

WORKING PAPER



Structure of the Administration Shell

Continuation of the Development of the Reference Model for the Industrie 4.0 Component

In Kooperation with



This publication is a result of the WG reference architectures, standards and norms in cooperation with the ZVEI – Zentralverband Elektrotechnik- und Elektronikindustrie e.V.
(German Electrical and Electronics industry e.V.)

Imprint

Published by

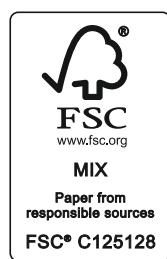
Federal Ministry for Economic Affairs and Energy (BMWi)
Public Relations
10119 Berlin
www.bmwi.de



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

Text and editing

Plattform Industrie 4.0
Bertolt-Brecht-Platz 3
10117 Berlin



Design and production

PRpetuum GmbH, Munich

Status

April 2016

Print

Spree Druck Berlin GmbH

Illustrations

Festo AG & Co. KG (Title)

This brochure is published as part of the public relations work of the Federal Ministry for Economic Affairs and Energy. It is distributed free of charge and is not intended for sale. The distribution of this brochure at campaign events or at information stands run by political parties is prohibited, and political party-related information or advertising shall not be inserted in, printed on, or affixed to this publication.



This publication as well as further publications can be obtained from:

Federal Ministry for Economic Affairs and Energy (BMWi)
Public Relations
E-mail: publikationen@bundesregierung.de
www.bmwi.de

Central procurement service:

Tel.: +49 30 182722721
Fax: +49 30 18102722721



Content

Authors	4
1 Preamble	5
1.1 Editorial notes	6
1.2 Objective and methodology of this document	6
2. Relevant Content from various Sources	7
2.1 Reference architecture model Industrie 4.0 (RAMI 4.0)	8
2.2 Industrie 4.0 Component in the Implementation Strategy of April 2015	8
2.3 Evaluation of detailed application scenarios	10
2.3.1 Application scenarios from ZVEI SG “Strategy & Use Cases”	10
2.3.2 Use Case “Self Optimization”	10
2.3.3 Consequences for the structure of the Administration Shell	12
2.4 Digital Factory	13
2.5 Composability	14
2.6 Semantic networks	14
2.7 IEC 61360 properties	15
2.7.1 Properties	15
2.7.2 Containers for properties	16
2.7.3 Representation in semantic technologies	17
2.8 Examination of various groups of properties	18
2.8.1 Many matterS characterise properties even now	18
2.8.2 Addressing different groups of properties	19
2.8.3 Mapping between property sets of different domains	20

3. Structure of the Administration Shell	22
3.1 Views	23
3.2 Requirements in regard to the Administration Shell	24
3.3 Classes of property	26
3.4 Requirements in regard to individual Elements of information	26
3.5 General structure of the Administration Shell	28
3.6 Asset structure	29
3.7 Compatibility with the Digital Factory	30
3.8 Identifiers	31
3.8.1 Initial situation	31
3.8.2 Determination of identifiers	31
3.8.3 Secure added-value networks	32
3.8.4 Association of further identifiers	32
3.8.5 Best Practices for Identifiers for Assets and Administration Shells	32
4. Methodology for the distributed formulation of I4.0 submodels	33
4.1 Formulation of a I4.0 submodel on the basis of an existing standard	34
4.2 Agile approach to the identification of new content	37
Literature/Appendices	38
Literature	39
Appendice A. Best practices for identifiers for assets and Administration Shells	39
1. Introduction	39
2. Who needs identification?	39
3. Principles	40
4. Information for identification	40
5. Technical Realisation	42
Appendice B. Lightweight Approach to the Standardisation of Vocabularies for Semantic Interoperability	44
1. Introduction	44
2. Requirements of collaborative vocabulary development	44
3. VoCol – Lightweight development of vocabularies	46

Authors

- Dr.-Ing. Peter Adolphs, Pepperl + Fuchs GmbH
- Prof. Dr. Sören Auer, Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme IAIS
- Dr. Heinz Bedenbender, VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA)
- Meik Billmann, ZVEI – Zentralverband Elektrotechnik- und Elektronikindustrie e.V.
- Martin Hankel, Bosch Rexroth AG
- Roland Heidel, Roland Heidel Kommunikationslösungen e.K.
- Dr.-Ing. Michael Hoffmeister, Festo AG & Co. KG
- Haimo Huhle, ZVEI – Zentralverband Elektrotechnik- und Elektronikindustrie e.V.
- Michael Jochem, Bosch Rexroth AG
- Markus Kiele-Dunsche, Lenze SE
- Gunther Koschnick, ZVEI – Zentralverband Elektrotechnik- und Elektronikindustrie e.V.
- Dr. Heiko Kozolek, ABB AG
- Lukas Linke, ZVEI – Zentralverband Elektrotechnik- und Elektronikindustrie e.V.
- Reinhold Pichler, DKE – Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE, Frankfurt am Main
- Frank Schewe, PHOENIX CONTACT Electronics GmbH
- Karsten Schneider, Siemens AG
- Bernd Waser, Murrelektronik GmbH

1. Preamble

1.1 Editorial Notes

This document was created in June 2015 and was submitted to SG "Models and Standards". It is based on content being discussed in the form of presentations by SG "Models and Standards" in May/June 2015.

The document is organized such that by contents and structure it is a logical continuation of the published paper in respect of the Industrie 4.0 Component in the Implementation Strategy of April 2015. It should be possible to incorporate and co-ordinate the proposed content in the committees of the new Industrie 4.0 platform and to publish it appropriately around late 2015.

Management/integration of content of an "IT-compliant" specification should be possible.

The subsequent discussion was based on this document. For inclusion of commentaries and for documentation the revision tracking function of the text system was frequently used. Texts, underlying graphics and also the structure were repeatedly revised; corresponding versions were saved.

For better readability, in compound terms the abbreviation "I4.0" is consistently used for "Industrie 4.0". Used on its own "Industrie 4.0" continues to be used.

1.2 Objective and methodology of this document

This document bundles the technical discussions of the SG "Models and Standards" of the ZVEI in respect to the structure of the implemented Administration Shell. The objective is to establish a consensus between the participating committees as to which properties, data and functions should generally be contained in an Administration Shell and how these can be represented. The observations should allow other participants, for example GMA FA 7.21 of VDI/VDE, to make proposals regarding IT structures and IT services. The observations should make it possible for all participants to make proposals in respect to information contained in the Administration Shell – so called "submodels". The objective is not to create a conclusive IT specification or a definitive specification for the implementation of an individual device or system. By contrast a direction should be reliably set in which the content-related discussion and standardisation in respect of the I4.0 Component will move during the next few months.

The observations contained in this document apply in equal measure to the industries of both factory automation and process automation. Terms such as "factory", "production" and "shop floor" thus also refer to the facilities of the processing industry.

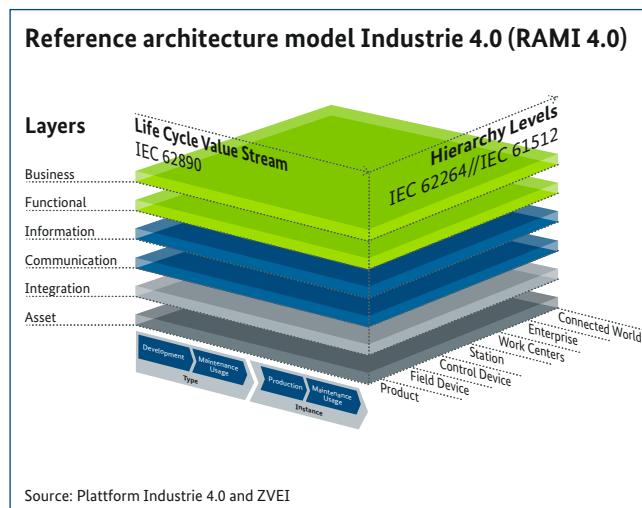


2. Relevant Content from various Sources

This section, similar to the document of Version 1 of the I4.0 Component, points out important matter from previous discussions or from other working groups. It should thus portray the cross-linking to other topics.

2.1 Reference architecture model industrie 4.0 (RAMI 4.0)

The reference architecture model Industrie 4.0 (RAMI 4.0) joins the most important elements of Industrie 4.0 in a three-dimensional layer model. On the basis of this framework Industrie 4.0 technology can be systematically classified and refined¹. It consists of a three-dimensional co-ordinate system that contains the most important aspects of Industrie 4.0. Complex inter-relationships thus can be broken down into smaller manageable packets.



Among other things it is worthy of note that according to this architecture the definitions and data in respect of one asset can and must be located and maintained at various positions in the Information Layer. For instance the manufacturer of a component will locate and maintain "his" data on the "Life Cycle & Value Stream" axis in the "Type" segment. Independently of this there are then data about the individual instances of the produced components, which are similarly used, maintained or even extended by the respective user (e.g. to include information about mainte-

nance and modification). Thus two requirements emerge from RAMI 4.0:

Requirement: It must be possible to use, maintain or even extend the definitions and data in respect of an asset throughout its lifetime if the Use Case so requires.

Requirement: It should be possible to preserve a link between "type" and "instance" definitions in respect of an asset throughout its lifetime².

2.2 Industrie 4.0 component in the implementation strategy of april 2015

The I4.0 Component was introduced in an initial version of the document "Implementation Strategy Industrie 4.0" from April 2015 (provided by the I40 platform). Key points of this specification were as follows:

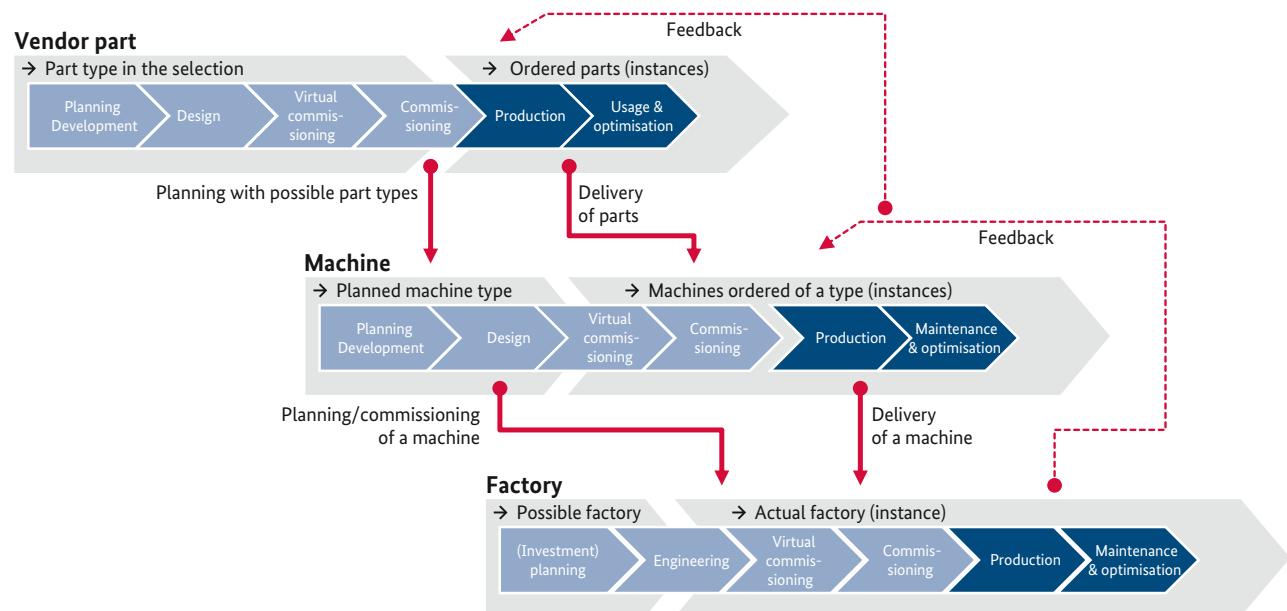
- building upon the definitions of GMA FA 7.21
- the suitability of the I4.0 Component for a wide range of life cycles in relation to the various partners of a value-added network (see image below)
- the possibility of locating the I4.0 Component in RAMI 4.0 (e.g. on the development side, on the production/usage side, at a wide range of hierarchy levels)
- the possibility of operating I4.0-compliant communication equally for both active and passive connected assets

Definition of the Administration Shell with Virtual Representation and Technical Functionality was also a key element of this introduction. The Administration Shell can refer to one or multiple assets. The "Manifest" was mentioned as an important part of the Virtual Representation, and this Manifest can be viewed as a directory of the individual data content of the Virtual Representation. Thus it also contains so-called meta-information. Besides the Manifest contains mandatory data in respect of the I4.0 Component, among other things the links with the assets by means of identification and security capabilities. The security capabilities of an assets must be in conformity with the

1 <http://www.zvei.org/Downloads/Automation/ZVEI-Industrie-40-RAMI-40-English.pdf>

2 The types and instances referred to here explicitly do not relate to the type-instance concept of object-oriented programming, but instead describe product types and product instances in the sense of automation technology. They are based on the product life cycle introduced in RAMI 4.0.

Typical cross-linking of the life cycles of supplied part, machine and factory – as type and instance respectively

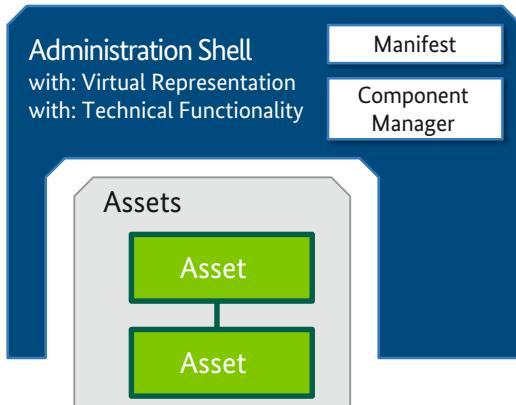


Source: ZVEI SG Modelle und Standards

required security capabilities of the Administration Shell. The Component Manager³ represents the link to the ICT technical services of the I4.0 Component, which allow external access to the Virtual Representation and Technical Functionality. The Component Manager can thus, for example, tie in a Service-oriented Architecture (SOA) or deploy the Administration Shell into a repository.

I4.0 Component as a combination of one or more assets with an Administration Shell

I4.0 Component



Source: ZVEI SG Modelle und Standards

3 In the previous documents the Component Manager is described as a Resource Manager; however, in future it shall be referred to as the Component Manager.

2.3 Evaluation of detailed application scenarios

The structure of the Administration Shell should be able to support the corresponding Use Cases of Industrie 4.0 in a suitable manner. Required data, functions and potential Security Requirements must be identified, and unnecessary additional effort in terms of definitions should be avoided. Some application scenarios have been defined by ZVEI for example; further Use Cases have for example already been defined by means of the “Recommendations for implementing the strategic initiative Industrie 4.0” (Acatech)⁴.

