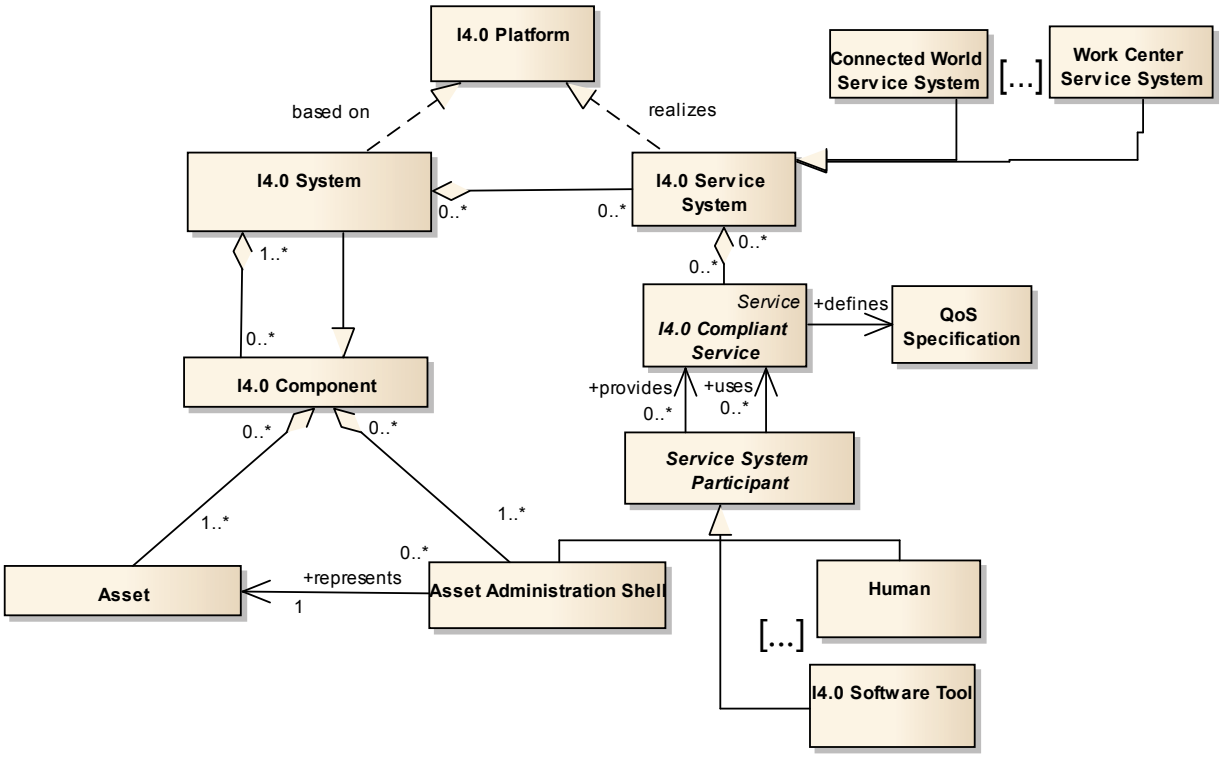

Figure 6. Overview model of the I4.0 Service System

5 Information services

Information services provide the basic mechanisms for data access and manipulation. To be stable under the evolution and replacement of middleware technologies, and to allow interoperability between different middleware technologies, information services are defined in a technology-independent manner. They can be considered as the conceptual, technology-independent interface of the information layer, which is then mapped onto technology-specific protocol(s) such as OPC UA. Applications, i.e. the elements of the I4.0 Functional Layer, can be defined against a kind of technology abstraction layer (this is a core feature of the information layer).

5.1 Service signature

The service calls have a common signature as described in Table 2. The functionality of each call is described by the data operation, the target address, and the accessed information content. In addition, there are non-functional aspects like the service quality, which mostly define the expectations on the underlying communication layer, see DIN EN 81346-1and [6].

The specific operations are explained in Table 3 The given information payload already indicates how each operation is responsible for one specific type of entity in the information meta-model described in Section 5.2. On a technology-independent level publish/subscribe may be considered as a mere read/write operation on topic identifiers. However, it is expected this type of data distribution to become very relevant in distributed, self-configuring I4.0 Systems and therefore these dedicated (convenience) operations are proposed. The details of entity addresses in a distributed system is subject of Section 4.2.

Instead of defining different operations for unconfirmed and confirmed (acknowledged) types of communication, an **RSVP flag** is proposed to indicate if the service should be executed with or without a confirmation. As stated in Table 3, some operations such as read on data elements naturally require a response to be sent.

As stated above, the service quality (**QoS**) includes expectations on the I4.0 Communication Layer whose services are discussed in Section 7. The quality of services on application/functional level is not within the scope of this document. Toward the communication layer, it is presumed that this layer offers a transmit and a receive service, which are configured using the target address and QoS parameters from the information service call. Defining a reference architecture for the communication layer is subject to the Platform I4.0 AG 1, UAG (sub-working-group) Network-Based Communication (DIN EN 81346-1), but the typical quality indicators are expected to be supported:

- availability and reliability[2)]
- fail over time
- latency, cycle or round-trip time[3)]
- update frequency, maximum age of information
- required bandwidth (per cycle)
- maximum acceptable jitter

In any case, the deployed physical communication resources limit what can be configured in software, availability and latency considerations must be taken into account already at network design time. It makes great sense to be able to query the communication layer for the maximum configurable quality of service properties.

