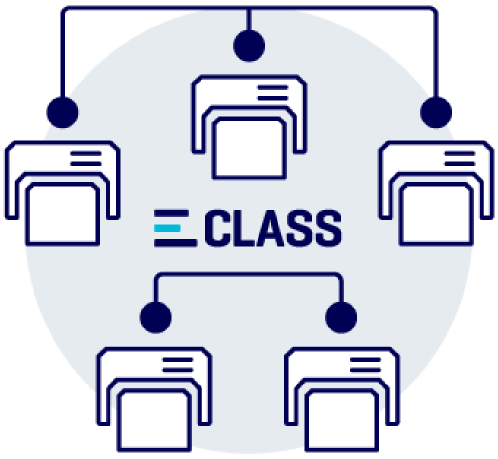


Whitepaper

Modelling the Semantics of Data of an Asset Administration Shell with Elements of ECLASS

Date: 2021-06-29

Authors and contributors

Alexander Belyaev, Ifak Magdeburg / Otto-von-Guericke-Universität Magdeburg

Dr. Christian Block, ECLASS Head Office

Dr. Birgit Boss, Robert Bosch GmbH

Prof. Dr. Christian Diedrich, Ifak Magdeburg / Otto-von-Guericke-Universität Magdeburg

Philippe Juhel, Schneider Electric SE

Dr. Wilfried Hartmann, BASF AG

Oliver Hillermeier, SAP SE

Nikolaus Ondracek, Paradine GmbH

Stephanie Pfeifer, ECLASS Head Office

Frank Scherenschlich, Class.Ing

Josef Schmelter, Phoenix Contact GmbH & Co. KG

Table of Contents

List of Figures

List of Tables

Abbreviations

AAS	Asset Administration Shell
AC	Application Class
AML	Automation Markup Language
API	Application Programming Interface
CDD	Common Data Dictionary
DLT	Distributed Ledger Technology
I4.0	Industrie 4.0
ICD	International Code Designator
IRDI	International Registration Data Identifier
JSON	JavaScript Object Notation
OPC-UA	Open Platform Communications United Architecture
RDF	Resource Description Framework
REST	Representational State Transfer
SI	Système international d’unités (International unit system)
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XSD	XML Schema Definition

1 Introduction & Scope

Plattform Industrie 4.0 defines the Asset Administration Shell (AAS) information model, which is described in the document series “Asset Administration Shell in Detail” [1, 2, 3]. These documents are available in the library of Plattform Industrie 4.0.

The aim of the Asset Administration Shells is to enable partners in value creation networks to exchange meaningful information by conforming to a specified set of standardized information elements. Furthermore, Asset Administration Shells will be capable to interact with each other directly without a leading system and connecting them to a cloud infrastructure in a standardized way.

Plattform Industrie 4.0 recommends ECLASS as the preferred dictionary of standardized semantic to create meaningful Asset Administration Shells or Digital Twins. The AAS has standardized the structure and ECLASS has standardized the semantics for information elements, which should be used for the definition of digital twins. The structure of the AAS and the ECLASS dictionary complement each other in a perfect manner to achieve this objective.

The objective of this whitepaper is to work out which structures of ECLASS can be used to define the semantics of structures of the AAS. This whitepaper therefore explores element by element how each element of the AAS can be semantically described within the means of the ECLASS Conceptual Data Model.

With the shift from implicit to explicit semantics in Industrie 4.0 as one of the basic assumptions the importance of standardized dictionary has increased drastically.

ECLASS is constantly updated and improved. To actively support and contribute to the development of Industrie 4.0, missing structures in ECLASS´ Conceptual Data Model necessary to meet AAS requirements are identified and described in this document. Proposals for the further development of ECLASS’ Conceptual Data Model are made. *Proposals for extensions of the ECLASS Conceptual Data Model to support the Asset Administration Shell are written in cursive letters*.

ECLASS e.V. is willing to further develop the ECLASS Conceptual Data Model according to the requirements that are needed to define meaningful digital twins. With the release of this whitepaper, concrete content for the definition of digital twins can be transferred to ECLASS e.V.

For further information use the homepage of ECLASS or contact the Head Office of ECLASS e.V..

2 Shift from implicit to explicit Semantics

In today's common industrial practice, machine-to-machine communication is designed in such a manner that the communication partners exchange characters e.g., bit patterns, between each other. The use of the exchanged bit patterns with the correct meaning is ensured by the fact that the developers of machine software have the same understanding of exchanged characters between sender and the receiver of data.

This type of exchange is called information exchange with "implicit semantics". The meaning of the exchanged information is known to both communication partners (see Figure 1).

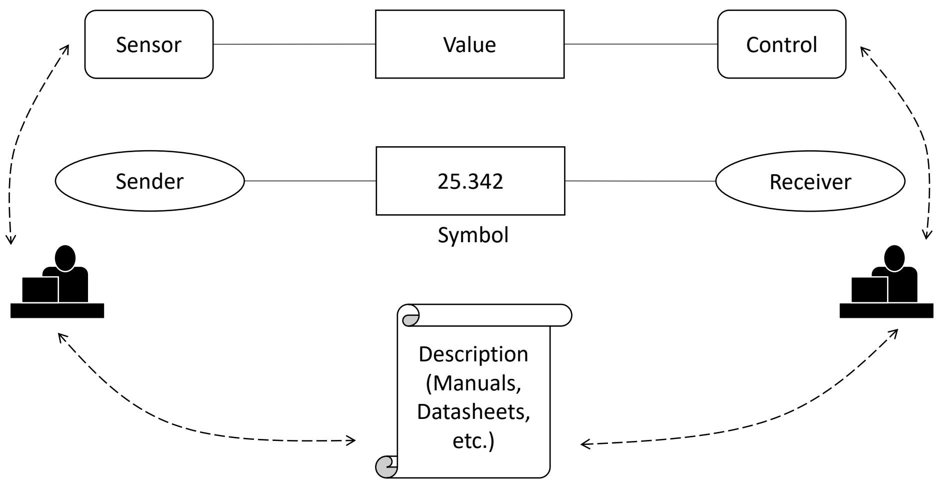


Figure 1: Information Exchange with implicit Semantics [3, 4]

If only an identifier of a property and its value, with a reference to the semantic definition of this property, are exchanged between the interacting partners, a context for the transferred value is thus provided. Information exchange of this kind is called Information exchange with explicit semantics (see Figure 2).