2.3.1 Application scenarios from ZVEI SG “Strategy & Use Cases”

The above image shows the application scenarios selected by Management Group Industrie 4.0 (SG Strategy and Use Cases) of ZVEI. The description and cross-linking of further application scenarios are co-ordinated by the Industrie 4.0 platform and its AG2. The application scenarios of ZVEI give a detailed description of typical specific applications of Industrie 4.0 in various industries (manufacturing industry, process industry and hybrid production). For each of the

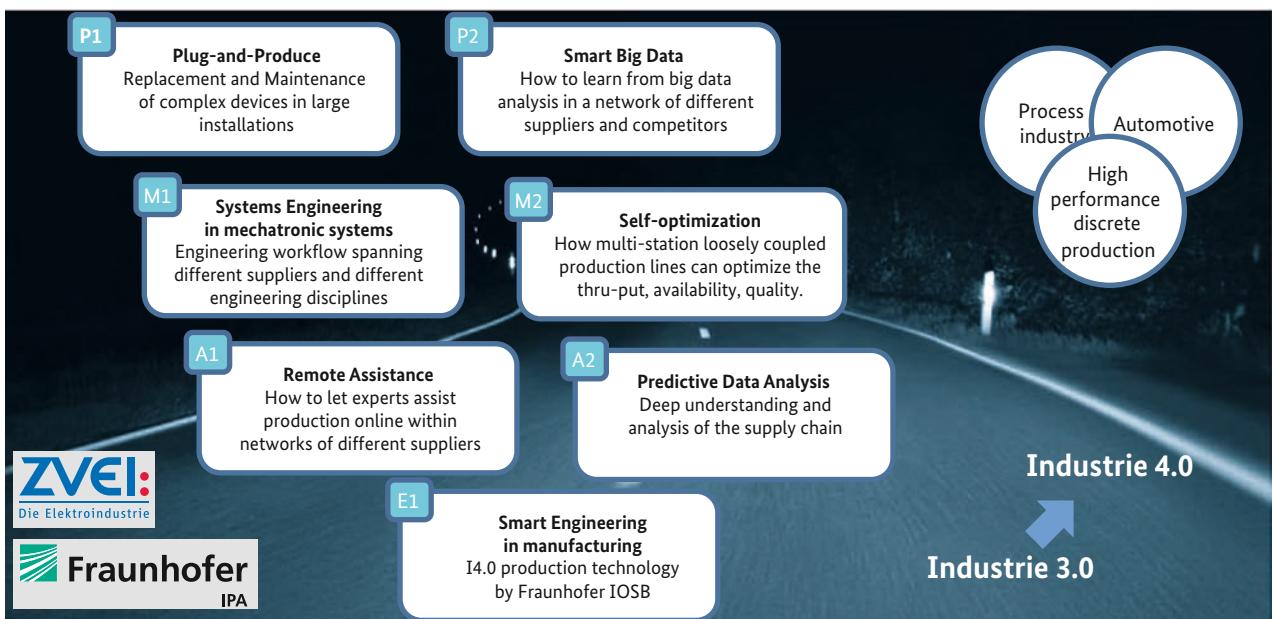
application scenarios several Use Cases are currently (August 2015) identified and described in standardised form.

2.3.2 Use case “self-optimization”

The use case “self-optimization” (also known as “Buffer-free-filling by M2M communication”, cf. Plattform I40 AG1/AG2) tackles the problem of misaligned production stations in bottling plants in order to improve production flexibility and efficiency. The production in existing bottling plants is often directed by a central Manufacturing Execution System that optimizes the production schedule across stations. Involved stations are for example stretch blow molding stations, labelling station, filling station, Cleaning-in-Place station, packing station, and palletizing stations (cf. Fehler! Verweisquelle konnte nicht gefunden werden.). Goods are transported between stations using conveyor belts and/or autonomous guided vehicles (AGV).

While in theory the central planning can lead to a globally optimal production schedule, the approach has shown inflexibilities upon certain failure conditions and small

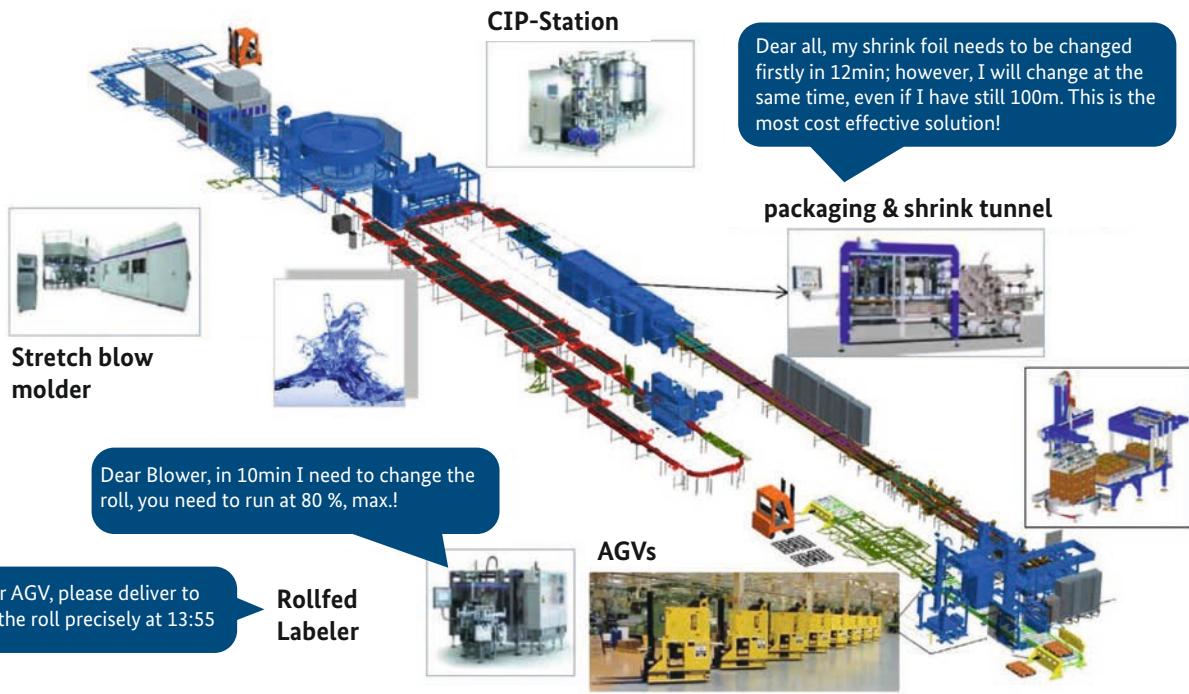
Application scenarios of Management Group Industrie 4.0 of ZVEI



Source: Festo AG & Co. KG

4 http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report_Industrie_4.0_accessible.pdf

Use Case Buffer-free-filling



Quelle: KHS GmbH und ABB

batches that require reconfiguration. For example, if a bottle is stuck in a product station essentially stopping the production line, the other production station currently do not react automatically, by for example lowering their product output to avoid larger buffers between stations or by executing scheduled maintenance tasks ahead of schedule during the stoppage. Today, the station operator needs to fix the issue and manually re-configure the production stations.

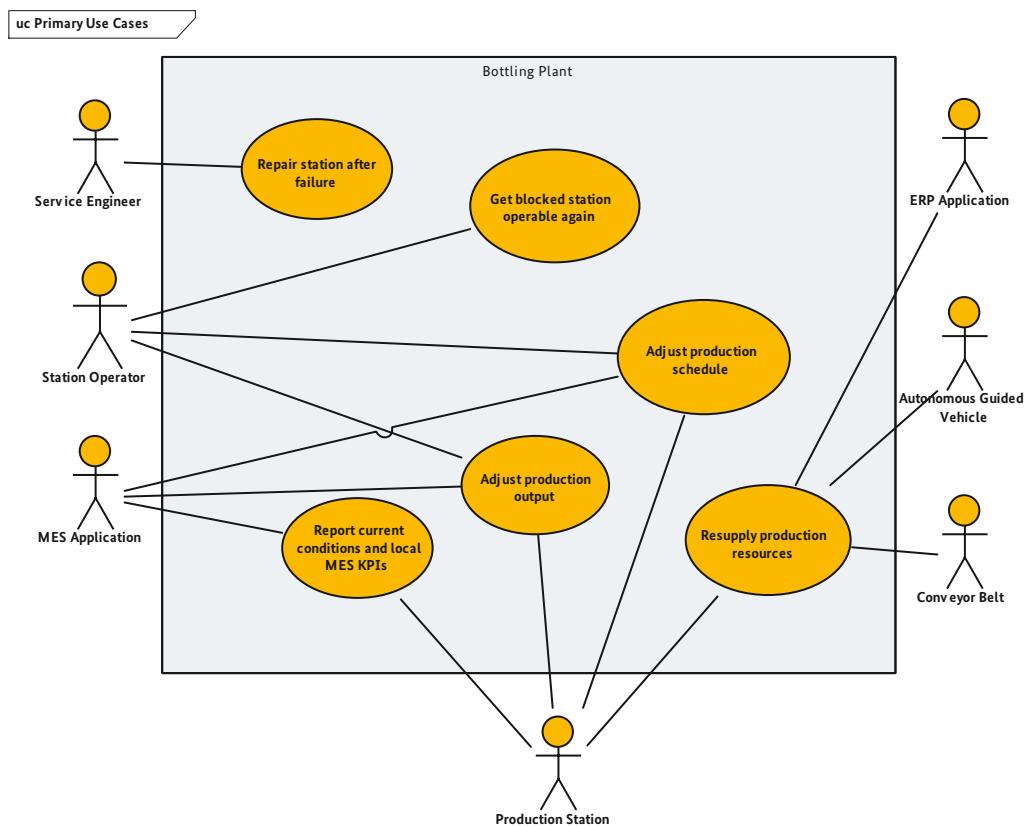
The buffer-free-filling shall be achieved by moving more intelligence from the MES to the production stations and by increasing the communication among production stations. Each station monitors its own status and provides certain MES KPIs (e.g., utilization rate) to the plant network. Other production stations can subscribe to this status information and draw local conclusions for optimizing their own functionality. For example, if an alarm is raised by a subsequent station in the production line, the current stage can decelerate its production rate in order to avoid large output buffers. The production stations shall be more aware of their own conditions and learn from past behavior. For example, a production station can predict when its produc-

tion resources (e.g., labelling paper, bottle raw material) is running out and the autonomously re-order supplies from other stations or transporters. It can prepone scheduled maintenance tasks that require stopping the machines if this suddenly fits into the production schedule.

The use case needs to be completed by assistance systems for service support personnel. It is conceivable to add additional security and safety functions based on the information exchanged between production stations. Supplies can be automatically ordered via an ERP system connected to the Internet.

If production stations of different vendors shall interact seamlessly, the use of standardized and secure communication protocols, information models, and functional specifications is essential for the implementation of this use case. Candidate standards are OPC UA, ISA-95/88, OMAC PackML, Weihenstephan Standards, and MES KPIs according to ISO 22400. It is however unknown whether these standards are sufficient to implement the use case in a vendor-neutral way.

Use Case Diagram for the Use Case “Self Optimization”



To support the implementation of this use case, the administration shell of Industrie 4.0 Components should be able to carry the (security-) information and provide the services imposed by the previously mentioned standards.

2.3.3 Consequences for the structure of the Administration Shell

In the overall recognition of the Use Cases of ZVEI from the above sections the following can be stipulated for the structure of the Administration Shell:

Demands for the structure of the Administration Shell

Industries	Manufacturing industry, process industry, hybrid production
Value chains	Logistics, procurement, production, outgoing goods, service
Value-added networks	Several partners, overarching
Security	<p>Yes, according to the principle "CIA" (Confidentiality, Integrity, Availability) through guaranteeing the confidentiality, integrity and availability of saved and transferred information</p> <p>For utilisation of data beyond company boundaries (e.g. through Cloud): pseudonymisation/anonymisation for personal data necessary, cross-company identity and rights management</p> <p>Yes, including guaranteeing of the confidentiality and integrity of the data and functions as well as maintenance of the availability of the Technical Functionality and of the functions of the underlying assets.</p>
Safety	Yes
RAMI 4.0: Layer	All
RAMI 4.0: IEC 62890	Requirements, concept, design, commissioning, operation, upgrading
RAMI 4.0: Hierarchy Level	Spans all defined hierarchy levels of the RAMI Model

2.4 Digital Factory

The standard for the Digital Factory, IEC 62832 CD2 Part 1, defines a framework of abstract definitions for:

- Assets of automated systems
- Structural and behavioural relationships
- Feature (property) management
- Hierarchical relationships
- Technical aspects

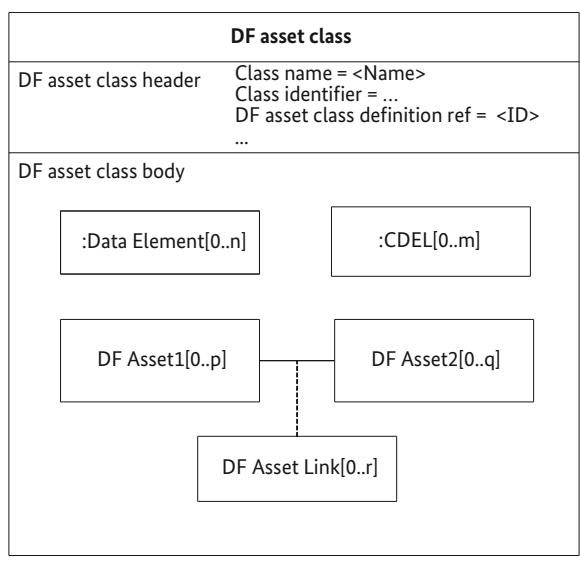
Thus this proposed standard has a similar domain to that of the RAMI 4.0 Model and the I4.0 Component. **IEC 62832 thus serves as a model for the structuring of the Administration Shell.**

Requirement: The definitions of the I4.0 Component should not conflict with the definitions of IEC 62832.

An asset is understood as a physical or logical object which is owned or managed by an organisation and which has an actual or perceived value for the organisation. The standard addresses in particular the utilisation phase of such assets and the related production facilities, their design, construction, commissioning, operation and maintenance. For the identification of concepts the standard uses identifiers according to ISO 29002-5. The assets under consideration can also be identified by means of other identifiers (e.g. URI).

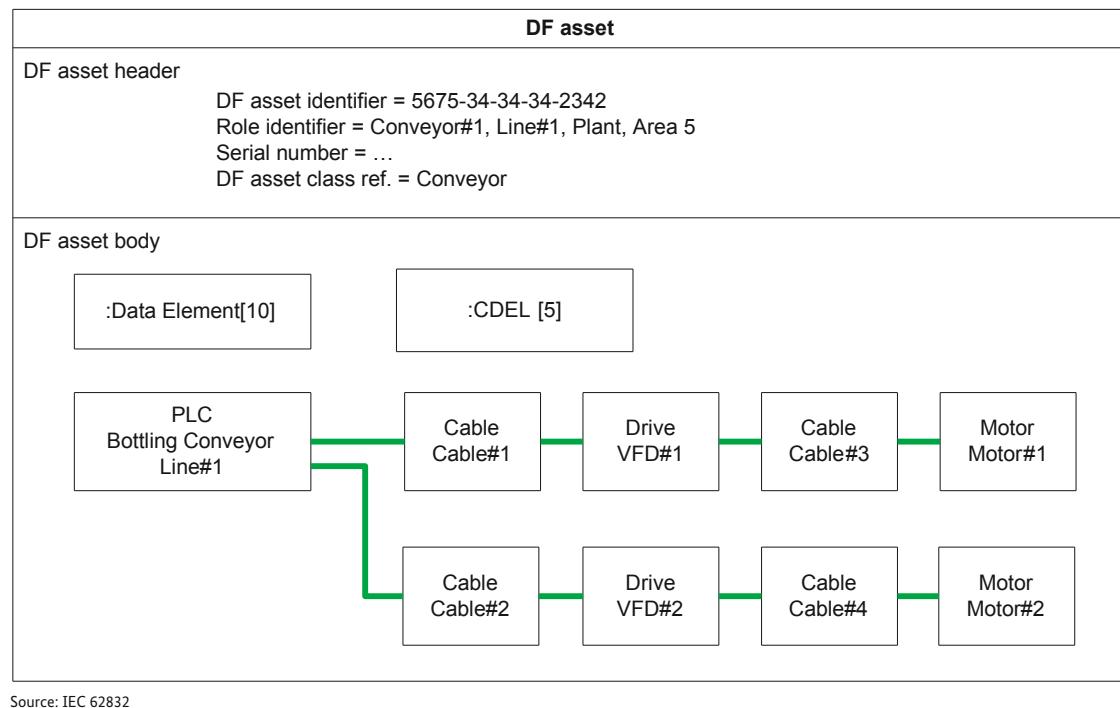
Assets ("PS asset", real or logical items) are described by means of asset descriptions ("DF asset", Virtual Representation). Classes of assets are modelled by means of so-called asset classes and thus stand respectively for one or more assets which share the same set of features. The features of the assets are described by means of data elements. If the described assets have a modular structure, the corresponding asset descriptions (asset classes) can similarly describe a modular structure. In this case the asset class describes the modules as assets and represents the links between the assets.