For security, it is important to distinguish between

- Authentication and authorization, i.e. confirming the (claimed) identity of the service provider/consumer and enforcing access rights accordingly. In the service hierarchy, authentication is a good example for a higher-level platform service (Section 6).
- Confidentiality and integrity, i.e. ensuring that information is never disclosed to any 3rd party and that manipulations by 3rd parties can always be detected (if not avoided). Data confidentiality an example for a function of the communication layer (Section 7).

Defining the details of secure identities and confidential communication, in particular in a long-term sustainable manner, is subject to Platform I4.0 Working Group 3, Security [6].

[2)] In practice, a redundancy factor of the underlying infrastructure might be much more usable. Still, it must be considered that the asset (data source) itself may not be redundant to begin with.

[3)] From the perspective of the functional layer, latency-related properties would need to include processing time in the end-points. It must be considered that e.g. process instruments may only support measurement cycles far lower than what a network is capable to deliver.

Any service call invariably takes place in a determined context. Following the examples in Table 4, this context is formed by the environment of the service participant. Some aspects like the used I4.0 Access Network are implicitly given, while the scope for service discovery may be configured by the application or user. Standardizing the view on this service context is a matter of security and convenience for service orchestration in I4.0.

Table 2. Signature of information service calls

Signature (o = optional)	Description
Operation	What is being done to the entity at the given address, using the specified information payload
Target addressing	identifying (and eventually addressing) the target of the service operation, i.e. attributes to set, parent objects to receive a new child object
Information payload	the information to be transferred, i.e. attribute values to assign, object types to instantiate
RSVP flag (o)	indicate whether a response/acknowledge on the information layer is expected [a)]
Context (implicit)	the context in which the service is called, i.e. the application, user, the local I4.0 Network, a global (discovery) scope
Context QoS (o)	The expectation on the reliability and timeliness that the communication layer needs to provide for this service call.
Context security (o)	the expectation on the confidentiality of the end-to-end communication[b)]

a) This is unrelated to the implementation of the transport layer (layer 4), where UDP (connection-less) or TCP (connection-oriented) might be used.

b) Note that the authentication of communication partners is considered to be a higher-level platform service.

Table 3. Service operations

Operation	Description	Addressed entity[a)]	Response (see Table 2)
Read/Write	get or set values of variables existing within the (structured) information model	data elements	value (mandatory for read), confirmation (optional for write)
Publish/ Subscribe	get or set content of topics (like read/write, but the target address is a topic)	data elements	N/A
Create/ Delete	insert or remove object instances into the information model (including the creation of references to structure the model)	objects, references	optional
Browse	traverse information model	references	object(s)
Method call	transport a complex payload whose contents are interpreted by business-logic, i.e. unlike a read or write operation, the effects are method-specific	services	optional, depends on semantics of the method call

a) see Section 5.2

Table 4. Examples for contexts of service calls in I4.0 Systems

Context facet	Example
Application	augmented reality application during device replacement
Users	service engineer
I4.0 Access Network	local I4.0 Network #123
Discovery scope	plant-local
System state	operation vs. maintenance phase of the process

In conclusion, quality of service, authentication and communication-layer security, along with the service context need further discussion. This requires close collaboration between I4.0 Platform Working Groups 1 and 3 and GMA FA 7.21.

5.2 I4.0 Information Meta-Model for Interoperability

One of the major efforts of I4.0 is to ensure interoperability between interacting systems. Thus, the service architecture is defined on a level which allows the integration of its underlying technical systems as good as possible. This is achieved by defining a technology-independent vocabulary for these services. On a formal, yet abstract, level this vocabulary forms a common information meta-model.

It is defined on an abstract level so that it provides sustainability under evolution. It basically aims at the “extinction” of technologies at an abstract level. Still, the meta-model defines the core concepts for information modelling that can be mapped to underlying technology-specific (cf. Section 8) meta-models such as the OPC UA meta-model.

Object-orientation has proven to be a useful concept in order to describe information on an abstract level. Starting from abstract modelling languages such as the UML to domain specific information modelling languages such as OPC UA or concepts similar to IEC 61850 are used to define and structure information models. These conceptual models are mostly based on the same paradigms such as objects (classes, elements, etc.) that have references (associations, links, etc.) to each other and provide data elements (variables, attributes, fields, etc.) as well as services (methods, operations, etc.). Additionally, the standard [IEC 62832], on Digital Factory which aims at providing a standard repository for digitally representing automation assets, defines similar concepts such as automation asset classes, data elements and references which have been taken into account.