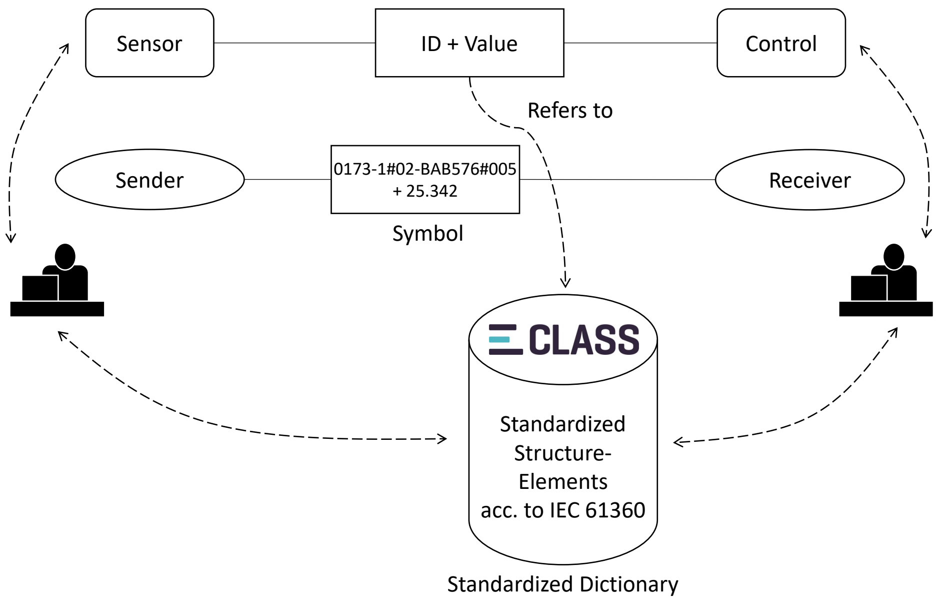


Figure 2: Information Exchange with explicit Semantics [3, 4]

Information exchange via identifier and value is possible if the interaction partners use the same semantic model of a characteristic. A unique identifier refers to a Concept Description (e.g., Structure Element in ECLASS; for specific identifier see 3.1.2). This Concept Description for a characteristic may include the name, definition, SI unit, value format, data type and other attributes that are unchangeable throughout the life of a characteristic. It serves to describe the actual meaning of a characteristic.

A Concept Description therefore provides the minimal context necessary for understanding the meaning of a value.

However, there are attributes, such as the value or the unit of measurement, which can be different for each individual use of the characteristic (i.e., for each instance of a characteristic). In addition, other attributes can also be added for individual applications, such as the time stamp, a statement about the validity of the value (a so-called status), statement logic (smaller, equal, greater), attribute statement (assurance, request, measured value, set value, estimated value, calculated value).

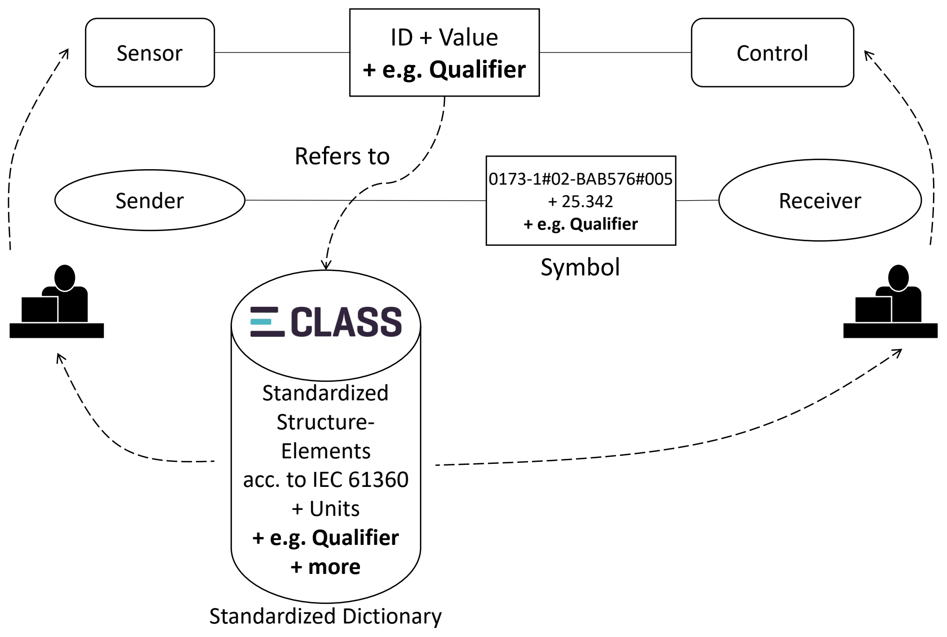


Figure 3: Transfer of the additional Context by Type and Instance related Attributes of a Characteristic [3, 4]

This points out that additional attributes may be transmitted in the information exchange in combination with a semantic identifier (which refers to a Concept Description) and the value. This enriches the exchanged data with additional context or meaning with e.g., Qualifiers (see Figure 3). The specification of this information which refers to the current aspect of the characteristic, relating to a point in time, is necessary to understand the meaning of the transferable value. It follows, however, that these instance-related attributes must also be available in a standardized form.