Structured asset class in the Digital Factory (DF)



Source: IEC 62832

Description of a structured DF asset which is composed from several assets



Most important is the breakdown of the description into “header” and “body”. The “header” contains information for identification and designation of the specific asset in the respective factory and stipulates aspects for administration of the asset. The “body” contains data for description of the features of the asset class with their respective disposition (i.e. data element values) for the concrete asset. For the specification of data elements the Digital Factory makes reference to the IEC 61360 properties (see on top).

2.5 Composability

Version 1 of the I4.0 Component stipulates logical composability in terms of possible modularisation. The Digital Factory follows a similar concept. In this manner machines, workstations, lines, production systems and whole factories

and processing plants can be understood as being hierarchical structures of I4.0 Components or “DF assets”. Please note: the composability of the I4.0 Component is a feature which depends on the Use Case (→ 2.3).

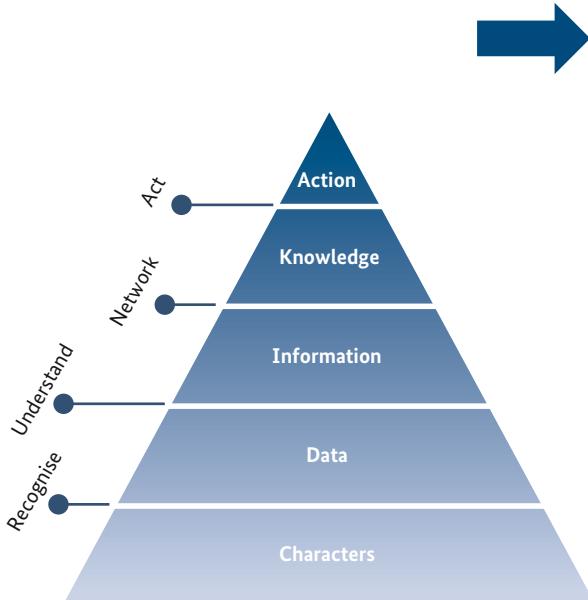
2.6 Semantic Networks

In accordance with the Implementation Strategy⁵ the I4.0 Component must also meet the challenges of semantic cross-linking of information⁶. The so-called knowledge pyramid organises corresponding terms. Ultimately it is important to be able to match information to knowledge in an over-arching manner. The image illustrates this by means of some selected examples:

5 Realisation Strategy Industrie 4.0” of the platform of April 2014, page 37, 51 or other

6 <http://www.bmwi.de/DE/Service/Veranstaltungen/dokumentationen,did=677026.html>

The knowledge pyramid places data, information and knowledge in relation to each other



The knowledge pyramid according to Fuchs-Kittowski.

Source: ZVEI SG Modelle und Standards

Our reference model should allow models...

- ▶ to make action possible, for example by means of **self-X capabilities**
- ▶ to **network** data and information about knowledge, e.g. by means of semantics
- ▶ to **provide** information, for example „temperature at process point X“
- ▶ to **provide** saved data, such as manuals
- ▶ to **provide** current data, such as runtime data

2.7 IEC 61360 Properties

In the context of the Digital Factory, and in the context of many standardisations in factory and process automation we find properties and lists of properties (LOP). Properties and their electronic data directories serve the following purposes⁷

- unique identification of hierarchies, classes and features and of their relationships,
- introduce a common distributed terminology and
- merge different attributes (e.g. SI unit, definition and data type) into one technical representation of a feature

and thus can be viewed as basic components and words of a common language between the various different entities of Industrie 4.0.

2.7.1 Properties

The IEC 61360 standard specifies various fundamental concepts, e.g. dictionaries, item classes, data element types and lists of values. These can be understood as fundamental information units “whose identification, description and value representation are established”⁸.

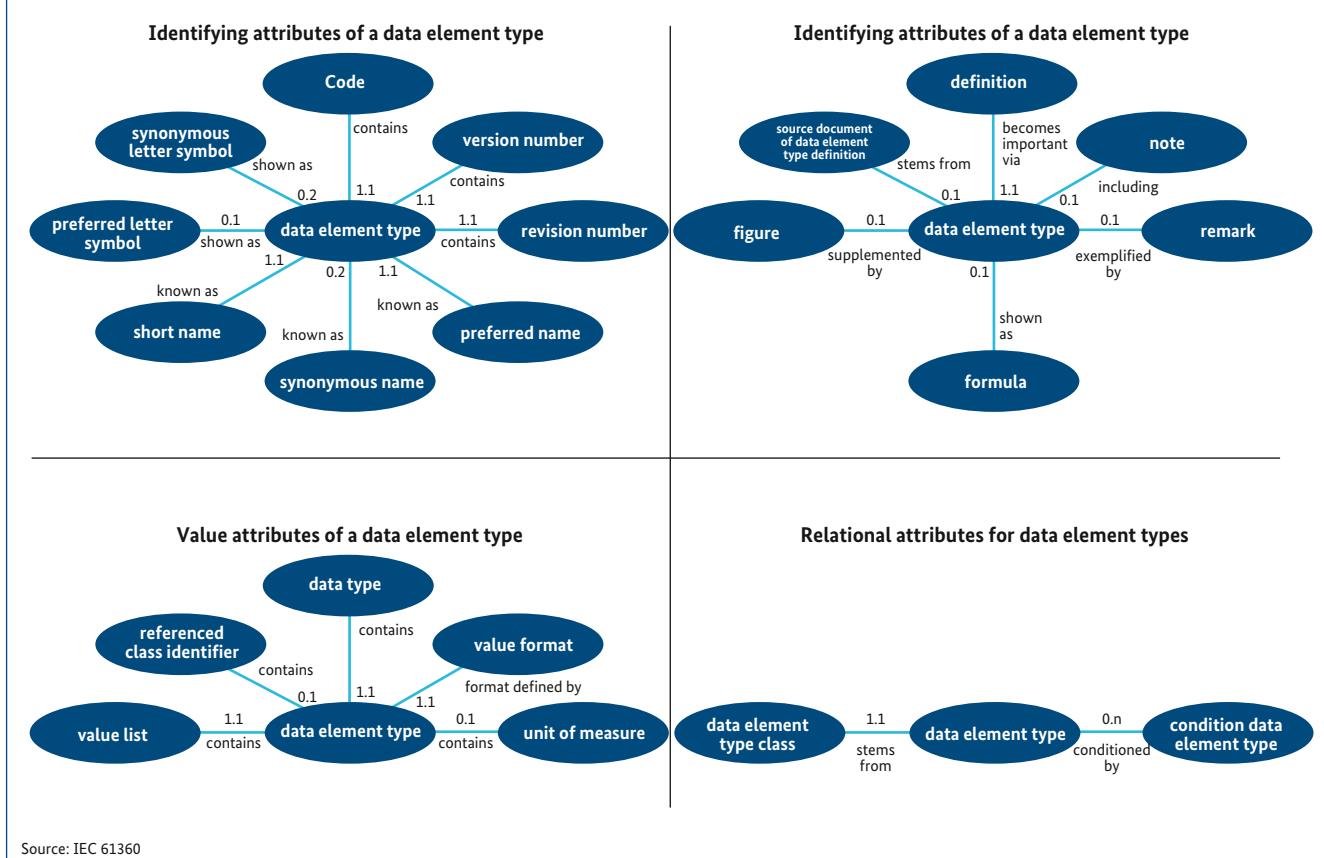
Data element types are also referred to as properties. The information model divides the attributes of a data element type into four primary groups:

- identifying attributes,
- semantic attributes,
- value attributes and
- relational attributes of data element types for the relationships between the entities.

7 <http://std.iec.ch/iec61360>

8 DIN EN 61360-1

Excerpts from the definitions of IEC 61360



The composition of the individual main groups is illustrated by the graphic explanations from the standard. As things stand at present there is no definition as to which attributes must necessarily be utilized by the I4.0 Component.

Based on this property definition dictionaries (domain ontologies) for the description of production resources can be created. In doing so, classes of production resources are defined which are associated with a well-defined set of properties. There are various organisations which maintain such ontologies, e.g. eCl@ss e. V.⁹ (as an example of an industry consortium) and IEC¹⁰ (as an example of a standard organisation). The various organisations can use different approaches for structuring the ontologies (e.g. as informal or formal hierarchies). On the basis of the concept identifiers (accordingly to IEC 29002-5) it is possible to work out which organisation has defined a class or a property.

Based on the dictionaries manufacturers of production resources can make libraries available with descriptions of products offered (i.e. electronic catalogues with DF asset classes). In principle it is also possible to use properties defined by different organisations.

2.7.2 Containers for Properties

The above standard specifies individual data element types. For joint model design and lifelong maintenance of the properties in respect of the data and definitions of "types" and "instances" of assets there is a requirement for ICT-related realisation of these models in so-called "Containers" which aggregate submodels of property lists. Examples of existing implementations are BMEcat¹¹, OpenTRANS¹² and the XML schemes of the eCl@ss Association. These constitute

9 <http://www.ecl@ss.eu/> Basic Ontology: <http://www.ecl@sscontent.com/>

10 <http://www.iec.org/> Ontology: <http://std.iec.ch/iec61360>

11 <http://www.bmecat.org/>

12 <http://www.opentrans.de/>

file formats; active implementation based on programming interfaces does not necessarily exist.

2.7.3 Representation in Semantic Technologies

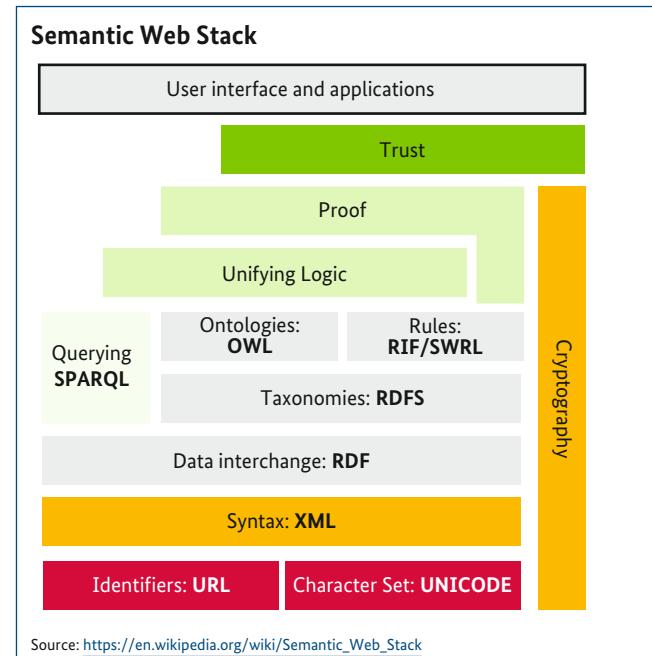
For realisation of the global project “Semantic Web” the international ICT community and also the World Wide Web Consortium (W3C) created a whole series of complementary standards and technologies. These are organised in the so-called “Semantic Web Stack”¹³:

Some of these terms should be explained briefly:

URIs serve as globally unique identifiers of concepts. They are generally known in their special type, URL, e.g. <http://www.zvei.org/Themen/Industrie40/Seiten/default.aspx>. Thus “<http://>” refers to the so-called URI Schema, so to speak the nature of the resource, “www.zvei.org/” refers to the general domain which is allocated on a globally unique basis by a recognised authority¹⁴, and “[Themen/Industrie40/Seiten/default.aspx](http://www.zvei.org/Themen/Industrie40/Seiten/default.aspx)” to the part of the resource/path that can be independently administered within the domain of an organisation. The concept of URIs thus explains how globally unique identifiers can be easily created and also be capable of providing ICT services by the same means¹⁵.

XML is used as syntax for a data representation, thus a format, such that characters from a defined character set, e.g. UNICODE, which can be formed into valid data units and structures of these.

RDF, the “Resource Description Framework”, is used for the formulation of logical statements regarding resources. It breaks each logical statement down into one or more “triplets” which correspond to the form “subject – predicate – object”, with each individual item of them represented as a single URI¹⁶ for unambiguous ICT processing.



Thus with RDF statements about individuals can be made. The chart below illustrates this by way of example:

Examples for semantic statements about an individual NBB1-3M22-E2

Subject	Predicate	Object
NBB1-3M22-E2	isA	proximity sensor (AAA110)
NBB1-3M22-E2	isProducedBy	Pepperl und Fuchs
NBB1-3M22-E2	hasOutputCurrent	0.1 A
NBB1-3M22-E2	hasOutputDiameter	3 mm

13 https://en.wikipedia.org/wiki/Semantic_Web_Stack

14 E.g. Internet Corporation for Assigned Names and Numbers (ICANN)

15 E.g. via REST interface, https://de.wikipedia.org/wiki/Representational_State_Transfer

16 https://de.wikipedia.org/wiki/Resource_Description_Framework

RDFS, the “Resource Description Framework Schema”, allows as a taxonomy definitions as to how logical statements should and may be formulated for a certain knowledge domain by means of RDF. It thus depicts inter-relationships and structures of knowledge and together with RDF allows the formulation of simple ontologies. With RDFS it is possible to make statements about classes (sets of individuals).

OWL, the “Web Ontology Language”, lifts these definitions to the next level and allows the formulation of more complex relationships, structures and conditions for the formulation of more complex ontologies.

SPARQL, the “SPARQL Protocol And RDF Query Language” defines a language for the retrieval of content from ontologies, e.g. in RDF format. Thus for example a request to list possible proximity sensors and their output currents, which meet a size range, can be stated as follows¹⁷:

Typical SPARQL request

```
PREFIX abc: <http://example.com/exampleOntology#>
SELECT ?current ?company
WHERE {
  ?x a abc:ProximitySensor .
  ?x abc:hasOutputDiameter ?y .
  ?x abc:isProducedBy ?company .
  ?x abc:hasOutputCurrent ?current .
  FILTER ( ?y < 4 )
}
```

Source: ZVEI SG Modelle und Standards

Together these standards and technologies constitute an ICT toolbox for which exist many different implementations and established processes for knowledge generation already¹⁸.

Mapping of IEC 61360 properties in RDF is trivial and also possible to be mapped in both directions:

Example for mapping of an IEC 61360 property to semantic statements

Subject	Predicate	Object
AAE867	hasPreferredName	output current
AAE867	hasSymbol	lopen
AAE867	hasPrimaryUnit	A
AAE867	hasDefinition	maximum dc output current of a semiconductor inductive proximity sensor at specified supply voltage
AAE867	hasDataType	LEVEL(MAX) OF REAL_MEASURE_TYPE
AAE867	hasFormat	NR2 S..3.3

2.8 Examination of various sets of properties

For discussion of the structure of the Administration Shell it is relevant which sets of properties must be taken into account, from which sources these originate (also organisational) and how cross-linking between them can be achieved. The following sections comment on certain fundamental observations.