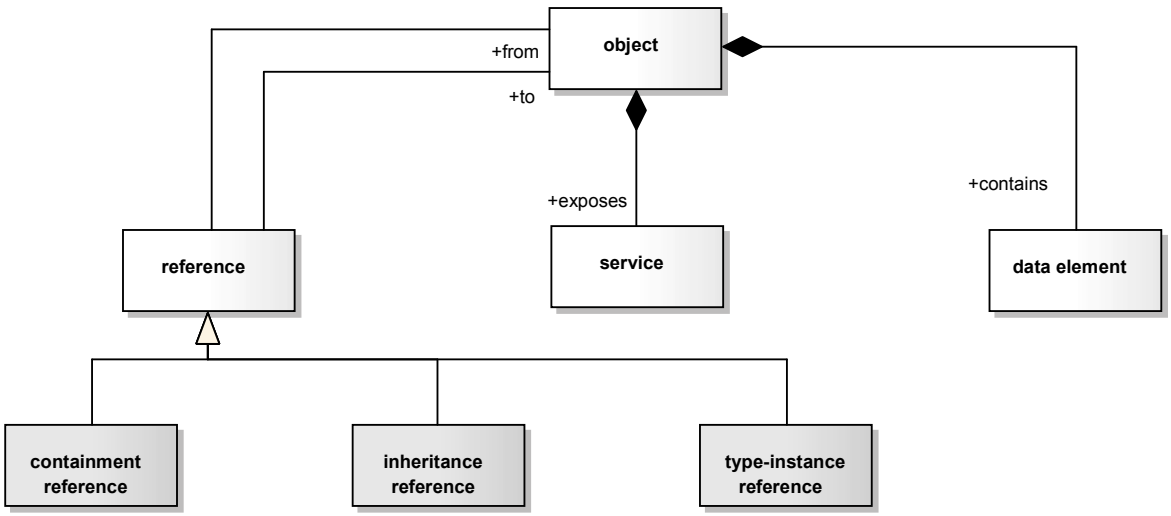

Figure 7. I4.0 Information Meta-Model [3]

More specifically, the following elements as depicted in Figure 7 make up the I4.0 Information Meta-Model:

- objects
  The main building blocks of information models that represent conceptual entities. It can be distinguished between objects that represent instances and those that represent types (or classes). Figure 7 does not explicitly distinguish between those concepts but implementations (such as OPC UA) of course can do this. However, a mapping back to object structures is still possible and we keep this generic concept in order to stay compatible to approaches not supporting this distinction.

- data elements
  holding and referring static as well as dynamic data values

- references
  Common rules for expressing relationships between model elements and particularly distributed models. Additionally, together with objects, references can be used for structuring the information models.

- service
  Services or methods define dynamic functionality that the objects expose. Note that this does not refer to the lower level service such as information and communication services but rather those on higher levels which actually represent the domain-specific functionality (application services).

The concept of objects with references can be used to describe extended use cases present in object oriented modelling/programming as depicted by the specific sub-types of reference in Figure 7:

- type instance relationship between object types and objects
  By using a special “is-type-of” reference it is possible to distinguish between object instances and types in an information model.

- inheritance/generalizations
  Is realized by using a specialized references that also have this fixed semantics. This also allows to build extended concepts like polymorphism on instance level.

- containment and aggregate structures
  Again, by defining specific references between entities aggregation semantics can be created. Additionally, by distinguishing between containment and aggregation references (as done e.g., in UML) also lifecycle handling can be declared.

- Other types of references can be represented by the generic reference element where the actual semantics then need to be defined in the information models introducing these (e.g., HasComponent, Organizes are defined in the OPC UA standard).

The meta-model presented above is capable to represent information in a structured way, complemented with the benefits of a type system. It is designed to allow a mapping to existing information meta-model standards in order to provide suitable implementations such as OPC UA or Digital Factory. To now specifically represent the information content of I4.0 ASS, should be proposed to consider further extensions to the meta-model. There is a particular need to be able to partition AAS information models with regard to purpose, source, and deployment/hosting (see Figure 3) of its content. While from an AAS perspective, such concepts are still being debated as views, perspectives, or sub-models, focus should be proposed on how to implement partitioning schemes, considering existing concepts such as aspect-objects (DIN EN 81346-1).

5.3 Addressing, identification, and semantics

As previously stated, an integral part of the service architecture are common rules for addressing, identification (Section 5.1), and understanding of the data exposed (see definition of interoperability in Section 3.1) in the information model (Section 5.2). The information model needs to reflect the where, who, and what of its elements, so to speak.

It is important to distinguish between address and identifier of information. Addresses may be transient, but describe the location of information at a particular point in time; they enable seamless data access in a globally distributed system. Identifiers needs to be immutable and always refer to the same object of interest. In particular, a fragment of an AAS does not have to directly be aware of its own address but only its identity. Providing access (via addresses) to the AAS of a particular asset (via identifiers) is then the task of the service system. For I4.0 Components to interoperate, the meaning of data must also be represented in the information model in a machine-readable form.

This is the topic of data semantics, where data elements in the information model are qualified through attributes that refer to a specific meaning that is presumed to be common knowledge. Data semantics is thus not about *defining* semantics but about *referencing* an existing definition. Arguably, the original definition can only be done in text form and is interpreted

by humans for the implementation of component behavior. The achievement of machine-readable data semantics is that two machines (components) implemented in such manner can interoperate without further human effort.

In summary, addresses, identifiers, and semantics are essential types of meta-data of any entity in the AAS. Conceptually, they all function as a kind of reference, either to other entities in any of the existing AAS or to agreed concepts outside of information models. They all can be realized using different entities from the I4.0 Information Meta-model in the following manner:

- addresses (referencing communication endpoints), implemented via local and global references
- identifiers (referencing the asset itself, its identity), implemented via designated identifier data elements,
- semantics (referencing an agreed and elsewhere defined meaning), implemented either by semantic identifier data elements or semantic references to an object whose meaning might be known or which again provide further semantic references that have to be resolved.

The I4.0 Service System must support the resolution of (semantic) identifiers to addresses, considering the context in which semantic data access is requested.

Data semantics as introduced before focuses on qualifying data with references items with identifiers of

What is pragmatically required in the next steps is an agreement on common practice address formats acknowledging the use of Internet technology. To address service hosts, DNS names or IPv6 addresses are good candidates. To address data within a host, local names or variable identifiers can be used. OPC UA already uses such a concept as illustrated in Table 5.

Machine-readable data semantics is the missing link in actually having M2M-type communication that allows instant interoperability or even self-configuration of I4.0 Components. This means:

- There must be a common way (meta-model) of expressing semantics in information models.
- It must be possible to integrate (re-use) the multitude of established model standards as illustrated in Figure 8, a demand also formulated as part of the ongoing AAS definition [12].