3 Introduction to ECLASS

The objective of ECLASS is to simplify the electronic, cross-industry data exchange through the classification of standardized product descriptions. Today, about 150 companies from almost every industry, organizations, and public institutions, are members of the ECLASS association. The central unique characteristic of ECLASS is the possibility of providing an unambiguous, language-neutral, machine-readable, and industry-independent description of products and services and their characteristics. Currently, there are about 45,000 product classes, that are organized in four levels. On the fourth level a set of properties is assigned. The set of properties are generated as a set of the 19,000 well-defined properties of the ECLASS Dictionary. A large part of the goods and services traded worldwide is represented in ECLASS. Meanwhile, ECLASS is used in thousands of companies nationally and internationally. [5]

The original application domain of ECLASS Property descriptions, was the classification of products and components for purchasing and sales. New areas of application are evolving, like engineering and smart manufacturing. For the digitalization of the industry the semantic standardization plays a fundamental role. ECLASS is the semantic dictionary with the broadest content, worldwide.

In the following subchapters main concepts, and elements of the ECLASS Standard are introduced. The detailed description of the ECLASS Conceptual Data Model can be found in [6, 7].

One of the most significant components of ECLASS is the Classification of products and services. ECLASS consists of a four-level hierarchy of Classification Classes (i.e., a tree structure), the first level being the most general, the fourth level being the most specific. This allows the construction of a hierarchy of technical, commercial, or other considerations, regardless of "mathematical" dependencies. From top to bottom these hierarchical levels are called:

- Segment
- Main Group
- Group
- Sub-Group or Commodity Class

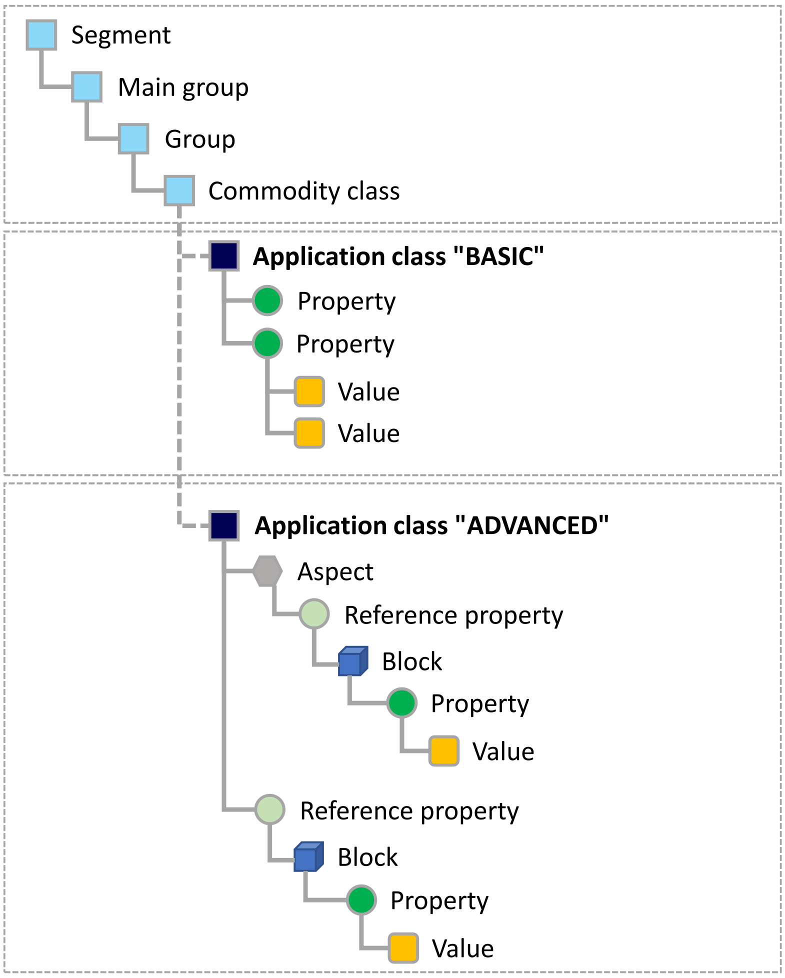
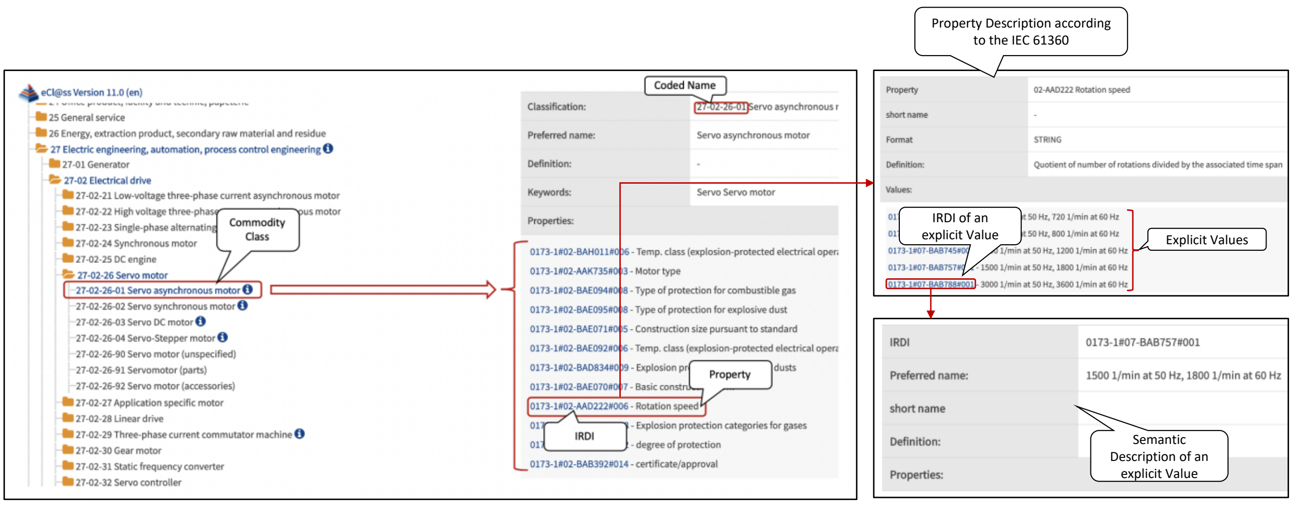