2.8.1 Many matters characterise properties even now

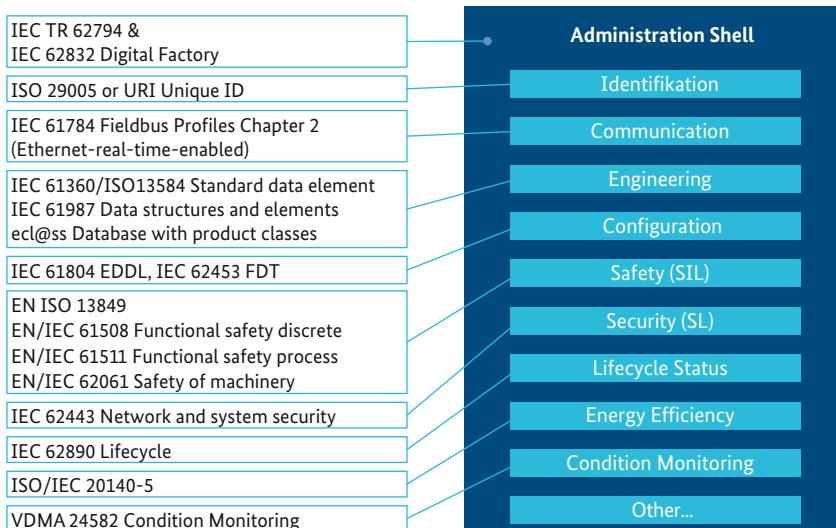
Many of today's international norms and standards specify classes and feature definitions and even values which can be portrayed in properties and which are crucial for a future Administration Shell of I4.0 Components. The following image shows some examples:

17 Similarly: <https://de.wikipedia.org/wiki/SPARQL>

18 E.g. <http://www.w3.org/wiki/SparqlImplementations>, <http://lod-cloud.net/>, <http://schema.org/>

Examples of norms and standards providing properties for submodels of the Administration Shell

Examples of content of the Administration Shell



Source: ZVEI SG Modelle und Standards

Many of the norms and standards (domains) pictured above will in future be developed independently from Industrie 4.0 definitions. Also, the asset of an I4.0 Component will determine the location within the RAMI 4.0 Model and thus also which norms and standards must be addressed in a specific Administration Shell. The following therefore applies:

Requirement: The Administration Shell should be able to accept properties from different domains in mutually distinct submodels which can be version-controlled and maintained independently of each other.

Requirements in regard to security can define a need for a graduated security model for properties, also in different submodels.

2.8.2 Addressing different sets of properties

In accordance with the RAMI 4.0 Model, the four main aspects of Industrie 4.0¹⁹ and the required suitability for a wide range of application scenarios (→ 2.3), an Administration Shell should be capable of holding data and functions for

different engineering disciplines, different life cycle phases and different application and analysis scenarios. The quantities of different properties to support should be correspondingly high and broad-based.

Different sets of property definitions



Source: ZVEI SG Modelle und Standards

Without loss of generality different kind of property sets can be identified:

Hard standards: For protection of the core tasks of a domain, so-called "hard standards" were created in the past and will also be created in future. Such hard standards stipulate commitment to a respectively small quantity of properties which must exist in order to meet necessary requirements in regard to the domain. These can be properties which guarantee the interoperability of components (e.g. DIN ISO 15552 or IEC 61987-13:2012) or which can describe the key figures for a specific component (e.g. pursuant to DIN ISO 61551-3). By their very nature these standards only define a relatively small set of properties; the process of their definition requires a relatively long runtime.

Consensual or cold standards: For a continuous increase in interoperability, e.g. for the procurement phase, manufacturers and organisations frequently agree on common data formats (e.g. STEP) and sets of property (e.g. eCl@ss). These can be established more quickly than by means of hard standards and can be more extensive in their formulation. Frequently optional or alternative data are permitted in the description of an item of subject matter. This optionality must therefore be taken into account by the processing systems.

Free property sets may be formed by means of a wide range of data formats or internal or company standards. These property sets can be formed rapidly and easily and can also serve to address the specificities of individual assets and the USPs of individual manufacturers in a suitable manner. They comprise by far the largest set of properties, but most likely cannot be used by different manufacturers. The Digital Factory (→ 2.4) similarly makes provision for lists of free property.

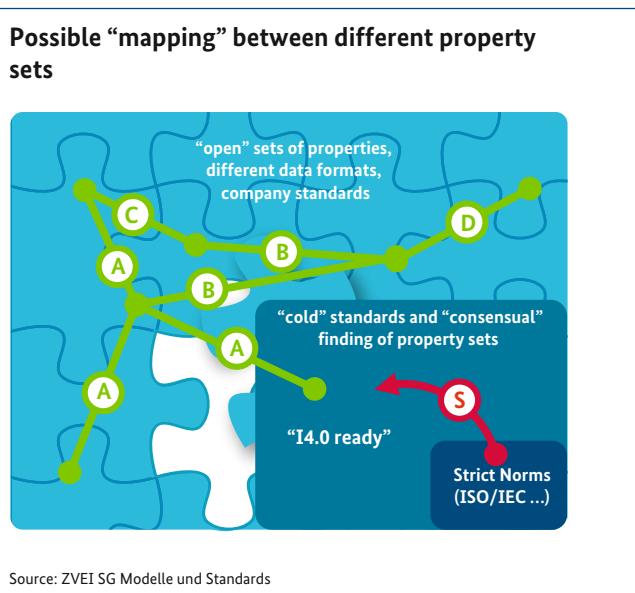
None of these sets of property can be excluded a priori for the Administration Shell; by contrast it must be taken into account:

Requirement: The Administration Shell should be capable of including properties from a wide range of property sets and domains and of differentiating them from each other.

Requirement: For finding definitions within each relevant property sets, in the framework of Industrie 4.0 different procedural models should be allowed which respectively meet the requirements of hard standards, consensually agreed standards and free property sets.

2.8.3 Mapping between Property sets of different Domains

The approach of standardisation by means of Industrie 4.0 creates the possibility of using new developments and technical systems across industries and domains. Value-added networks should be formed between a wide range of partners and new partnerships should also be made possible. On one hand this means that the Administration Shell structurally must be able to meet the requirements of various disciplines and domains. On the other hand, in such new partnerships, it must be possible to transfer properties which have been defined and stated in one knowledge domain to another domains. This approach creates the possibility for existing property sets, even when they come from different domains, to be used directly. To guarantee interoperability it must be possible to relate the properties to each other by means of "Mapping":



The left image shows by way of example how different domains are related to each other by means of a network of mappings (A), (B), (C), (D). They then constitute a graph of knowledge domains, which with appropriate mappings can be strongly connected, thus for each required property in a domain providing a targeted mapping into a property of any other domain.

Requirement: The Administration Shell should be designed such that an internal or external system or organisation can implement an appropriate mapping between different property sets and domains.

Ideally also required

Additional requirement: Suitable mappings between different sets of properties and domains should be designed such that a meaningful, strongly connected graph of mappings can be implemented. Individual mappings can be defined in a distributed approach.

The above image also suggests that the definitions of hard standards should be depicted in the sets of consensually detected properties, ideally by means of a 1:1 mapping.

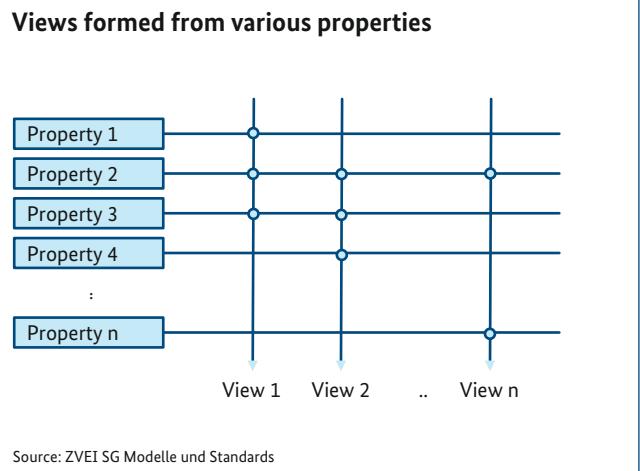
Requirement: Prerequisites should be created which map the given property sets of hard standards (IEC/ISO) in a suitable manner into the group of consensually detected properties.

3. Structure of the Administration Shell

In this section a structure for the Administration Shell will be designed which meets the requirements of Industrie 4.0 in general and the already implemented definitions of the different working groups. It should be based on established concepts in both automation and ICT technologies and be equipped for future developments regarding the relevant I4.0 aspects (Horizontal Integration, Vertical Integration, Integrated Engineering and Interaction with Humans)..

3.1 Views

In accordance with the above discussions the Administration Shell should be able to include data and functions in respect of all conceivable application scenarios (→ 2.3, 2.8.2) and should be able to support a wide range of systems and life cycle phases from the factory²⁰. Not every property (and thus also data or functions) is relevant to each system and each life cycle phase. It is therefore stipulated that each property should be allocated to one or more so-called views²¹:



This classification must be carried out for every property and can only in very few cases be determined on an automated basis. In addition views only make sense if the retrieving systems and life cycle phases have specific expectations of the information sets provided by the views; the views must therefore be known a priori and be defined with sufficient accuracy by means of best practice. For this reason this document establishes the necessary basic views a priori. The table below states the **basic views** and proposes appropriate examples in each case.

Defined basic views for the Administration Shell

Basic view	Best practice/ examples
Business	Data and functions are deposited which allow judging on the business suitability and performance of a component in the life cycle phases Procurement, Design, Operation and Realisation. Examples: prices, terms of delivery, order codes
Constructive	Contains properties relevant for the constructive deployment of the component, thus for selection and building structure. Contains a structure classification system pursuant to EN 81346. Contains numerous properties in respect of physical dimensions and regarding start, processing and output values of the component. Contains a modular view of subcomponents or a device structure. Allows an automation view with inputs and outputs of different signal types.
Performance	Describes performance and behavioural characteristics in order to allow a summary assessment and Virtual Commissioning (V-IBN) of an overall system.
Functional	Makes statements on the function pursuant to EN 81346 and on the function of the subcomponents. Here location of the individual functions of the Technical Functionality also takes place, thus for example so-called "skills", interpretation, commissioning, calculation or diagnosis functions of the component.
Local	Makes statements on positions and local relationships between the component or its parts or inputs and outputs ²² .
Security	Can identify a property as security-relevant. This property should be taken into account for an assessment of security.
Network view	Makes statements in respect of electrical, fluidic, materials flow-related and logical cross-linking of the component.
Life cycle	Contains data on the current situation and historical utilisation in the life cycle of the component. Examples: allocation to production, maintenance protocols and past applications.
People	In all views properties, data and functions should appear such that humans can understand individual elements, inter-relationships and causal chains.

20 Compare also Version 1 of the I4.0 Component

21 The Digital Factory calls these views "ViewElements" and the properties "standardized data elements"

22 The mere position of the component forms part of the header data

Where necessary the Administration Shell can make additional views available. For example these views can be defined in accordance with the lists of properties (LOP) on IEC 61987-10.

The following applies in accordance with the above descriptions:

- Requirement:** The structure of the Administration Shell should always support the aforementioned basic views..
- Requirement:** The structure of the Administration Shell should support further views of the data and functions included.

3.2 Requirements in regard to the Administration Shell

Based on the above descriptions (sections 2.1 to 3.1) therefore, important requirements in regard to the entirety of the structure of the Administration Shell can be formulated²³. These are thus valid irrespective of the content-related requirements for the Administration Shell, which are defined later.

These requirements should be explained individually:

Regarding (a) The Administration Shell consists of body and header

Regarding (b) The body contains information about the respective thing

Regarding (c) The header contains information about utilisation of the asset

Pursuant to Section 2.4 the definitions of the Administration Shell should be appropriate for the Digital Factory. The Administration Shell therefore takes care of structuring in “header” and “body”. The “header” contains information about identification and designation of the tangible assets in the respective factory and where applicable refers to selected capabilities of the assets and views. In the “body” the actual information about the assets is saved which is not directly dependent on the factory-specific²⁴ specifications for utilisation. Thus the “body” becomes the actual data medium. The observations about the structure of the Administration Shell, unless otherwise defined, relate to the “body”.

Requirements in regard to the Administration Shell:

- (a) The Administration Shell consists of body and header
- (b) The body contains information about the respective asset
- (c) The header contains information about utilisation of the asset
- (d) It consists of the Manifest and Component Manager
- (e) The information in the Administration Shell is accessible by means of a service-oriented architecture (API)
- (f) It represents information about different application aspects of the asset
- (g) Structuring according to views (pursuant to MES ISO/IEC81346, Digital Factory BCFLP, ...)
- (h) The Administration Shell has an unique ID
- (i) The asset has an unique ID
- (j) Also a factory is an asset, it has an Administration Shell and is accessible by means of ID
- (k) Types and instances must be identified as such
- (l) The Administration Shell can include references to other Administration Shells or I4.0 information
- (m) Additional properties, e.g. manufacturer-specific, must be possible
- (n) A reliable minimum number of properties must be defined for each Administration Shell

23 SG models and standards, of 8 and 25 June 2015

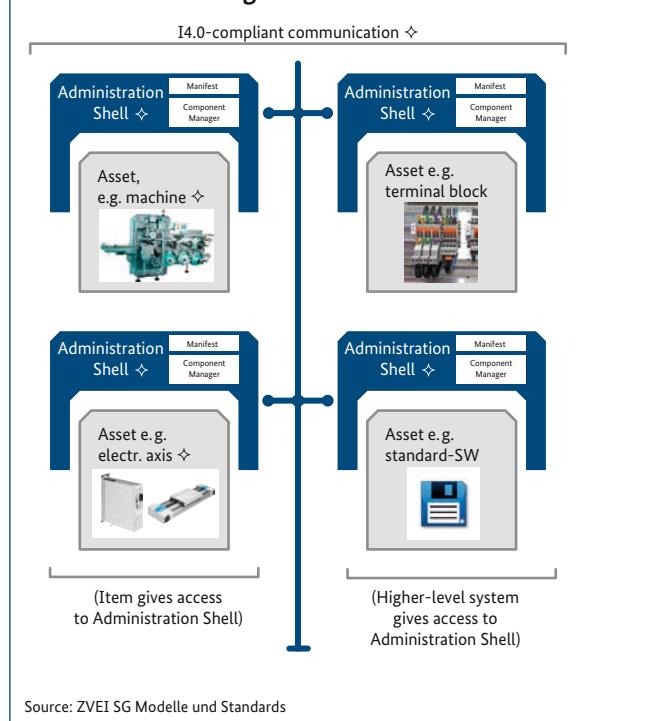
24 This also refers to processing industry facilities (→ 1.2)

Regarding (d) It consists of the Manifest and Component Manager

Regarding (e) The information in the Administration Shell is accessible by means of a service-oriented architecture (API)

As already described in Version 1 of the I4.0 Component, the Administration Shell should have a so-called Manifest and a Component Manager. The Manifest acts as a clearly locatable table of contents for all information, data and functions in the Administration Shell. The Component Manager directly or indirectly constitutes²⁵ an expanded service²⁶ designed to realise both lifelong maintenance of the information included and efficient retrieval options. In principle the capabilities of this service should by means of the I4.0-compliant service-oriented architecture make I4.0-compliant communication available to all participants²⁷, taking into account corresponding security requirements

I4.0-compliant communication, which provides access to a wide range of Administration Shells



Regarding (f) It represents information about different application aspects

Regarding (g) Structuring according to views (pursuant to MES ISO/IEC81346, Digital Factory BCFLP,...)

The application scenarios (→ 2.3) and the correspondingly a priori defined views (→ 3.1) should be respected.

Regarding (h) The Administration Shell has a unique ID

Regarding (i) The asset has a unique ID

This requirement states that both the individual Administration Shells and the associated assets have a unique ID in terms of the passive communication of VDI/ VDE GMA FA 7.21. It should thus be ensured that the link between assets and Administration Shells does not break, even if they are saved in digital repositories or saved in a manner which spans all value-added partners.

Regarding (j) A factory is also an asset, it has an Administration Shell and is accessible by means of ID

The concept of nesting (→ 2.5) should be applicable.

Regarding (k) Types and instances must be identified as such

Administration Shells can be formulated for both types and instances of assets (→ 2.1). It must be possible to differentiate between these. Ideally a data relationship will also be established between component producers and system integrator which, where required, allows updated developments in regard to a type of an asset to be communicated to the system integrator and conversely feedback to be transmitted to the component producer about the component's use (→ 2.2).

²⁵ It is conceivable that the Component Manager is deployed by means of a central service, for example if the I4.0 Component is held in a repository

²⁶ Here service is understood to mean technical IT service, in contrast to the functions of Technical Functionality

²⁷ Needless to say access rights and data security must be considered

Regarding (l) The Administration Shell can include references to other Administration Shells or I4.0 information

For the cross-linking of information to knowledge (→ 2.6) it is important that this can also take place on an over-arching basis. Thus, for example, a component can model the dependencies on other components or can contain a circuit diagram which makes reference to other components.