Table 5. Examples of local and global references to data points

Generalized address	Technology-specific example
Fully qualified dns host name + application + local variable identifier	opc.tcp://dms.local/FDI/ Objects/DeviceSet/TTH300/ParameterSet/PV

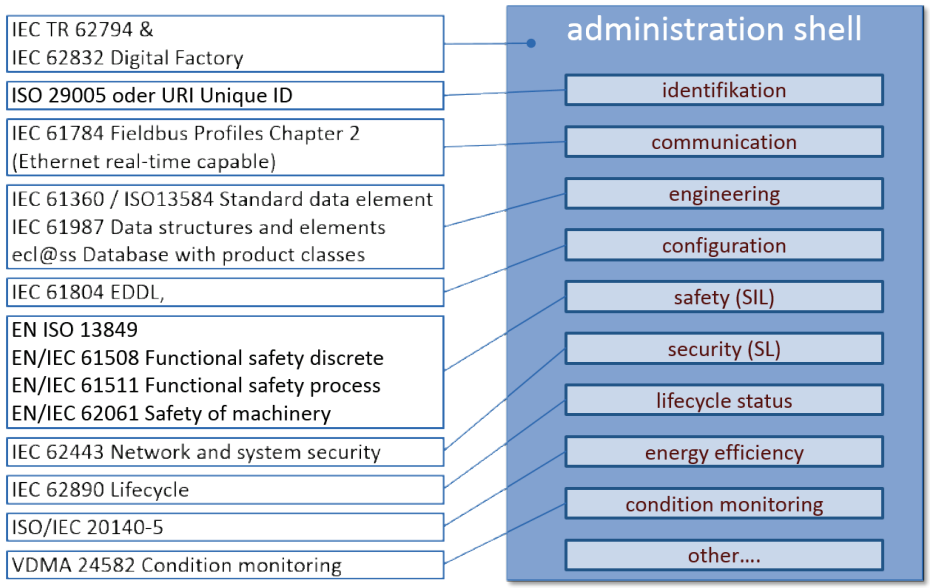

Figure 8. Examples of existing standards to be integrated into the AAS [12]

```
variable.semanticId[0]="urn:tag:eclass.org,2016:0173-1#02-BAB726#007"
variable.semanticId[1]="urn:tag:profibus.org,2106/names/PV_SCALE.EU_at_0%"
variable.semanticId[2]="urn:tag:profibus.org,2016/indices/11[0]"
```

Figure 9. Mock-up for data semantics defined in multiple "languages"

To be able to integrate semantic definitions from existing standards, a kind of "multilingualism" is needed for the same data element. A data element must be able to hold multiple references/identifiers (i.e. multiple "languages"), and each reference/ identifier must state the namespace in which its value is to be interpreted. It seems to be no viable option to base the entire definition of semantics in I4.0 only on object names or type systems.

This approach allows objects, data elements, or even references to be semantically defined using multiple (existing) dictionaries based on one common mechanism. Figure 9 illustrates this for the "lower range-limit of temperature" of a temperature transmitter.

This approach appears to be very effective, its simplicity promises easy adoption. However, it requires a clear governance of the used namespaces, which on one hand must be unique, but on the other hand belong to different existing organizations.

In this, the use of a type system (see also Section 5.2) has its benefit to efficiently construct address spaces (i.e. information model instances). However, a particular conclusion of multi-lingualism and multi-semantics is that it is probably not feasible to generally hinge the representation of semantics in I4.0 only on type models. Rather, the illustrated mechanisms of semantic tagging and references might be used, and a type reference can be one special case in this.

5.4 Addressing

For an information modeling in distributed settings, it is crucial to define some identification and addressing mechanism. However, in the area of IIoT (Industrial Internet of Things), many standards define or reference different technical solutions. In this section, we propose a common format for identification and addressing for the IIoT by adapting established web-standards. This common format is intended to bridge the "semantic borders" between standards and technologies.

Identifiers make recourse to a local or global authority for uniqueness. Within the scope of the authority, entities with the same identifier are assumed to be equal but not vice versa. That means there may exist multiple identifiers for the same entity. Identification authorities may claim to prevent such collisions by strict control over the identifier assignment process, but this cannot be assumed in general. Many identifiers are build up in a hierarchical manner, such as USB device identifiers or MAC addresses. Another example is ISO 29002-5 where identifiers start with an ISO-registered prefix for the registration authority. In the domain of web, uniform resource identifiers (URI, RFC 3986) are the dominant format for both identification and addressing. All valid URIs begin with a scheme prefix that defines the semantics of the following hierarchical part of the URI. A repository of registered scheme-prefixes is governed by the IANA organization and accessible at http://www.iana.org/assignments/uri-schemes/ uri-schemes.xhtml. URIs can make reference to the semi-centralized domain name service (DNS) for distributed authorities where the top level domains (TLD) are also governed by IANA and delegated to local authorities, e.g., DENIC eG for German domains with the country code ".de". The use of RFC 3986 URIs is proposed for all identification and addressing purposes. First, in the context of this publication, identifiers are defined as URIs without an authority part (that would be following after the double-slash). Identifiers may denote any entity, such as virtual resources, physical objects, theoretical concepts, and so on. Examples for authority-less URIs are **tel:+1-816-555-1212** and **urn:isbn:0-486-27557-4.** The latter example follows the urn URI scheme. It requires that the hierarchical part of the URI begins with a namespace shortname registered at IANA. An example for a registered namespace is ucode, a global identification scheme for physical resources [11] with ucode URNs, such as **urn:ucode:_0123456789ABCDEF01234567 89ABCDEF,** defined in RFC 6588.