Figure 4: ECLASS Classification and Hierarchy Levels

The hierarchy of the Classification Classes is represented with the help of the coded name or rather Class Code. The coded name consists of an 8-digit integer number, two digits for each hierarchical level. The number of trailing zeros in the end indicates the level of hierarchy, e.g., 16-00-00-00 (Segment "Food, beverage, tobacco"), 16-04-00-00 (Main group "Fruit"), 16-04-03-00 (Group "Berry fruit"), 16-04-03-01 (Commodity Class "Blackberry"). In this way every product or Classification Class can be identified with a unique identifier (Class Code). The Commodity Class or Product Group (or Application Class) is then further described with the help of properties and property values in the fourth level. Properties and values form the basis for the product description. To familiarize yourself with the ECLASS Standard, visit: https://www.eclass.eu/en/standard/search-in-eclass.html


Coded Name
Commodity Class
Property
IRDI
Property Description according to the IEC 61360
IRDI of an explicit Value
Explicit Values
Semantic Description of an explicit Value

eCl@ss Version 11.0 (en)
- 25 General service
- 26 Energy, extraction product, secondary raw material and residue
- 27 Electric engineering, automation, process control engineering
  - 27-01 Generator
  - 27-02 Electrical drive
    - 27-02-21 Low-voltage three-phase current asynchronous motor
    - 27-02-22 High voltage three-phase ous motor
    - 27-02-23 Single-phase alternating
    - 27-02-24 Synchronous motor
    - 27-02-25 DC engine
    - 27-02-26 Servo motor
      - 27-02-26-01 Servo asynchronous motor
      - 27-02-26-02 Servo synchronous motor
      - 27-02-26-03 Servo DC motor
      - 27-02-26-04 Servo-Stepper motor
      - 27-02-26-90 Servo motor (unspecified)
      - 27-02-26-91 Servomotor (parts)
      - 27-02-26-92 Servo motor (accessories)
    - 27-02-27 Application specific motor
    - 27-02-28 Linear drive
    - 27-02-29 Three-phase current commutator machine
    - 27-02-30 Gear motor
    - 27-02-31 Static frequency converter
    - 27-02-32 Servo controller

Classification:	27-02-26-01 Servo asynchronous r
Preferred name:	Servo asynchronous motor
Definition:	-
Keywords:	Servo Servo motor
Properties:	

0173-1#02-BAH011#006 - Temp. class (explosion-protected electrical opera
0173-1#02-AAK735#003 - Motor type
0173-1#02-BAE094#008 - Type of protection for combustible gas
0173-1#02-BAE095#008 - Type of protection for explosive dust
0173-1#02-BAE071#005 - Construction size pursuant to standard
0173-1#02-BAE092#006 - Temp. class (explosion-protected electrical opera
0173-1#02-BAD834#009 - Explosion pr dusts
0173-1#02-BAE070#007 - Basic constru
0173-1#02-AAD222#006 - Rotation speed
017 - Explosion protection categories for gases
017 - degree of protection
0173-1#02-BAB392#014 - certificate/approval

Property	02-AAD222 Rotation speed
short name	-
Format	STRING
Definition:	Quotient of number of rotations divided by the associated time span
Values:	

01 t 50 Hz, 720 1/min at 60 Hz
01 t 50 Hz, 800 1/min at 60 Hz
0173-1#07-BAB745#00 0 1/min at 50 Hz, 1200 1/min at 60 Hz
0173-1#07-BAB757# - 1500 1/min at 50 Hz, 1800 1/min at 60 Hz
0173-1#07-BAB788#001 - 3000 1/min at 50 Hz, 3600 1/min at 60 Hz

IRDI	0173-1#07-BAB757#001
Preferred name:	1500 1/min at 50 Hz, 1800 1/min at 60 Hz
short name	
Definition:	
Properties:	

Figure 5: Description of an Application Class with Properties

3.1 ECLASS Conceptual Data Model

In this chapter you can get a first overview of the Conceptual Data Model of the ECLASS Dictionary (see Figure 6). For the complete description please use the official ECLASS documentation [6, 7], of which this is an extract.