Regarding (m) Additional properties, e.g. manufacturer-specific, must be possible

The I4.0 Component can only meet future requirements if, in addition to the information content stipulated by standardisation, free properties can also be quickly agreed and processed (→ 2.8.2). The Administration Shell should therefore support this free and proprietary information content and, associated accordingly, necessary collaboration processes (for example like schema.org).

Regarding (n) A reliable minimum number of properties must be defined for each Administration Shell

Other I4.0 Components and additional systems should be able to access and use the properties, data and functions of the Administration Shell. This benefit is enhanced and guaranteed if certain basic properties are always available for many assets. Therefore basic properties should be defined which are always available for certain classes of assets and contain logical values.

3.3 Classes of Property

The two requirements (m), (n) thus result in four classes of property:

Different classes of property

Basic properties	Properties that are mandatory and standardised for all Administration Shells.
Mandatory properties	Properties that are mandatory and standardised for submodels of Administration Shells (→ 2.8.1).
Optional properties	Properties that are standardised but non-mandatory for submodels of Administration Shells.
Free properties	Properties that are non-standardised and non-compulsory for submodels of Administration Shells, e.g. manufacturer-specific properties.

3.4 Requirements in regard to individual Elements of Information

Now that the above section has set out the requirements in regard to the entirety of the structure of the Administration Shell, the following section will more closely examine requirements in regard to individual properties and also data and functions within the Administration Shell (compare → 3.1).

Without loss of generality the concept is established that the Manifest of the Administration Shell (→ 3.2 (d)) administers elements of information in the form of **properties** (→ 2.7.1) and that the Shell itself, as already defined, continues to be capable of accepting data objects and Technical Functionality.

Requirements in regard to individual properties, data and functions:

- (o) The properties and other elements of information in the Administration Shell must be suitable for types and instances
- (p) There must be a capability of hierarchical and countable structuring of the properties
- (q) Properties must be able to reference other properties, also in other Administration Shells
- (r) Properties must be able to reference data and functions of (or at least within) the Administration Shell
- (s) Properties must take into account aspects of information security by means of a graduated guarantee²⁸ of availability, integrity, confidentiality, visibility and authenticity

²⁸ This security classification can include the state “No Security” in the sense of graduated security. However, this must be consciously decided and established.

These requirements should be explained individually:

Regarding (o) The properties and other elements of information in the Administration Shell must be suitable for types and instances

In the same way as the Administration Shell in its entirety the individual properties and further data and functions must differentiate between types and instances of Administration Shell in regard to the respective assets. In individual cases this can also mean that individual properties of an Instance keep a record of whether they have for this instance been added, amended or deleted or whether intended information equality with the data of the Type Administration Shell should be guaranteed (→ 2.1 and 3.2 (k)).

Regarding (p) There must be a capability of hierarchical and countable structuring of the properties

The volume of properties to be organised is rather large (→ 2.8.2) and it is anticipated that it will steadily increase in the progress of Industrie 4.0. The means should therefore ensure that these quantities remain manageable for human and machine. It is thus necessary for properties to be capable of being structured hierarchically. In the same way a property can sometimes contain several equally important alternatives or detailed information, such as a list of languages or certificates. To ensure this, countable structures, e.g. fields or arrays, should be possible. This requirement is enforced by the Constructive View with its modular model design, which breaks structures into hierarchies of individual elements (→ 3.1).

Regarding (q) Properties must be able to reference other properties, also in other Administration Shells

In the same way as for the Administration Shell in its entirety the individual properties must also be able to reference I4.0-compliant entities and information outside their own Administration Shell. In this way cross-linking of information to knowledge is made possible (→ 2.6). Also different knowledge domains (thus for example properties from different standards) can be linked to each other.

In particular views thus become possible (→ 3.1) to the extent that one view/ set of information is hierarchically formed (→ (p)) and the other views link to this by means of referencing²⁹.

This requirement should also allow for named relationships, as is necessary with semantic technologies (→ 2.7.3 RDF/ triplet).

Regarding (r) Properties must be able to reference data and functions of (or at least within) the Administration Shell

The Manifest serves as a clearly locatable table of contents for all information, data and functions within the Administration Shell. Properties are elements for the Manifest of uniform structure and are already standardised (→ 2.7.1). Data and functions on the other hand can be very different, polymorphic and multi-layered. It is therefore stipulated that properties, as part of the Manifest, should be able to reference data and functions within the Administration Shell.

In this manner uniformly locatable properties can serve as an anchor point for any conceivable type of data volume. In the same manner search ability of, description of and access to the functions of Technical Functionality can be guaranteed.

Regarding (s) Properties must take into account aspects of information security by means of a graduated guarantee of availability, integrity, confidentiality, visibility and authenticity

Properties, data and functions will also contain information which not every partner within a value-added network or even within an organisational unit should be able to access or whose integrity and availability should be guaranteed. Therefore the structure of the Administration Shell should from the outset be able to take account of aspects such as access protection, visibility, identity and rights management, confidentiality and integrity. If the executed risk assessment allows, a “No Security” situation can also be realised.

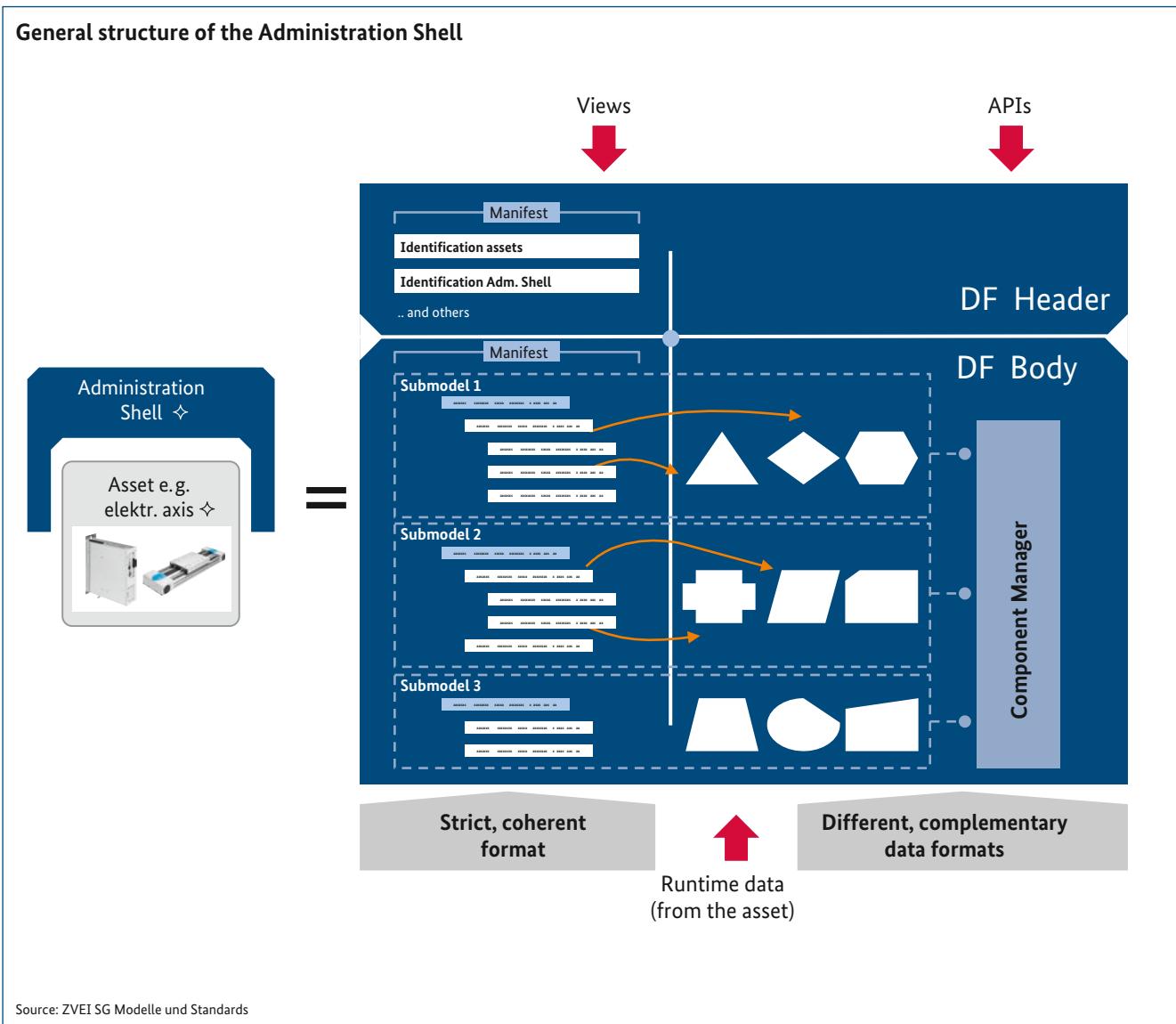
It should be possible to subsequently carry out detailed differentiation of these aspects by means of profiles and views.

3.5 General Structure of the Administration Shell

With the help of the requirements, especially from sections 3.2 and 3.4, a general structure for the Administration Shell can be developed. The chart below still does not represent an ICT-compliant specification, but instead serves to explain important concepts. An ICT-compliant specification is necessary, but can be perceived in a subsequent step in another place.³⁰

With this design the Administration Shell, like a “DF asset” of the Digital Factory, distinguishes between “**header**” and “**body**”. In the “**header**” a list of properties ensures identification and designation of the tangible assets and of the Administration Shell in the respective context and where applicable makes reference to selected capabilities of the assets and views.

The data in the header (including identification of Administration Shell and assets) must be saved as properties in terms of requirement (s) (→ 3.4).



In the “body” can be found the **component manager**, which administers individual **submodels** within the Administration Shell. Each submodel has hierarchically organised **properties**, which refer to individual **data and functions** (white geometric elements). Not shown, but possible, is mutual referencing and the creation of views.

The entirety of the properties of all submodels therefore constitutes the **Manifest** of the Administration Shell, which can hence serve as a clearly locatable table of contents of all data and functions. In this manner it becomes possible for the respective property structures to be arranged in a strict, **standardised format** (based on IEC 61360, → 2.7.1), while for the different data and functions various complementary **data formats and methods of access** are possible.

Outwardly the Administration Shell can where applicable record and depict **runtime data** (from the asset), e.g. the actual position and actual currents for a servo amplifier. It should be possible to outwardly portray the information sets by means of views (→ 3.1). An I4.0-compliant, service-oriented **API** (Application Programmer’s Interface) should make the **services** of the resource manager available externally. A part of these services is lifelong **maintenance of the properties, data and functions** within the Administration Shell, the **addressing and identification of Administration Shells and assets** (→ 3.2 (h), (i)) and an **efficient search** for properties and referenced data and functions. This search should make more than just browsing of properties possible; it should support view support, tolerant search, synonym search and similarity relationships. Support of SPARQL is conceivable (→ 2.7.3).

3.6 Asset Structure

For formulation of Administration Shells it is relevant to know for which assets³¹ these can exist:

- A **software package** made available can represent an important asset of a production system and thus an asset.
- An **asset can have several Administration Shells** which are relevant in different reference frameworks, thus RAMI Models. Hence for example, for its internal purposes the manufacturer of a servo amplifier can maintain an Administration Shell under “Type/ Development” (→ 2.1) and file its internal development data there. For the purposes of its customers this typical manufacturer can provide an external Administration Shell of the model series under “Type/ Usage”. And ultimately, for example, for each instance delivered the respective user can derive and continue to maintain an Administration Shell under “Instance/ Usage”.

The above example also raises the possibility of a requirement for automatic referencing and matching of individual elements of Administration Shells between themselves, for example by being able to execute updating of an Administration Shell of a “Type” into an “Instance” (see also → Requirement (l))

Requirement: Different Administration Shells in respect of an asset must be capable of referencing each other.

In particular elements of an Administration Shell should be able to play the role of a “copy” of the corresponding components from another Administration Shell.

- **One or more assets can be portrayed in an Administration Shell**, for example where mechanical axis, motor, servo amplifier and additional assets constitute a “encapsulate-capable” I4.0 Component³².

³¹ See the first version of the Industrie 4.0 Component in “Realisation Strategy Industrie 4.0” of the platform of April 2014

³² See also “Realisation Strategy Industrie 4.0” of the platform of April 2014, “Capsule Capacity” section

Mapping of several assets in the Administration Shell using an electrical axis system as an example



Source: ZVEI SG Modelle und Standards

The above example depicts a situation in which the Administration Shells of several individual assets, which a manufacturer brings onto the market individually, for example, are consolidated into one Administration Shell, if this typical manufacturer also sells a whole axis system. The following therefore applies, also with the specifications of the Digital Factory (→ 2.4):

Requirement: Individual Administration Shells should, while retaining their structure, be combined into an overall Administration Shell.

3.7 Compatibility with the Digital Factory

The objective of the work in the IEC is to create a standard (known as “Digital Factory”) for electronic, machine-readable representations of production systems which can be used in all life cycle phases of the production systems. At the same time exchange of information between the different participants in the design, construction and operation of a production system should be supported. An important area of focus is the semantic clarity of the information, which is why the standard “Digital Factory” is based on property descriptions of the components of the production system. The utilisation of property descriptions for the components and component groups allows automated processing of the description of the production system – e.g. in validation of the system design. At the same time the standard “Digital Factory” assumes that the data in the description of the production system are centrally administered.

An objective of Industrie 4.0 is automated configuration of a production system in accordance with varying production orders. This objective can only be achieved based on a machine-processable description of the production system

Similarities in the structures of the Administration Shell and the Digital Factory, as exemplified by an electrical axis system



Source: IEC 62832

IEC CD 62832

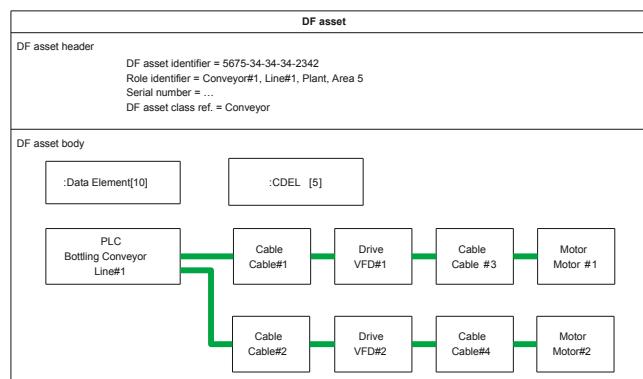


Figure 9 – Example of composite DF asset body

and the products to be manufactured. The information model for Industrie 4.0 is therefore based on the standard “Digital Factory” and extends it to include further required information. In order to achieve the required flexibility for the automated configuration of a production system the production resources are not only represented by passive data elements, but also by independent I4.0 Components, which are not only centrally administered, but which can also be understood as being a distributed repository. The data elements of the Digital Factory (referred to there as DF assets) at the same time form the basis for the Administration Shells of the corresponding I4.0 Components.

3.8 Identifiers

3.8.1 Initial Situation

Various requirements (→ 3.2 (h), (i)) demand the availability and clarity of identifiers (IDs) (compare also → 2.1). In the same way properties and class relationships must be clearly identifiable. A further reference is also the at least passive communication ability of entities (assets), which is defined by VDI/VDE GMA FA 7.21:

Unit with passive communication ability

A physical unit is a unit with passive communication ability if it has a data medium which can be read out from system interfaces. The data medium itself is passive, though it allows reading out of its data and thus e.g. identification of the asset (RFID, barcode, etc.).

Discussion of different aspects of the provision of services (production, logistics and operation), of different industry sectors and different regions suggests that exclusive commitment to a system of identification which must also offer global uniqueness is difficult to achieve. Instead identifiers should be found which indirectly allow association of further identifiers (dereferencing), and which therefore allow the number of variants accepted for Industrie 4.0 to be kept small.

Requirement: An identifier must by means of a finite number of accepted variants achieve unique identification of assets, Administration Shells, properties and class relationships and as far as possible offer global uniqueness.