For users who have not registered their own namespace, should be used the tag namespace (RFC 4151) followed by a tagging authority before the main hierarchical part. The tagging authority is qualified by a DNS-based TLD and a date of the tag. The uniqueness of the identifiers is assured by the authority at an explicit point of time. Therefore, tags are agnostic to the ownership transfer of TLDs. The hierarchical part after the tagging authority can be freely used as long as the general syntax rules for URIs are uphold. Examples for tag URN identifiers are:

urn:tag:companyA.org,2016:unit:subunit:position, urn:tag:companyB.org,2016:id4809234

identifying some position in companyA and some asset of companyB with a serial number, respectively.

A co-existence of different identification standards can be provided by specialized URI schemes, URN namespaces and tagging authorities. For example, ISO 29005-5 conform identifiers can be represented as an URN tag, such as:

urn:tag:iso.org,2015:ISO-29002-5:0173-1#02-BAA580#005,

which in this (ad-hoc and not normative) example points to the eCl@ss "nominal speed" property of a motor (2015 is the release year of ISO 29002-5). Note that ISO has registered their own URN namespace (RFC 5141). But it is currently only used to identify ISO Standards, such as **urn:iso:std:iso:9999:-1:ed-1:en**, and not for identifiers in reference to an identification scheme from an ISO standard. Locally scoped tagging authorities can be used to reference resources that are unique only in a local scope, such as

urn:tag:local.domain,2007-11-02:my-identifier:3456 .

Here, identifiers are only unique in the scope where "local.domain" maps to the same identification authority. This use is not well-aligned with RFC 4151 due to a non-guaranteed uniqueness outside of this local scope. However, the possibility of local scoping is essential for complex distributed systems in the IIoT domain. The context of scoped URIs can be defined by means of base URIs as described in RFC 3986. In conclusion, the URI-based approach enables nearly effortless decentralized identification of resources that can be combined with common industrial standards.

Second, addresses are defined as URIs with an authority part following the well-known scheme-and-doubleslash prefix (commonly known as URL), such as

http://server.name/index.html

opc.tcp://server.com:4840/ns=0,i=2253,
coap://local.server/bike/lock.

The difference to identifiers is that addresses can only denote virtual resources that can be encoded as data literals. Addresses contain enough information, such as the access protocol, to retrieve the current value of the referenced data literal. That is, modulo accessibility issues, an address can be used as stand-in for the resource representation it denotes. For unknown representation encodings, a fallback solution is to present the resource representation as a byte string. This is in line with the use of HTTP for RESTful interfaces where every response is encoded as a human readable string starting with an encoding format tag at the beginning. Since addresses make reference to a single resource representation, in case of the OPC UA example, it might be more sensible to denote not nodes but node attributes for more fine-grained access, such as **opc.tcp://server.com:4840/ns=0,i=2253#BrowseName**. From these examples, it becomes evident that technological differences can be abstracted away via transparent mappings of identifiers and addresses into technology-specific formats and accessibility adapters encapsulating specific protocols and serialization formats. This provides compatibility both across technological borders and with upcoming standards and protocols.

6 Platform administration services

This category of services shall be used for the management of the platform and of I4.0 Component s (I4.0 component). It contains services that are essential for middleware solutions. The services shall be used to gather information on the components, their status, their identification information, their available models, etc.

These services allow maintaining the platforms on which business- and production-oriented services are executed. Technically, platform and production-related services are all refinements and compositions of the same information services as already indicated in Figure 5.

Table 6 summarizes the core topics of platform administration that need to be detailed further.

Table 6. Functionality required for the self-administration of the I4.0 Service System

Service topics	Short description	Proposed responsible
Registration and (semantic) discovery of service instances	Allow service participants to enter/leave a possible collaboration context and to find each other within that context (see Section 3.1). This allows loose and flexible coupling of the service participants which for an I4.0 System at any point in time. A key feature are semantic queries, where semantic identifiers (see Section 5.3) can be resolved into communication addresses (see Section 5.4) of data elements (see Section 5.2).	GMA FA 7.21
Access to common meta data (identification, versioning, etc.)	access to the asset meta data attributed to the administration shell header like identity	Plattform I4.0 WG 1 SWG Ontology Plattform I4.0 WG3 Security ZVEI SG Models and Standards
Authentication	Proof of identity of service providers/consumers; an asset directly implementing such services can thus prove its identity.	
I4.0 Networks	creation of I4.0 Networks and end-to-end connections, including the negotiation and observation of their QoS properties	Plattform I4.0 WG1, SWG Network Communication

7 Communication services

The core function of the communication layer is to provide end-to-end data exchange paths between service providers and consumers across existing I4.0 networks with a defined service quality. It is the main purpose of the communication services to expose this functionality.

A key ingredient of end-to-end data exchange is a common addressing concept as discussed in Section 5.3. The communication services thereby focus on addressing the hosts on either end of the data path, addressing host-internal information objects is a matter of the information services.

Table 7 provides an overview of the main functionality of the communication layer for which the services need to be provided. Defining the actual set of communication services and their signatures is the task of Platform WG1, SWG Network-Based Communication [5]. As with the specific signature given for the information services in Section 5.1, handling the context of communication plays a large role. In particular, most requirements on service quality and security from the information layer directly translate to the functionality that the communication layer must provide.

From the perspective of the communication layer, an information layer service call is thus a communication layer method call with a signature as outlines in Table 2.