3.1.1 Overview

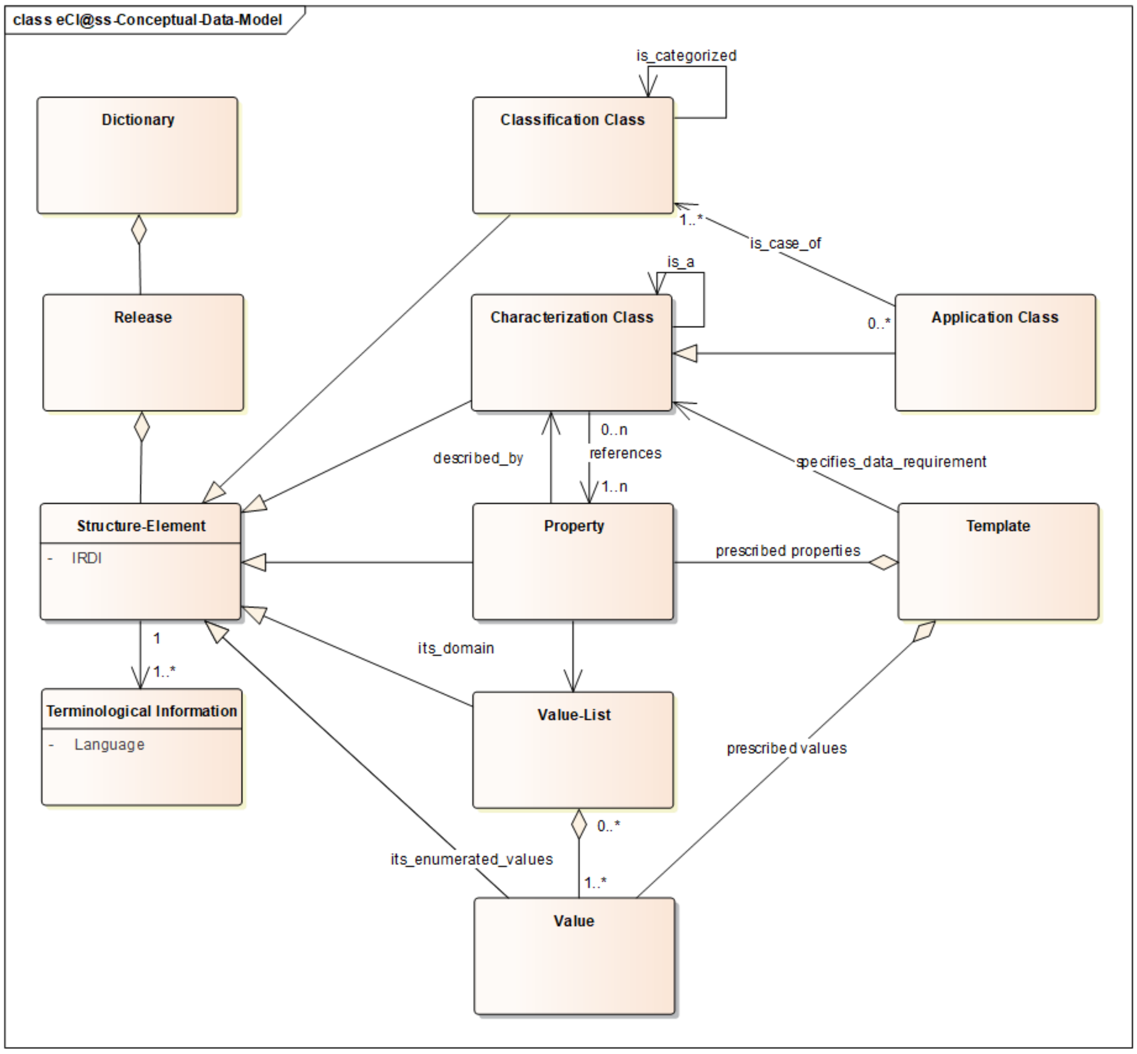


Figure 6: ECLASS Conceptional Data Model

ECLASS is a formal semantic dictionary which is used for product description and product classification.

If a given concept represents a class of products which may be described by the same set of properties, this class of products is a Characterization Class, which is identified as an Application Class and categorized in a Classification Class.

The enumeration of all values assigned to a property is the Structure Element Value List.

A Characterization Class may have zero to many templates, which are defining the formal data requirement specification (according to ISO 22745-30) in this class.

3.1.2 IRDI

ECLASS uses globally unique identifiers for every Structure Element included in the ECLASS Standard. This globally unique identifier is called IRDI (International Registration Data Identifier). These identifiers are used to ensure that the semantic of an element is unique in the overall system.

The IRDI is based on the international standards ISO/IEC 11179-6, ISO 29002, and ISO 6532. Every institution registered by the registration authority has an unique ICD (International Code Designator) identifier. In the case of ECLASS this is the "0173". ECLASS Dictionary provides IRDIs for all Structure Elements e.g., Classification Classes, Application Classes, Properties, Units of Measure, Property Values (in case of coded values), Value Lists, Aspects, Blocks and Templates (see also [8]). For example, the IRDI 0173-1#02-AAO677#002 stands for a property defining the “Manufacturer name”.

For further information please see chapter 3.3.

3.1.3 Structure Element

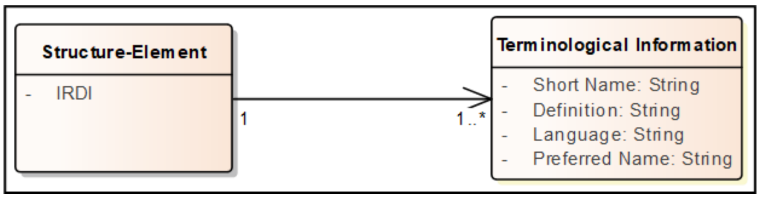


Figure 7: Structure Element

A Structure Element poses a semantic representation of a real-world concept in the ECLASS Dictionary. The Structure Element is uniquely identified by an IRDI. For details see the following chapter.

For human understanding each Structure Element is described by translatable Terminological Information.