Requirement: An identifier must as far as possible allow association of further identifiers and of other variants which refer to the same object.

3.8.2 Determination of Identifiers

Thus for the following international and freely available standards appropriate variants of identifiers are determined: (see page 32)

Global identifiers are recognised as being those which allow the interaction of the Administration Shell or its elements with the Administration Shells of other partners in the value-added networks. For these identifiers global uniqueness is required and ISO 29002-5 or URI are specified as variants:

Identification by means of ISO 29002-5

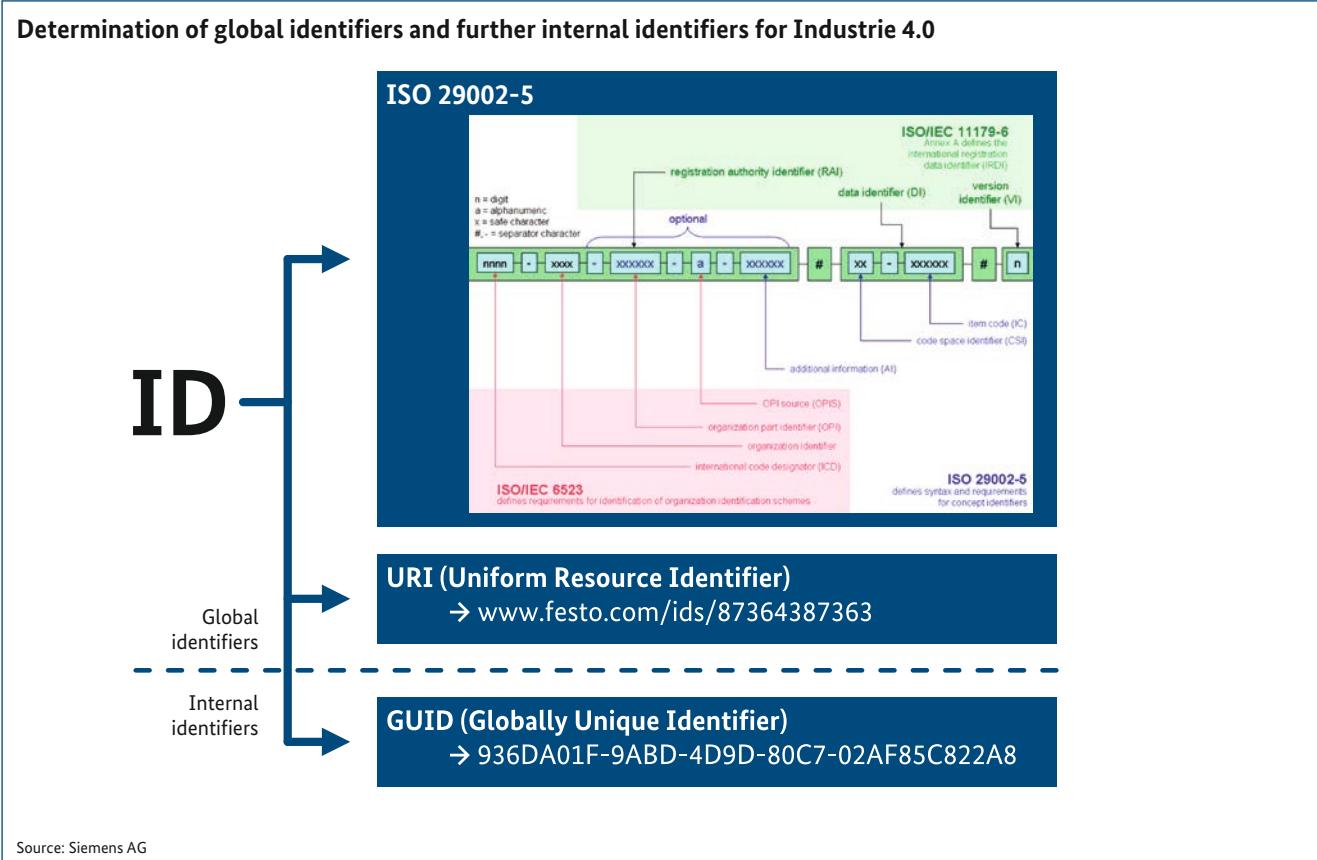
The ISO 29002-5 standard establishes an identification scheme for globally unique identifiers. These are very well suited in particular to norms, standards and consensually agreed property sets and classes.

Dereferencing of other identifiers would require the ICT services of a central registration body.

Identification by means of URIs

The ICT technologies have similarly developed and approved a global standard for identifiers (W3C). As described in Section 2.7.3, by means of URI Schema, domain and path a type-secure, globally unique and distributed volume of identifiers can be realised.

Valid internal identifiers are any which do not need to be accessible to other partners in the value-added networks. These can be identifiers of manufacturers' internal data elements, for example. Other variations can also serve as these identifiers.



3.8.3 Secure added-value Networks

Specific security requirements may demand for secure identities. A summary and concise description of identifiers, identities, unambiguous identities and secure identities, including a matrix chart, is worked out by the workgroup “secure identities” and is not part of this specification³³.

3.8.4 Association of further identifiers

In order to meet the requirement for different identifiers for various aspects of the provision of services (production, logistics and operation) (→ 3.8.1), it makes sense to place in the Administration Shell further properties which hold further indirect identifiers, such as a GS1 identifier, locally. I4.0-compliant communication would therefore be directed against the variants of identifiers established above; by means of an API service the Administration Shell can locate further indirect identifiers:

Requirement: The Administration Shell should exhibit properties which for used global identifiers (IDs) of Administration Shell, assets and properties can also provide indirect further identifiers, such as a GS1 identifier, locally.

3.8.5 Best Practices for Identifiers for Assets and Administration Shells

Pursuant to Section 3.8.1 identifiers can be used for various purposes. As far as identification of assets and Administration Shells is concerned, there are numerous overlaps with further interests in companies. Anhang A gives further information on how these overlaps can be resolved in execution.

33 Plattform Industrie 4.0, Technical overview: Secure identities (April 2016)

4. Methodology for the distributed Formulation of I4.0 Submodels

In this section several approaches are presented which by means of a regulated procedural model allow good, scalable parallel development of several I4.0 submodels at the same time. Overlapping of the individual submodels need to be avoided. Requirements (q) and (r) from Section 3.4 ensure that the submodels relate to each other or can build on each other.

Submodels need to describe their properties, data and functions to be used in respect of a well-defined purpose. The regulated procedural model to be used can be selected accordingly.

In any case an orchestration of the content of the submodels must take place. It is initially assumed that ZVEI/SG "Models and Standards" will take on this function of "responsible office". Nevertheless, the role of "responsible office" can at any time be transferred to another entity.

4.1 Formulation of a I4.0 Submodel on the Basis of an existing Standard

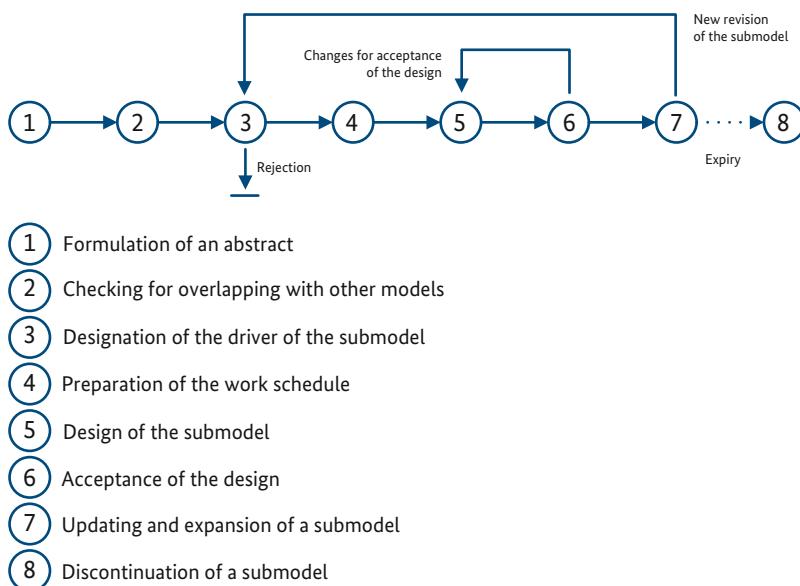
This procedural model is especially well suited for transferring the properties of already existing standardisation projects into submodels. It works on the basis of various steps which are carried out by different parties. In formulating the model attention was paid to compatibility with the IEC proposals for the formulation of standards³⁴.

The individual steps are as follows:

① Formulation of an abstract

(a) An interested party writes an abstract in respect of the arrangement of a submodel. The contents of this abstract incorporate, in a generally understandable form:

Procedural model for the formulation of a submodel, for example for existing standards



Source: ZVEI SG Modelle und Standards

- Name of the project
 - Standard(s) to be used as source
 - Location in RAMI, in particular: life cycle phases concerned, hierarchies concerned, possible links to other models/I4.0 Components
 - Textual description: description of what type of I4.0 Components the submodel is relevant (e.g. "electric servo-axes")
 - Standardisation bodies to be consulted
 - Partners to consult such as manufacturers or associations
- (b) The interested party submits the project to the responsible body.
- ② Checking for “overlap”**
- (a) The responsible body designates a group of standardisation organisations and associations involved with standardisation.
 - (b) Based on the standards referred to in (1), this group checks:
 - whether another submodel is processing the same or closely related standards,
 - whether the standardisation bodies to be consulted have been correctly and fully designated,
 - whether a competing/ contradictory submodel has already been registered (refer to RAMI location).
- (c) The above group makes proposals as to which additional entities, groups, persons or submodels are to be taken into account.
- ③ Designation of the driver of the submodel**
- (a) The interested party drives forward the project in respect of the responsible body. The responsible body examines the project based on the information from (1) and (2). In the event of a competing/ contradictory submodel the responsible body provides clarification in respect of the already existing submodels.
 - (b) The responsible body can declare a proposal to be irrelevant for the submodels of the Industrie 4.0 Component.
 - (c) In the event of a positive decision the responsible body designates a manager/driver for the preparation of the new submodel.
 - (d) The responsible body takes into account the proposals from (2).
 - (e) The responsible body can designate an advisor who assists the driver in execution of the preparation of the submodel. This advisor should have excellent knowledge of RAMI 4.0, I4.0 Components and existing submodels.
 - (f) The result of the discussion is made public.
- ④ Preparation of the work schedule**
- (a) The driver and advisor speak to the standardisation bodies and additional groups, entities and persons designated in (1) and (3) and form a working group. The group can also be a subset of a standardisation body.
 - (b) They jointly agree on a work schedule, which is documented. The objective of the work schedule is a votable draft of a revision of the respective submodel.
 - (c) Working meetings can be organised at will and can also be held in the course of existing meetings.

⑤ Design of the submodel

- (a) The parties referred to in (4) hold the working meetings. The advisor has an advisory function.
- (b) If necessary the driver and/or advisor match content with the responsible body.
- (c) The submodel provides details of properties, their structures and the data and function objects associated with them.
- (d) The properties and their structures are to be implemented in accordance with the specifications of the Industrie 4.0 reference architecture.
- (e) Where data and function objects differ in their formats and designs from the preferred formats designated in the Industrie 4.0 reference architecture, this must be justified.
- (f) The design also stipulates which properties, data and functions must be provided at all times (mandatory properties) and which of these properties, data and functions are to be used on an optional or alternative basis (→ 3.3).
- (g) For optional/ alternative use of this type “best practices” are to be described.
- (h) The design can also propose properties that are mandatory and standardised for all Administration Shells (basic properties, → 3.3). Implementation of these properties is the prerogative of the responsible body.
- (i) The design should also already give instructions and recommendations in respect of protection of the properties, data and functions within the meaning of data security.
- (j) For a consultation regarding a first final revision the later users of the respective submodel, thus distributors of components and software for example and manufacturing companies from representative industry sectors, should be considered accordingly in an exemplary manner.

(k) Unless decided otherwise by the standardisation bodies, a consultation will take place based on a simple majority.

⑥ Acceptance of the design

- (a) The proposed revision of the submodel is submitted to the responsible body for examination.
- (b) The responsible body is responsible for ensuring that the definitions of the submodel
 - do not contradict definitions of other submodels or of other parts of the current Industrie 4.0 reference architecture and
 - in respect of choice of name, identifiers etc. are appropriately formulated and sufficiently adapted to other submodels.
- (c) Where required by the driver or advisor changes to the design can be resubmitted to the working group for changes and for subsequent adaptation.
- (d) In the event of a positive examination the responsible body accepts the design, allocates a revision number for this submodel and organises publication of the content.
- (e) The responsible body can accept the (5) basic properties proposed and implement them in an appropriate manner³⁵.

⑦ Updating and extension of a submodel

- (a) The driver, the advisor or the respective representatives of the standardisation bodies designated in (1) and (3) check at regular intervals of two years or less whether an existing submodel should be updated and expanded.
- (b) The responsible body can similarly initiate updating or expansion.

³⁵ For example make a submodel and make the basic properties defined therein standardised and compulsory, integrate basic properties in a submodel or make provision for them separately in the Industrie 4.0 reference architecture

(c) In both cases the abstract from (1) should be updated.

(d) Steps (3) to (6) are subsequently repeated.

⑧ Discontinuation of a submodel

- (a) No provision is initially made for expiry of a submodel, as they are preserved in the Administration Shells over the life cycle of the assets.
- (b) An examination/ initiative pursuant to (7) can result in a “Discontinued” flag being posted for this submodel.

(1) Domain Expert, (2) Knowledge Engineer, (3) Application Developer and (4) Scientist (see Appendix B).

(c) At the heart of this development environment is a distributed version control, e.g. “Git” (see Appendix B), with the help of which each authorised user can make changes to the submodel.

(3) Agile development

(a) Before development begins, the authorised developers can under certain circumstances define a breakdown of the development into different branches. These can also be carried out based on the different developer roles.

(b) During development each authorised user can obtain the latest version of the submodel, change it and upload it to the versioning system as a new version.

(c) Where the changes are only to be understood as a proposal, the developers can set up additional new development branches to work on. Merging of the branches is the responsibility of the core team.

(d) The Domain Experts are responsible for the content-related accuracy of the submodels, while the Knowledge Engineers are responsible for the quality of the submodels from the modelling perspective. In particular the developers in these two roles are responsible for continuous further development.

4.2 Agile approach to the Identification of new Content

Detailed structuring of the procedural models ensures a certain level of quality in the models produced. Nevertheless they result in cumbersome processes which can be very resource-intensive. A first version of a submodel that is ready for productive use takes a relatively long time. However, agile approaches ensure that very rapidly usable versions are available which can be continuously refined. In less critical systems which can be operated with interim solutions from submodels, it therefore makes sense to use agile approaches to model development.

(1) Proposal of a submodel to be developed in an agile manner

- (a) If demand for the development of a submodel for a subdomain by means of application of an agile approach is established by any user, this domain is briefly presented and described so that it can be synchronised with the activities already ongoing.

(2) Examination and approval by the responsible body

- (a) The responsible body examines the suitability of the proposed subdomain in respect of the applicability of an agile approach to development.
- (b) If the proposal is accepted the responsible body provides the necessary development environment and grants the interested participants access rights. A core team is defined which is responsible for the establishment of future Releases. Furthermore the developers are subdivided into four roles:

(4) Acceptance of the design

- (a) The core team of developers can at any point in time of development define so-called Release Versions which exhibit a certain level of quality and stability.
- (b) Before these are released as such, they are submitted to the responsible body. The responsible body decides whether the proposed version will in fact be accepted and released as a Release Version.

Literature Appendices

Literature

Technical overview: Secure identities, Publication of the Plattform Industrie 4.0, April 2016

Appendix A: best practices for identifiers for assets and administration shells

1. Introduction

Identification is one of the key topics for Industrie 4.0. This appendix is concerned with the unambiguous designation and identification of things, assets and objects including software, documents, people, etc. It must be ensured that identification is globally unique.