Table 7. Communication functionality exposed by communication services

Service Topics	Short description	Remarks
Host discovery	find I4.0-compliant hosts in the network	
Transmission	transmit, respond of data telegrams	with or without the context of a session
Connections	connect, disconnect to another host	e.g. to realize a session concept
Quality	negotiate/request service quality for data transmission	with or without the context of a session
Security	encryption for confidentiality and integrity (secure end-to-end communication contexts)	focus on a secure context, which may or may not be based on a session concept; e.g. OPC UA publish/subscribe provides security without relying on session

8 Technology Mapping

8.1 OPC UA

In this section possibilities of mapping of the proposed operations and service calls to OPC UA and RESTful web services are discussed. The advantages of OPC UA is the efficient parsing of the messages, the built-in security model and rich meta-modelling facilities. The protocol is internationally standardised as IEC 62541 and is accepted to be a part of reference architecture model for I4.0 (DIN SPEC 91345). The advantages of RESTful web services are their firewall friendliness, an ad-hoc possibility to write servers/ clients due to the existing APIs or libraries for most programming languages and their prevalence in the domain of web.

The utilized service operations in Table 2 are close-aligned to both communication protocols. For example, all the required operations are already implemented by OPC UA communication protocol. A possible implementation of the proposed service architecture using RESTful web services over HTTP covers all the needed operations except publish/subscribe and custom method calls. The operations have to be implemented by some additional protocol extensions like server-sent events or simple application-layer workarounds like polling. Method calls can also be easily mapped to some existing HTTP operation, e.g., POST.

Another dimension of a service call is the established context of a service call. On one hand OPC UA provides elaborate mechanisms for establishing a Session, i.e. an active shared context between a server and a client. On the anther hand, RESTful service are per definition stateless, therefore the whole context has to be stored on the client and transferred during every service invocation. A possible compromise between these two opposite concepts is on one hand the usage of OPC UA in a RESTful way, e.g., only by self-describing requests. On the other hand, even RESTful services may introduce or rely on some server state, e.g., for security that is described in the following paragraph.

OPC UA has standardised security mechanisms, i.e., client/server authentication and non-repudiation for message. In the domain of RESTful services non-repudiation mechanisms can be obtained from the use of HTTPS technology. Authorization is a more problematic issue and is achieved by using pre-shared security tokens in most of the current RESTful APIs. These keys are usually obtained from the server by performing an authorization handshake, e.g., via OAuth. The security token links the client to some authentication context and is therefore also a key for some additional context information on the server.

8.2 Mapping to RESTful web-services

Mapping of operations to request endpoints. A convenient naming is needed, e.g., by a suitable suffix, e.g., */read, */write, */browse, etc. An XML/JSON format for requests and replies can be easily derived from presented service signatures. The format of information payload, i.e., the representation of the object can be subject of variation and use, e.g., JSON, ASN1 or even base64 encoded binary OPC UA objects.

8.3 Mapping to OPC UA

Two ways of mapping the model to OPC UA are possible: the first mapping assumes the semantic equality of entity and OPC UA variable nodes as well as relation and OPC UA references. The second mapping maps operations directly to OPC UA method calls. For both implementation possibilities OPC UA node IDs have to mapped to URIs and vice versa. URIs are unique locally in the application address space. Otherwise, URIs begin with “opc.tcp://<server>/application/” to denote a remote application.

The first mapping has the advantages of reusing OPC UA services like read, write, browse, subscribe etc. These are usually implemented in OPC UA stack and therefore do not need to be implemented by the application. The disadvantages of the first mapping approach is a need of mapping between URIs and OPC UA node IDs as well as the creation and maintenance of nodes in OPC UA server. The last task can be solved by dynamic construction of nodes “on request”.

The second implementation ignores most of OPC UA meta-modelling facilities and uses only OPC UA as a transport to pass the data to the application via method calls. This approach does not require a mapping between URI and OPC UA Node IDs needed, since the entities are not represented by OPC UA nodes. The disadvantage of this approach is the need of re-implementation of built-in services like read and browse is needed by the server application layer.

9 Outlook

In the next steps, the given perspective on the I4.0 Service Architecture needs to be refined toward a formal specification. To this end, a collaboration (discussion and support) from the related I4.0 Committees is required. This means in particular the following topics and committees:

- formal specification of the information services and information meta-models within GMA FA 7.21, including concepts for data semantics. This must be on a level of proficiency to allow the specification of information models and actual implementation of information services in components (provided a technology mapping is available for the selected technology). Alignment with the general ASS concept including the representation of header and meta-data need to be aligned with both WG1, SWG Ontology and ZVEI SG Models and Standards.
- outline of the platform administration services within GMA FA 7.21. This includes directory services to discover the (distributed) contents of the ASS. Services for identification and authentication need to be defined by WG3 Security. Services for handling access to I4.0 networks need to be defined by WG1, SWG Network Communication.
- Outline of the individual communication services by WG1, SWG Network Communication, including a set of common-practice addressing formats. Definition of a minimum ASS for the communication layer.
- formal technology mapping of the different service classes to selected technologies. OPC UA is proposed including its recent AMQP extension.

On this basis, we can initiate the migration of the wide range of Industrie 3.0 information standards. This is essential to reduce the “time to market” of I4.0, so to speak, because the combined existing know-how in these standards can be re-used easily. As suggested in [3] these standards can either be transformed into I4.0-conforming information models or they will be accessible as complementary formats [3]. The latter further facilitates migration because existing standards can be use as they are, the former allows for a deeper level of semantic interoperability.