3.1.4 Release and Representations

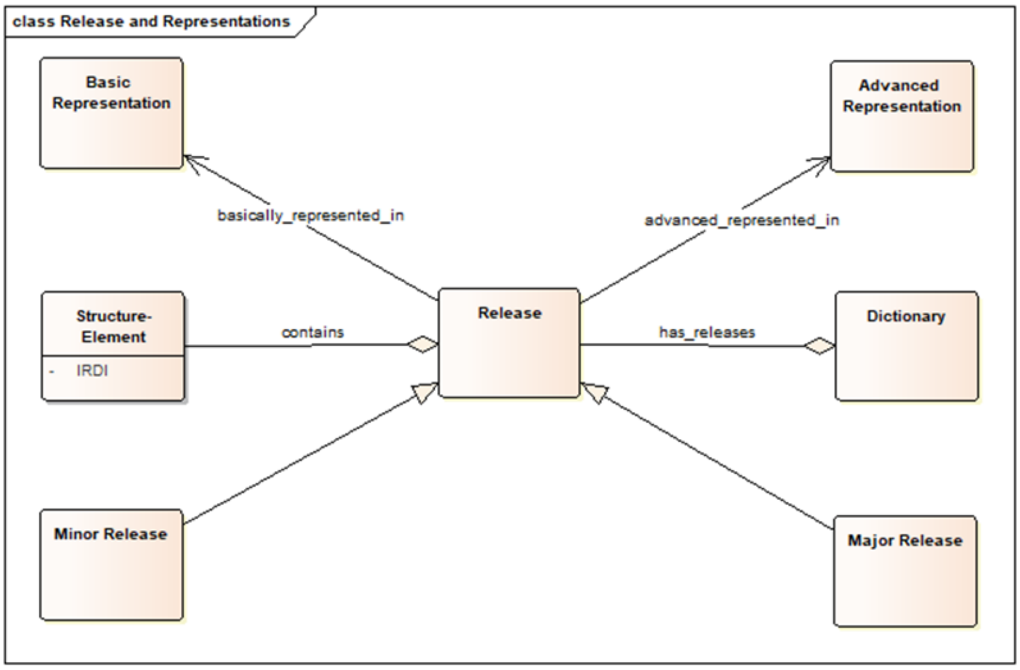


Figure 8: Release and Representation

ECLASS is a formal semantic dictionary which is used for product description and product classification.

The ECLASS Dictionary is maintained in Releases. Each Release contains an extensive and reproduceable set of Structure Elements.

ECLASS distinguishes between Minor-Release and Major-Release, depending on business rules of the releases' content.

Releases are delivered in two different representations, Basic Representation and Advanced Representation.

3.2 Mappings

In addition to the definition of a product's classification and description, as well as exchange as a semantic dictionary, the ECLASS Standard integrates semantic mappings. Semantic mappings are created between dictionaries or dictionary releases.

The semantic mappings within the ECLASS Standard are divided into:

- ECLASS Releases (e.g., Release 11.1 to 12)

- ECLASS Representations (Basic to Advanced)
- ECLASS and external standards (e.g., ECLASS to ETIM, or ECLASS to IEC CDD),

These Semantic Mappings support the automatic conversion (“transformation”) of product descriptions and classifications between ECLASS Releases, ECLASS Representations and ECLASS and external standards.

3.2.1 Semantic Mapping between ECLASS Releases

During ECLASS Release generation, Semantic Mappings are created which may be used in the ECLASS Release Update Processes (see [9]) to simplify and ease transformation of product data from an ECLASS Major-Release x to the subsequent Release x+1.

3.2.2 Semantic Mapping between ECLASS Representations

After ECLASS Release Generation, Semantic Mappings are created. For instance, the path of the [BASIC AC ID] - [Property1 ID] is mapped to the path of the [ADVANCED AC ID] - ([Aspect1 ID]) - ([Block1 ID]) - [BlockN ID] - [Property 1 ID]. For more information see [10].

These mappings are used to support the transformation of a Basic Encoding into an Advanced Encoding within one Release of ECLASS.

3.2.3 Semantic Mapping between ECLASS and external Standards

The mapping between ECLASS and external standards includes cross standard mapping e.g., for a mapping from ECLASS to ETIM.

These mappings are used to support the transformation of a product description, based on ECLASS, into the encoding of the external standard.

3.2.4 Serialization of Semantic Mappings

The serialization of Semantic Mappings between product descriptions is called Transaction Update File (see [11]).

The serialization of mappings between product classifications is called Classification Update File (see [12]).

Both mappings follow the schema in [13].

3.2.5 Interpretation of Semantic Mappings (Value Transformation)

Semantic Mappings are the basis for transforming a given Value Encoding (e.g., based on ECLASS 8.1) to a changed Value Encoding (e.g., based on ECLASS 9).

Mappings may be interpreted to transform Value Encodings based on the semantic of the Source Dictionary to the Destination Dictionary Encoding. The reference algorithm [14] is an example of such a transformation.

ECLASS enables the interpretation of mapping plans to be implemented within target systems.