This chapter describes identification in general terms, the prerequisites and principles, and presents some examples of execution.

2. Who needs Identification?

Identification is necessary for various processes within Industrie 4.0. It is required in various processes, but the various different processes require different methods of execution in practice:

2.1 Technical Process

For unambiguous classification of objects in Industrie 4.0 each object included in the Industrie 4.0 network requires unique identification.

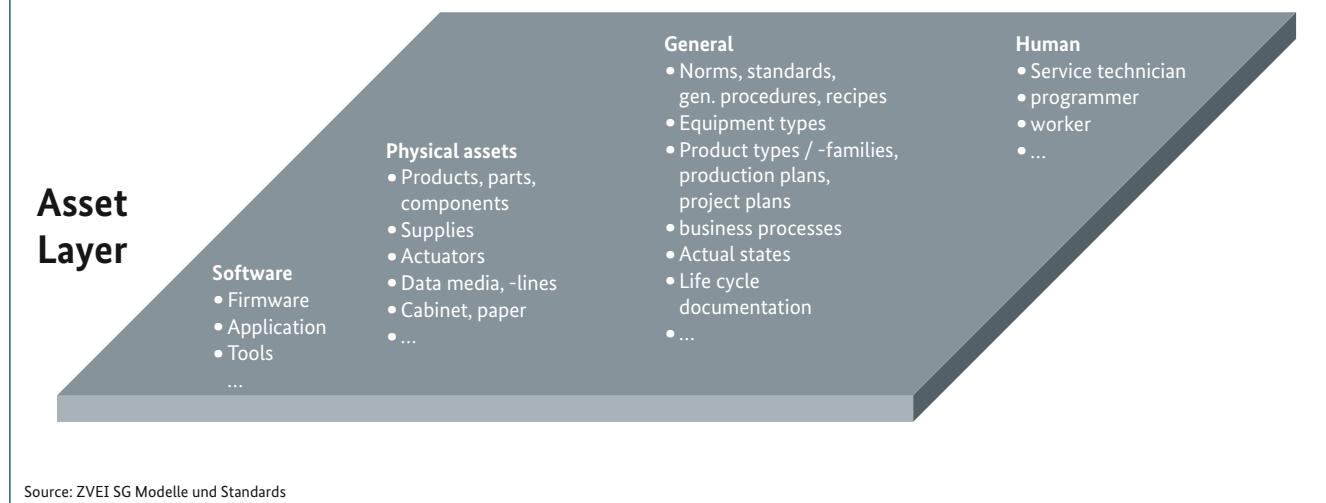
This includes all objects or assets as they are defined in the Asset Layer of RAMI 4.0.

However, all data and properties in the Information Layer must also be clearly discernible or identifiable. Unique identification of the functions and software components is an additional issue.

2.2 Logistics

Identification throughout the movement of goods continues to play an important role. This concerns the entire logistics chain such as products, though also packaging, transport vehicles, containers, warehousing, etc. Vendor parts are also affected, which while not themselves being sold, are included in a product or system. Several identification marks are merged in a device or system and the device/ system similarly receives an identification mark. The interrelationships between the individual parts and thus also the identification marks are the responsibility of the manufacturers.

Possible objects on the Asset Layer of RAMI 4.0



Identification as related to different industrial processes

	Production	Supply chain	Sales, Marketing, Service
Target	Simple, fast and error free identification in production	Simple, fast and error free identification in material flow	Customer access/loyalty, extended services: e.g. autotuning, commissioning, etc.
Customer	Internal processes	Internal processes, suppliers & customer processes	Potential buyers, customers
Technology	PLC interface, industrial readers	Backend systems (ERP), industrial readers	Standard smartphone, no specific apps
Contents	Material numbers, serial number	Material numbers, serial number, quantities & additions	Catalogue information, spare parts, sales contacts, support, CAD, etc.
Further comment	Secure labeling, attached to product or carrier	Secure normed labeling, nearly all products, utilities like trays, package units, etc.	Additional labeling, Selected products, Simple and fast
Market usage	Barcode, Data Matrix, RFID, QR-Code	Barcode, Data Matrix, RFID, QR-Code	QR-Code, Data Matrix

2.3 Distribution, After Sales, Marketing

Customers and prospects need quick and simple access to information about the product. This can be information about the product, though also about support (commissioning) or service (e.g. replacement parts, service engineers, remote maintenance, etc.).

Globally unique identification is similarly necessary for these processes.

3. Principles

- As few characters as possible in the identification, properties, features and further data can be retrieved under the identification
- ID as short as possible in order also to be able to mark small parts which have little space for a code
- The object should be identifiable throughout the world by means of one identification (ID), one ID String
- The ID String consists of ASCII characters, only a few special characters should be used (see below, “use of characters”)
- The ID is always globally unique (“my ID only exists 1x, I am an individual”)
- Without overlaps with other users, i.e. inclusion of the manufacturer/ company

- Companies internally ensure non-overlapping of their IDs
- If codes are used then the issuing entity of the codes must also be shown
- Technical data, properties, features, etc. are not components of the ID
- The ID can include descriptive elements such as material number, product name and type
- ID is independent of technical realisation, it should be realisable at least with Quick Response Code (QR Code), Data Matrix Code (DMC), Near Field Communication (NFC) and RFID (Radio-Frequency Identification). There may be further technical realisations in future.

4. Information for identification

In the manufacturing sector various classification and identification systems are used in order to unambiguously identify parts, products, people, software and services and thus also to make them technically workable.

Organisation (company name, manufacturer)

Each organisation or company internally ensures non-overlapping issue of identification marks. If identification is stated together with the company/ organisation, unique identification can take place. A prerequisite is that the organisation name or company name is also globally unique.

Product name

Product names are mostly identical across all manufacturers and identify the same class or category of product, e.g. servomotor, control unit, pressure sensor. In addition manufacturers have additional abbreviations of their own, which specify the product name even more precisely. They belong to the product name (e.g. servo motor MKS140).

Product/material (type)

A product is a usable and saleable asset resulting from production. Examples of superordinate product categories are services (e.g. development), software (e.g. computer program), hardware (e.g. valve) and process engineering products (e.g. lubricants).

In addition to the product name and its abbreviation each possible version (e.g. with options) receives an exact product and material designation. This is still a type whose version is precisely designated, however. It is identified by a unique material number.

Serial number (instance)

The serial number is a unique combination of ASCII letters, numbers and/or characters selected by the manufacturer and intended to distinguish a product from other products with the same product designation. (Identification of instances of the same type.) A given serial number is only issued on a single occasion by a manufacturer and is thus unique

Examples:

If an individual instance is considered indication of at least the organisation/ manufacturer and serial number are necessary (e.g. a specific servomotor, a specific frequency converter, a specific sensor).

With many assets no serial number is allocated, e.g. screws, cables, etc. Here naming of the manufacturer and product number is sufficient for unique identification.

Remarks on the use of characters

1) Permitted characters

Numerical: 0...9 and alphanumeric: letters A...Z and a...z
(no special national characters, e.g. ä, å, ...)

Additional characters should be avoided:
blank spaces ! # \$ % & ' () * + - _ ~ . , / : ; = ? @ []

Certain characters identify and separate the individual segments of a URL and facilitate its processing:

- A question mark (?) introduces the specific path specifications (query string) of the URL
- An ampersand (&) acts as a separator for the parameter (data field)
- An equals sign (=) stands between the name of a parameter (= data field) and its value (= data content)

Please note: when used these characters must therefore be previously encoded (see also RFC 3986)

2) String lengths

Some devices and also software packages make it possible to transfer an almost infinite volume of characters. However, in transmission some are limited to 245 characters. In order to guarantee safe transfer, therefore, a maximum of 245 encoded characters should be transferred in a string.

ID implementation

Sample values:

Organisation with issuing entity:	7777777 (GS1) 123456789 (DUNS) http://Firmaxy.com (DNS)
Product name	Servomotor_MKS140
Material number	1122334455
Serial number	667788990012345

1) Typical implementation with URL

The respective company is responsible for ensuring that information on the product is also retrievable under the Web address.

Example: Access to a unique serial number in the event of a service call

<http://Firmaxy.com/667788990012345>
<http://Firmaxy.com/ID/667788990012345>
<http://Firmaxy.com?s=667788990012345>

Example: Access to development documentation of a product/material

http://Firmaxy.com/Servomotor_MKS140/1122334455
<http://Firmaxy.com/1122334455>
<http://Firmaxy.com?m=1122334455>

2) Typical execution with ISO

The ISO/IEC Schema allows a certain amount of freedom in selection of the Code Schema

Issuing entity	e.g. UN for Dun & Bradstreet
Application Family	
Identifiers (AFI) identification	Object class, A1 for product
Data Identifier	Structure of the ID String, 25S or 37S recommended

a) Example with Issuing Entity = UN, AFI = A1,
Data Identifier = 25S executed as RFID code

RFID, URI:
urn:iso:id:obj:25SUN123456789Servomotor_MKS1401122334455667788990012345

b) Example with Issuing Entity = UN, AFI = A1,
Data Identifier = 37S executed as RFID code

RFID, URI:
urn:iso:id:obj:37SUN.123456789.Servomotor_MKS140122334455+667788990012345

5. Technical Realisation

The technical realisation of a code relating to the products can vary to a very great extent. Below are practical examples of implementation by companies participating in the Plattform Industrie 4.0:

Company A:



<http://dc-qr.com?m=R911345469&t=HMU05.1N-F0140-0350-N-A4-D7-N1N-NNNN&s=7260403890047>

Company B:



Attached is an example of how today a link can be made from a QR code for an article to the data in the eShop.

The seven-digit article number is allocated in the master data record upon creation of the article. In the PLM system this article number is supplemented by a revision number from the product documentation.

In respect of Industrie 4.0 we would like to see agreement on one procedure (Global Unique ID).

www.phoenixcontact.net/product/2832632

Company C:



Version 1 – URL+“Enter“+serial number.

<http://shop.murrelektronik.de/en/7000-14041-0000000>
(Enter)
123456789123456789



Version 2 - URL+“Enter“+serial number.

<http://me23.me/7000-14041-0000000/123456789123456789>
Version2 (24px=1,4cm)



Current (22px=1,3cm)

Company D:

<http://go2se.com/Referenz> reference for an object. Multiple sections of data and requests per “/” with appropriate key-words for a object.

An Industrie 4.0 Component can therefore be simply linked like our products by means of ref = ID or sn = ID

e.g. <http://go2se.com/ref=TM241CE24T>
<http://go2se.com/ref=TSCEGWB13FA0>

Or as part of an object's data (though “data=” can also be replaced by other abbreviations).

(notional link)

[http://go2se.com/sn=TM241CE24T/Data=\(I4.0ID\)](http://go2se.com/sn=TM241CE24T/Data=(I4.0ID))

Company E:



Valve terminal VTUG

<http://pk.festo.com/3s7pl9xL2QK>

Proportional pressure control valve – VPPM



<http://pk.festo.com/3S7PL9JS583>

MS6-SV soft-start and exhaust valve



<http://pk.festo.com/3S7PL810PFQ>

Appendice B: lightweight approach to the standardisation of vocabularies for semantic interoperability

Dr. Gökhan Coskun, Prof. Sören Auer, Fraunhofer IAIS

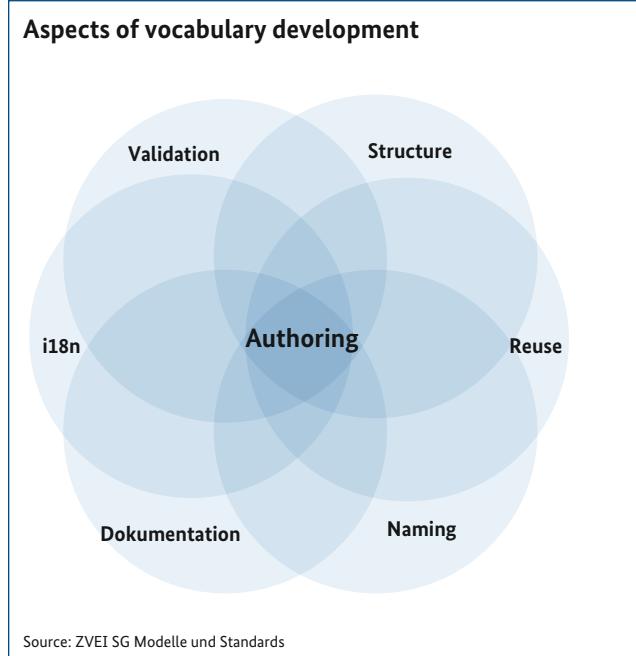
1. Introduction

Definition of vocabularies and software architectures as standards for the realisation of data and system integration is a common approach to addressing the heterogeneity of complex systems. At the same time the traditional approach of defining a standard is a drawn-out and time-consuming process. Inspired by the dynamic, diverse and open Web, in the present document a lightweight and collaborative approach to the development of vocabularies is defined. This is intended to complement the currently used standardisation processes with a newer more agile approach. The primary objective is to strengthen the awareness of the standardisation bodies regarding the pragmatic, lightweight and agile processes of the Web and to integrate the relevant experiences and findings of this domain in the work process for the definition of standards

2. Requirements of collaborative vocabulary development

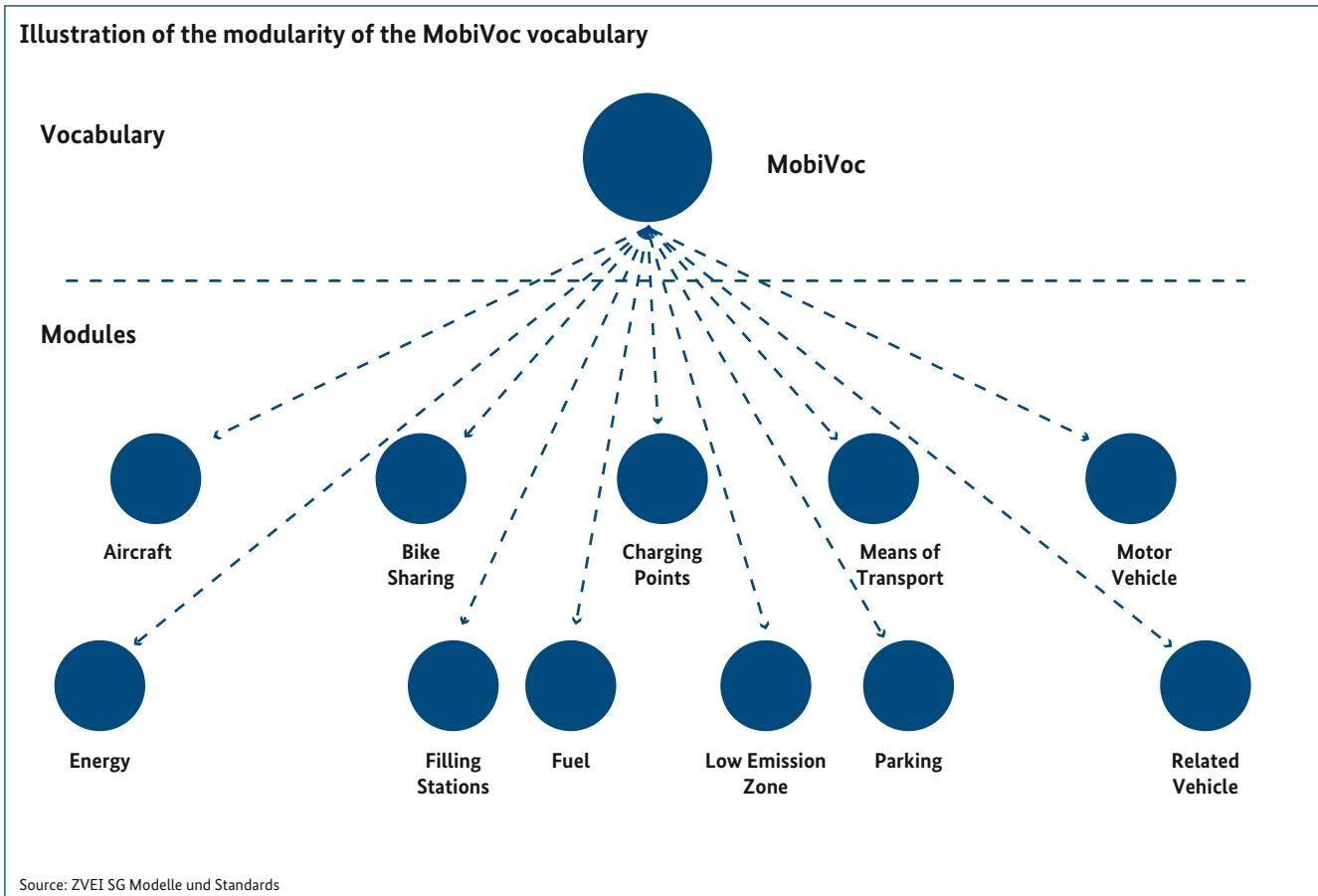
In distributed and heterogeneous software systems common data formats and vocabularies – usually also called metadata – provide a common understanding of the domain. They are used to facilitate the exchange of data between the systems. That is to say the internal data of the systems are transferred into the representation by means of these vocabularies before the exchange and subsequently sent. For this part the recipient can translate these into his own internal presentation. Due to this close linking of the exchange formats with the internal representations the problem of vocabulary development has mostly been viewed as part of the software development and has been ignored as an independent problem. Firstly with the growing need to intensify integration of data in order to draw more benefits from them on the one hand and to provide users with better support by means of integrated systems on the other hand, the problem of vocabulary development was given more attention. This need was not least reinforced by the success of the Web, the countless successful mash-ups in the Web and the vision of the semantic Web. In particular in the context of the semantic Web numerous methodologies for the creation of expressive ontologies ini-

tially came into being with the focus on knowledge representation in the sense of artificial intelligence. In the meantime more pragmatic approaches are increasingly gaining ground, and these regard vocabulary development as an independent problem, while at the same time also applying existing techniques and tools of agile, pragmatic and collaborative software development. First and foremost this results in the challenge of adapting these techniques and tools as well as possible to the requirements of vocabulary development. Based on the experience of past work together with analysis of existing Web vocabularies, the following six aspects (see Fig. 1) were identified as being the most important requirements of vocabulary development.