To enable interoperability and sustainability, a technology-independent approach is a solid foundation for the service architecture. However, actually implementing any kind of solution cannot be done without choosing a particular technology. While OPC UA is an often-discussed candidate for the core technology of I4.0, a formal technology selection in the Plattform I4.0 has not been made and is needed at some point.

In conclusion, the presented model and service concepts are the foundation for more detailed specifications of the I4.0 Service Architecture. Therefore, they need to be brought to the level of a formal specification in the next step to be validated from an application (functional) perspective.

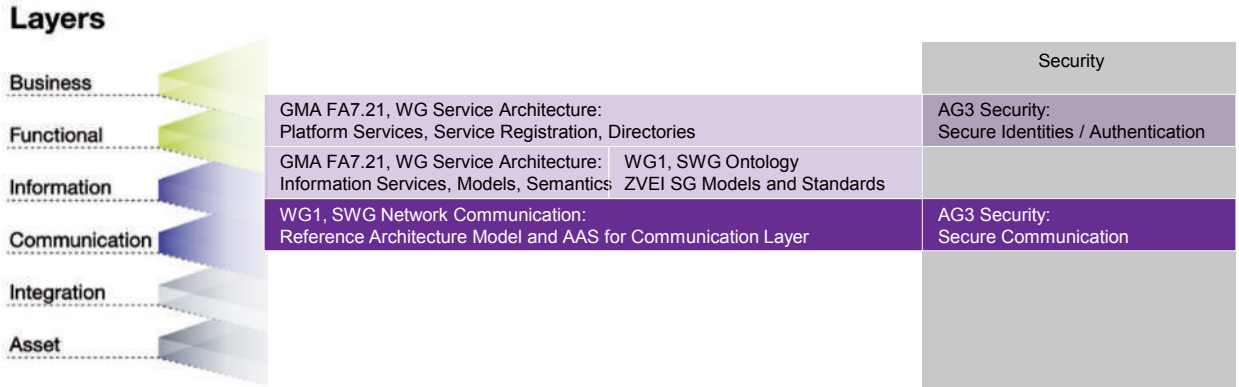

Figure 10. Illustration of proposed collaboration topics between GMA FA 7.21 and other committees

10 Glossary and Definition of Interoperability

The terminology of this status report adheres to the definition given in existing publications [7] [3].

With interoperability being the key achievement expected from a common service architecture, it is important to give a clear definition of this term.

Interoperability thereby is a specific degree of compatibility, meaning the capacity of different devices to collaborate in a (distributed) system as indicated in Figure 11.

The following main features are used for the definition of compatibility levels (Table 8).

A system is a composition of several functionality.

Interoperability has to be defined and proved by singular functions related to their variables/parameters. For example a field device can be interoperable or even exchangeable related to its main measurement value function chain but not related to its parameter set.

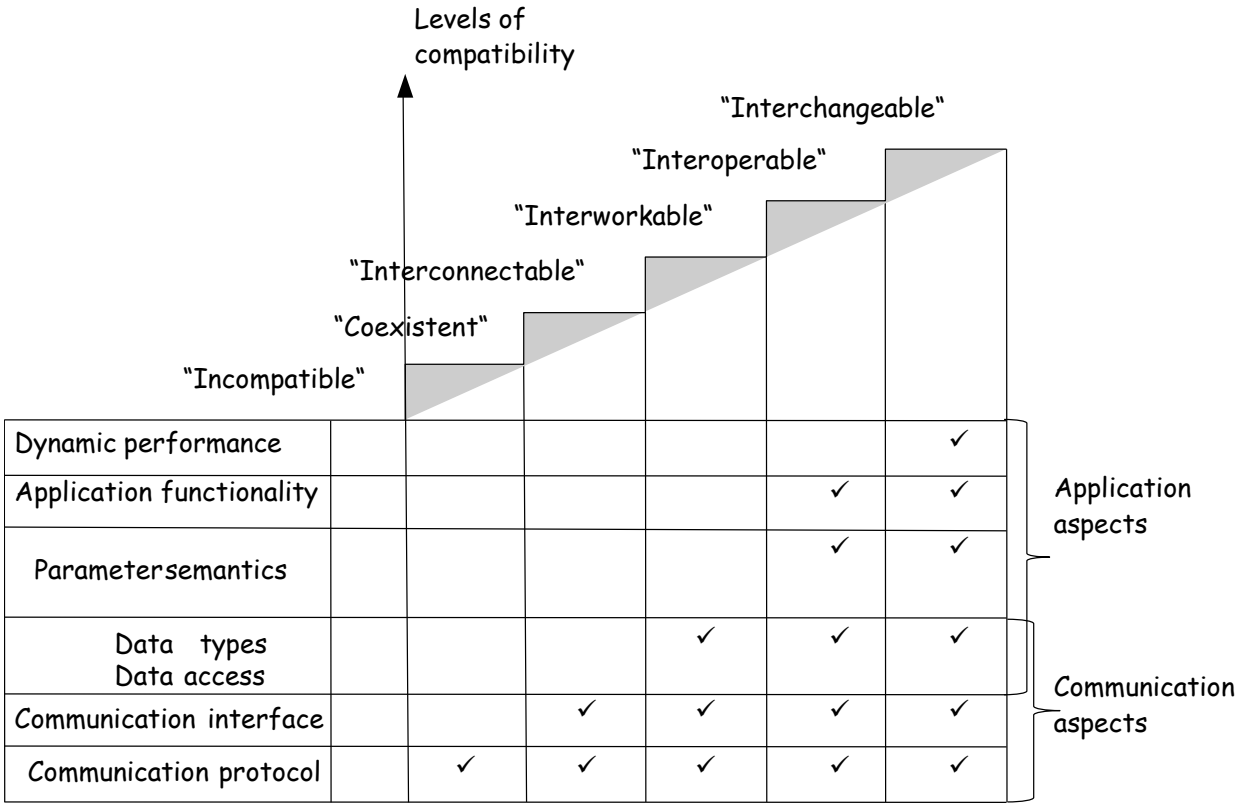

Figure 11. Levels of functional device compatibility (DIN EN 61804-2)