3.2.6 Data model of Semantic Mappings

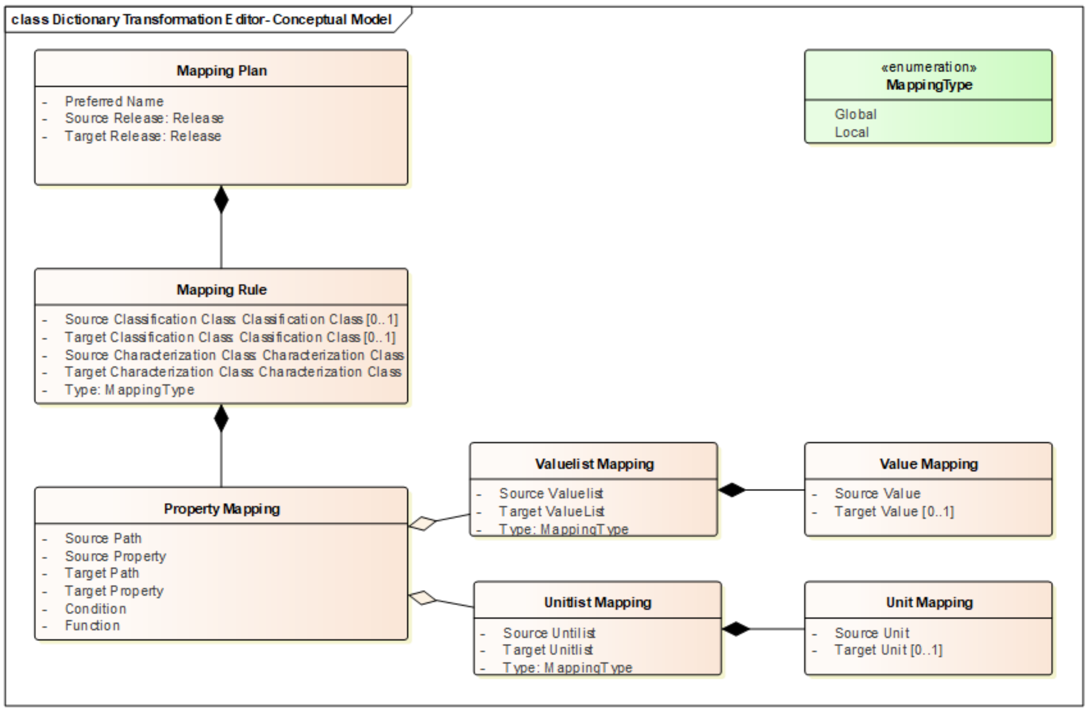


Figure 9: Data Model of Semantic Mappings

Semantic Mappings are collected in so called Mapping Plans which map a release (of a dictionary) with another release (of a dictionary).

The Mapping Plan consists of a set of Mapping Rules between Characterization Classes with other Characterization Classes. Mapping Rules define a path map from the source path to the destination path. A Mapping Rule is established per pair of Characterization Classes, and creates a set of Property Mappings, which map the structure (Properties, Values, Units) in context of these Characterization Classes.

Property mappings may be conditional, e.g., during value transformation the mapping applies only if a given condition applies (mostly Property Value).

Property Mappings may convey a mathematical function which is used to transform values during interpretation.

Depending on the datatype, the Property Mapping may consist of a set of Value List Mappings and/or a set of Unit List Mappings. Value List and Unit List Mappings may be global, this means the mapping applies irrespective of contexts (e.g., Characterization Classes) for all occurrences.

3.3 ECLASS Context

The chapters above describe ECLASS and its included features in general. It is important to understand basic constructs. However, it is important to understand how ECLASS is used in the final application as well as how semantic uniqueness of a property is given in the context of an application based on ECLASS.

In general, in ECLASS a property can occur only once in a Characterization Class (Application Class, Block, Aspect). However, a property in ECLASS can be used in different contexts. That means, that a property can be reused inside several Characterization Classes like blocks, which belong to one Application Class for instance. Here the Application Class would define a root context. Additionally, each block defines a further sub-context for the property and extends the meaning of the property. Two basic approaches can be distinguished:

3.3.1 Usage of Structure Elements without Context

Although the ECLASS Conceptual Data Model includes the use of properties with context reference, the usage of structure elements without context as a standalone element is possible. This allows to have a characteristic “current” to be identified as a current together with a unit (e.g., Ampere), etc. in an asset description. Although for example, it cannot be determined whether it is the input or output current. This could be solved by defining two properties “current”, one for input and one for output.

3.3.2 Usage of Structure Elements in Context according to the ECLASS Structure

In addition to the definition of specific properties like in the paragraph above a semantic specialization can also be included in the context of the property. This is possible for example by using the property “current” in a block that describes the output. The context of the block “Output” would make clear, that output current is meant. This example makes clear that the context provides important information, that must not get lost, when using Structure Elements of ECLASS for the definition of Digital Twins. Even more complex would be, using a generic connector block that only becomes an output type through another property in the context of a polymorphism. Additionally, if there are multiple outputs, a machine cannot semantically determine which port and current it is. This is made possible with the modeling feature of cardinality in the ECLASS Standard.

Note: The following paragraphs describe the application of ECLASS inside of the AAS. Please see chapter 4 for an AAS introduction.

To not lose the context information of the used ECLASS Structure Elements in AAS, the AAS Elements should correspond to a specific path inside the ECLASS Structure, where the parent element defines the context of the nested elements.

For instance, and irrespective of the defined way to define semantics for a specific AAS element defined in this document (chapter 5), an ECLASS Characterization Class could be used as semantic of a Submodel. In addition, the ECLASS Properties describing this Characterization Class could be nested Submodel Elements. Then the semanticId of the Submodel points to the IRDI of the Characterization Class and the semanticIds of the Submodel Elements to the IRDIs of the ECLASS Properties. However, it becomes more complex with ECLASS Advanced Modelling Concepts like Aspect, Block, Cardinality and Polymorphism.

Here the authors propose to take over the IRDI-Path idea, which is described in [15], but use it as a relative path of the current context in the AAS and not as an absolute path. Through this the AAS Element would correspond to a specific path and get a clear semantic as well as semantic uniqueness. This allows Submodel Elements, which have the same IRDI as key in semanticId to be semantically distinguished from each other.