Validation

Validation is an important aspect of vocabulary development. In this connection the correctness of the vocabularies in respect of syntax, completeness and consistency as well as availability is examined. During the entire life cycle of a vocabulary its validity must be guaranteed. Therefore continuous validation during the entire development process is necessary.



Structuring

The size and complexity of vocabularies can steadily increase during development, so after a certain length of time both further development and support and maintenance in particular are made more difficult by this. In order to maintain lightness and agility, appropriate structuring, partitioning and modularisation must take place. The responsibilities of the developers in respect of the subvocabularies must be explained clearly and explicitly. By way of example Chart 2 illustrates the breakdown into subvocabularies by means of the MobiVoc³⁶ vocabulary.

Reuse

The availability of numerous vocabularies in the Web – in particular in the context of the Linking Open Data Cloud endeavours – suggests the value of looking for at least

partly reusable components prior to total new development. If successful this not only saves time, but also ensures a certain degree of quality, as it can be assumed that frequently reused vocabularies also undergo frequent appraisals and have been frequently improved.

Designation

A further important aspect of vocabulary development is selection of the correct designations for the terms. Consistent and appropriate designation avoids conflicts, improves clarity and thus increases reusability. The use of designation schemes such as CamelCase is highly recommended.

36 <http://www.mobivoc.org>

Documentation

In order both to be able to follow the overall development process and to be able to quickly and correctly understand the current content of a vocabulary, comprehensive documentation is essential. The developers must have the ability to quickly and simply document their modifications. These must be reliably recorded and retrievable in an appropriate format both for the members of the developer team and for potential reusers.

Multilingualism

In view of the fact that developer teams are increasingly spread across the world and software systems are used internationally independent of language, an important aspect is the multilingualism of vocabularies. It is advisable to translate the vocabularies at least into English in order to increase the number of potential reusers and thus the number of assimilable software systems.

3. VoCol – Lightweight development of vocabularies

VoCol is a lightweight approach to collaborative vocabulary development which is heavily oriented towards proven software development practices and environments. These are only slightly adapted in order that the described requirements can be met. Essentially the VoCol environment is the distributed versioning system Git, which is not only enjoying growing popularity in software development projects, but is already used in the development of Web vocabularies. Particularly worthy of mention at this juncture are Schema.org³⁷, Description of a Project³⁸ (DOAP) and Music Ontology³⁹. Besides this principles and roles are defined.

³⁷ <https://github.com/schemaorg/schemaorg>

³⁸ <https://github.com/edumbill/doap>

³⁹ <https://github.com/motools/musicontology>

3.1 Methodological principles

Unlike extensive methodologies, VoCol limits itself with defining principles for development whose pursuit is promising for the efficiency and effectiveness of the overall development process.

- **Quick and simple editability**

Access to the development process must be simple and uncomplicated so that the developers can co-develop with simple tools such as a Web browser. Each interested party should have the opportunity to participate and make his contribution.

- **Constant adaptation to changing conditions**

Both the parts of the vocabularies and the elements of the development environment should be replaceable in a fast and uncomplicated manner in order to allow the possibility of adjustment of the changes. The teams should act on a largely independent basis and should also have an open-minded attitude to pronounced changes.

- **Address user satisfaction by means of quick delivery**

Versions with differing levels of stability should be made available quickly and automatically. In this way vocabularies are viewed as continuously changing products and not as unique and completed development activities. Besides this users must have the opportunity to give feedback, which is taken into account in the development process.

3.2 Roles

In VoCol the developers are subdivided into the following four different roles:

- **Domain Expert**

These have very precise detailed knowledge about the domain to be represented in the vocabulary. However, they have little or no experience in modelling or formal knowledge representation.

- **Knowledge Engineer**

These experts in modelling and knowledge representation ensure the quality of the formal aspects of vocabularies. They are responsible for application of the currently applicable best practices.

- **Application Developer**

Applications for end-users with fully developed user interfaces which build upon and use the vocabularies are produced by the Application Developers.

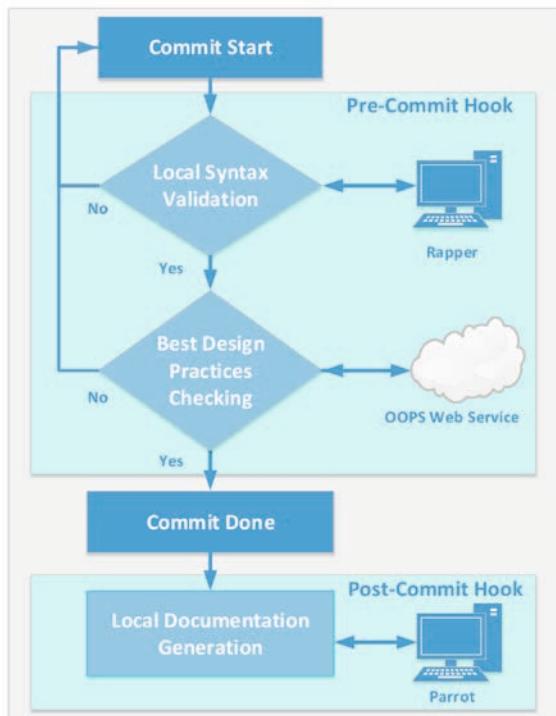
- **Scientist**

The Scientists support the development process of the vocabularies from a scientific point of view with the latest insights from research.

3.3 Git4Voc – Versioning of vocabularies with Git

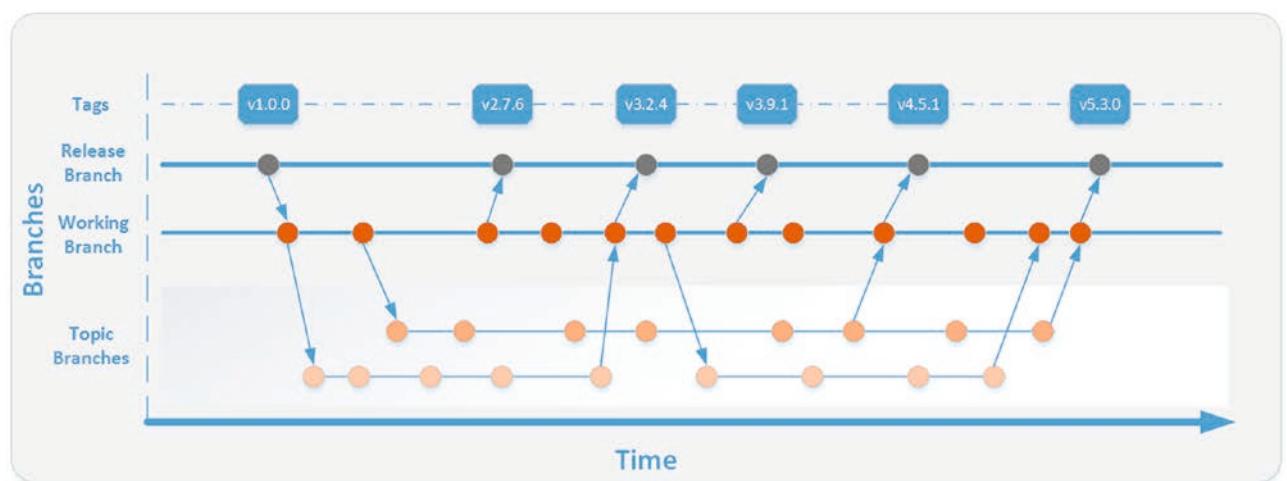
As already mentioned at the outset, the Git distributed versioning system is at the heart of the VoCol development environment. The reason for this is that all developer activities converge here and so the versioning system has access to all the necessary information. The versioning system knows precisely who has carried out what change and at what point in time, and can initiate additional required services such as validation and generation of sets of documentation.

Initiation of external services by means of the Hooks mechanism of Git



Source: ZVEI SG Modelle und Standards

Branches, Tags, Releases and Merges with Git

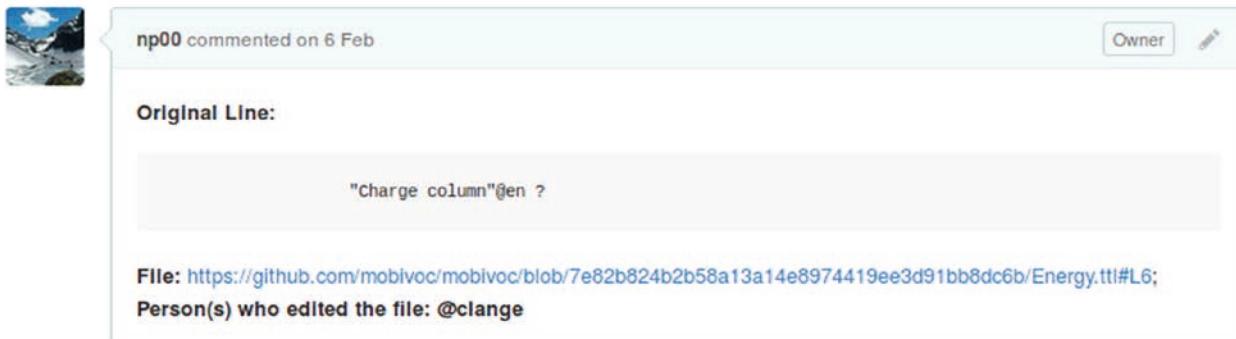


Source: ZVEI SG Modelle und Standards

Description of an issue in Issue Tracker

[Energy.ttl] syntax error, unexpected string literal #31

 np00 opened this issue on 6 Feb · 0 comments



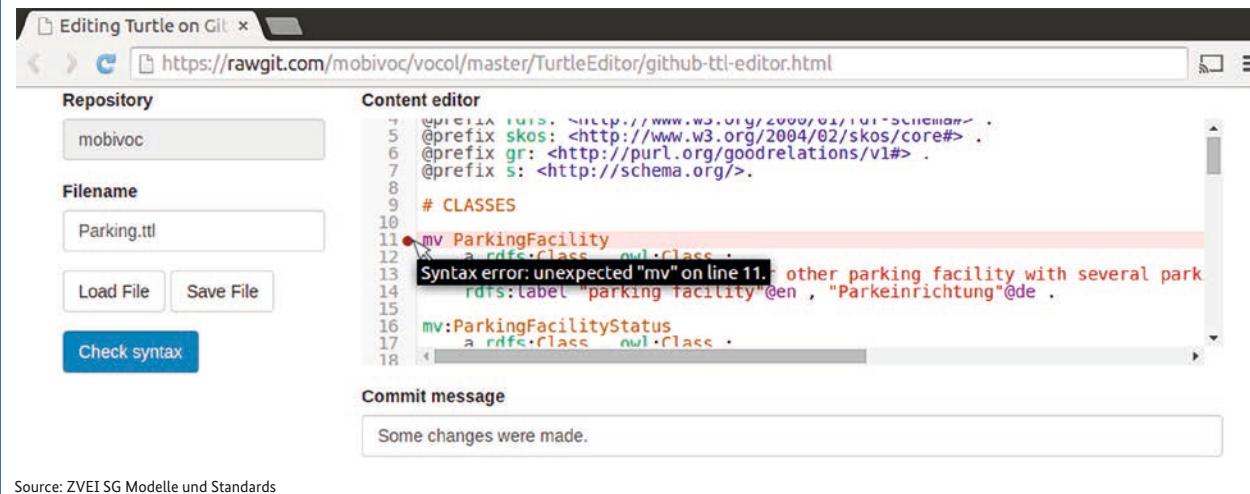
The screenshot shows a GitHub issue page for a file named 'Energy.ttl'. The issue is titled '[Energy.ttl] syntax error, unexpected string literal #31'. It was opened by 'np00' on February 6th and has 0 comments. The original line of code is shown as:

```
"Charge column"@en ?
```

Below the code, it says the file is at <https://github.com/mobivoc/mobivoc/blob/7e82b824b2b58a13a14e8974419ee3d91bb8dc6b/Energy.ttl#L6> and that person(s) who edited the file are '@clang'. There is also an 'Owner' button and a pencil icon.

Source: ZVEI SG Modelle und Standards

Visualisation of the TTL Web Editor



The screenshot shows a web-based TTL editor interface. The URL is <https://rawgit.com/mobivoc/vocol/master/TurtleEditor/github-ttl-editor.html>. On the left, there's a 'Repository' section with 'mobivoc' selected and a 'Filename' section with 'Parking.ttl'. Below that are 'Load File' and 'Save File' buttons, and a 'Check syntax' button. The main area is a 'Content editor' showing the following TTL code:

```

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix gr: <http://purl.org/goodrelations/v1#> .
@prefix s: <http://schema.org/> .

# CLASSES
mv: ParkingFacility
  a rdfs:Class , owl:Class .
  Syntax error: unexpected "mv" on line 11. other parking facility with several park
  rdfs:label "parking facility"@en , "Parkeinrichtung"@de .

mv: ParkingFacilityStatus
  a rdfs:Class , owl:Class .

```

A red box highlights the line 'mv: ParkingFacility' with the error message 'Syntax error: unexpected "mv" on line 11. other parking facility with several park'. At the bottom, there's a 'Commit message' field containing 'Some changes were made.'

Source: ZVEI SG Modelle und Standards

At the same time the various useful features such as user administration and the creation of Branches, Tags, Releases and Merges should be extensively and appropriately used. Chart 3 below shows how development between different development branches could take place.

To initiate additional services the hooks mechanism of Git is used. Chart 4 below illustrates how, following a Commit, external services such as validation of syntax and quality control in respect of the application of best practices can be started.

Besides this additional functionalities of common implementations of Git such as Issue Tracker (Fig. 5) and Web Editors (Fig. 6) can be used with syntax checking functionalities.