Table 8. Features influencing the compatibility levels of system functionality (DIN EN 61804-2)

Feature	Description
Communication aspects	
Communication protocol	This feature is defined by all protocols of layer 1 to 7 of the ISO/OSI reference model, i.e. from the physical medium access to the application layer protocol.
Communication interface	This feature is defined by the communication service definition of the application layer including the services and the service parameters. Additional mapping mechanisms can be necessary. The dynamic performance of the communication system is part of this feature.
Data types	This feature is defined by the data type of the block data input, data output or parameter.
Parameter semantics	This feature is defined by the characteristic features of the data (for example, this can be data name, data descriptions, the data range, the substitute value of the data, the default value, the persistence of the data after power loss and deployment).
Application functionality	This feature is defined by specifying the dependencies and consistency rules between the variables inside the blocks. This is done in the data description part or in a separate behavior section.
Dynamic performance	This feature is defined by time constraints which influence the data or the general device behaviour. For example, the update rate of a process value can influence block algorithms.

Bibliography

Technical rules

DIN EN 61804-2:2007-10 Function blocks (FB) for process control; Part 2: Specification of FB concept. Berlin: Beuth-Verlag

DIN EN 81346-1:2010-05 Industrial systems, installations and equipment and industrial products; Structuring principles and reference designations; Part 1: Basic rules. Berlin: Beuth-Verlag

DIN SPEC 91345:2016-04. Reference Architecture Model Industrie 4.0 (RAMI4.0). Berlin: Beuth-Verlag

IEC 62832 Industrial-process measurement, control and automation; Reference model for representation of production facilities (Digital Factory). Berlin: Beuth-Verlag

ISA-TR88.00.02:2015 Machine and Unit States (PackML)

Literature

[1] VDI/VDE Society Measurement and Automatic Control (GMA) and ZVEI: Status Report; Reference Architecture Model Industrie 4.0 (RAMI 4.0), Düsseldorf, July 2015

[2] ZVEI: Industrie 4.0: The Industrie 4.0 Component , Frankfurt, April 2015, http://www.zvei.org/Downloads/Automation/ZVEI-Industrie-40-Component-English.pdf

[3] VDI/VDE Society Measurement and Automatic Control (GMA) and ZVEI: Status Report; Struktur der Verwaltungsschale - Fortentwicklung des Referenzmodells für die Industrie 4.0-Komponente (structure of the asset administration shell – continued development of the reference model for Industrie 4.0 Component s), Düsseldorf, April 2016

[4] Bitkom, VDMA, ZVEI: Implementation Strategy Industrie 4.0 - Report on the results of the Industrie 4.0 Platform, Jan 2016, http://www.zvei.org/Publikationen/Implementation-Strategy-Industrie-40-ENG.pdf

[5] Plattform Industrie 4.0, AG1, Sub-working-group (UAG) Network-Based Communication: Netzkommunikation für Industrie 4.0 (network-based communication for Industrie 4.0), Hannover Fair, April 2016

[6] Plattform Industrie 4.0, AG3 (Security): Sichere unternehmensübergreifende Kommunikation (secure inter-company communication), Hannover Fair 2016

[7] Glossar Industrie 4.0 des Fachausschuss VDI/VDE-GMA 7.21 „Industrie 4.0“, http://www.iosb.fraunhofer.de/servlet/is/51204/

[8] Field Device Integration (FDI), IEC 62769, Ed 1.0, 2015

[9] FieldComm group organization, http://www.fieldcommgroup.org/

[10] User Association of Automation Technology in Process Industries (NAMUR), homepage, http://www.namur.net/

[11] N. Koshizuka and K. Sakamura, “Ubiquitous id: standards for ubiquitous computing and the internet of things,” IEEE Pervasive Computing, no. 4, pp. 98–101, 2010.

[12] DKE: Die Deutsche Normungs-Roadmap Industrie 4.0. Version 1.0 (Stand 11.12.2013). German Commission for Electrical, Electronic and Information Technologies of DIN and VDE, 2013, http://www.dke.de/de/std/informationssicherheit/documents/nr%20industrie%204.0.pdf

[13] OASIS. “Reference Model for Service Oriented Architecture 1.0”. Committee Specification 1, 2006. http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[14] VDI/VDE Society Measurement and Automatic Control (GMA); Statusreport; Industrie 4.0 - Auf dem Weg zu einem Referenzmodell. Düsseldorf, April 2014; http://www.vdi.de/industrie40

[15] Drive Engineering / Servo, DriveCom User Group e.V., 1994, http://www.drivecom.org/download/driv_22_e.pdf

About VDI

Advocates, promoters and networkers

Engineers need a strong association to support, promote and represent them in their work. The Association of German Engineers, VDI, performs that function. For over 160 years, it has been a reliable partner to engineers in Germany. Over 12,000 experts work on a voluntary basis each year to promote our location for technology. With convincing results: With around 155,000 members, VDI is the largest engineering association in Germany.

VDI Verein Deutscher Ingenieure e.V.
VDI/VDE Society Measurement and Automatic Control (GMA)
Dr. Heinz Bedenbender
Phone. +49 211 6214-485
bedenbender@@vdi.de
www.vdi.de