Note: It should be noted here, that an ECLASS Characterization Class like a block is described by properties only. To link a block to a Characterization Class or to embed a block in a block for instance, the block is linked via a Reference Property in ECLASS. Therefore, two structure elements with two unique IRDIs are inside the ECLASS Dictionary. However, only the IRDI of the Reference Property is used since this Property points to the specific Block again. During the interpretation, the knowledge of the Reference Property is used in the AAS. Therefore, a unique path would be like AC-IRDI/Ref-IRDI/Ref-IRDI/Property-IRDI and NOT AC-IRDI/Ref-IRDI/Block-IRDI/Ref-IRDI/Block-IRDI/Property-IRDI.

4 Introduction to the Asset Administration Shell

In tomorrow's industrial world, components, devices, or applications will be able to communicate and interact with each other and work together seamlessly and interoperable - across company, industry, and national boundaries without a leading system. Plattform Industrie 4.0 offers concrete technical concept and specifications to achieve this vision. The central concept for semantic interoperability is the Asset Administration Shell (AAS).

4.1 Asset Administration Shell

The AAS is the implementation of the "digital twin" for Industrie 4.0 and enables cross-vendor interoperability and interaction. In the Industrie 4.0 world, an asset is digitally represented by an AAS. It contains a description of all information and functionalities that are necessary to realize new digital business models and use cases that the AAS is to support.

The main functional elements of the AAS are the Submodels. Submodels are the standardized sets of standardized elements representing and describing characteristics, configuration parameters, states, capabilities, and services offered for the represented asset in a machine-interpretable form. They are the basis for an interoperable, cross-company information exchange. The following types of information can be represented in Submodels [1]:

- Product characteristics: These are often static characteristics of asset or product types, such as catalogue data. The static data of product instances can also be defined (for example, product serial number).
- Process variables and parameters, telemetry data: These are dynamic values for one specific instance of an asset type.
- References to external data sources or files: They enable the integration of information stored in external systems. Blobs can be stored directly within the Submodel.
- References and relationships within the same or other AAS or their parts (for instance to Submodels or single Submodel Elements), also with other value-added partners: Among other things, this can be used to build up composite components, as described in the publication "Relationships between I4.0 components - composite components and intelligent production" [16].

- Technical capabilities of the I4.0 components and description of related operations: This allows the capabilities of the asset and the parameterization to be specified and the associated methods to be executed.
- Collections of characteristics, such as lists or tables: Collections allow you to implement complex characteristic structures and are used for structuring within Submodels.

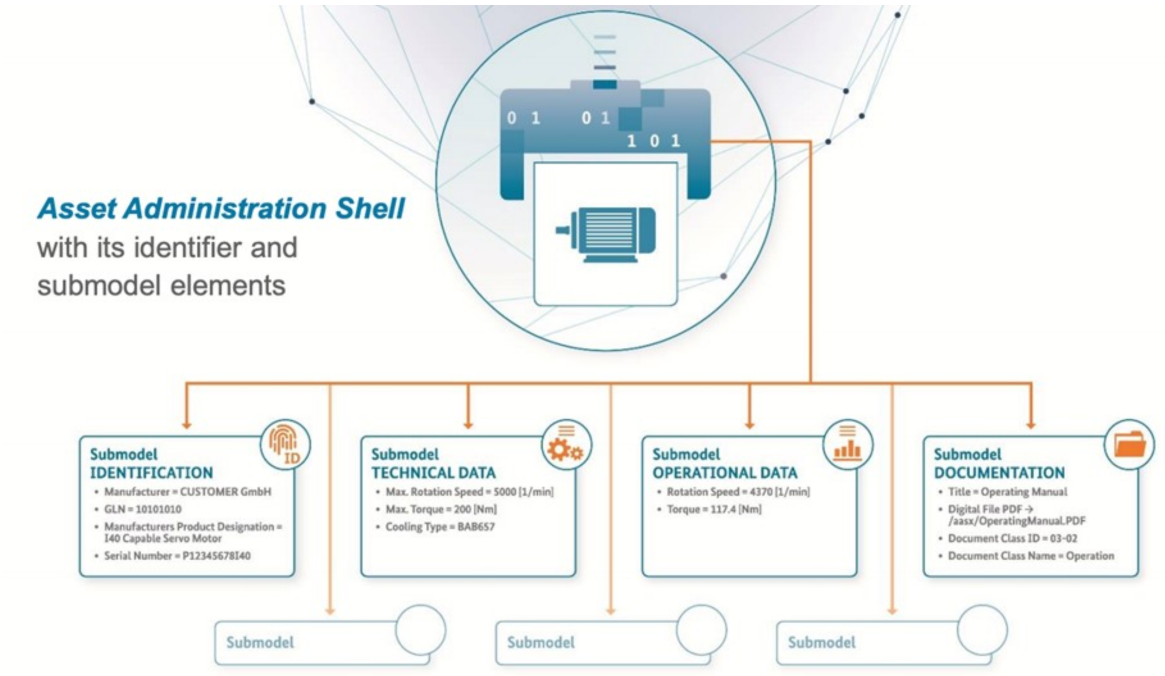

Asset Administration Shell
with its identifier and
submodel elements
0 1 0 1
1 0 1
Submodel
IDENTIFICATION
• Manufacturer = CUSTOMER GmbH
• GLN = 10101010
• Manufacturers Product Designation = I40 Capable Servo Motor
• Serial Number = P12345678I40
Submodel
TECHNICAL DATA
• Max. Rotation Speed = 5000 [1/min]
• Max. Torque = 200 [Nm]
• Cooling Type = BAB657
Submodel
OPERATIONAL DATA
• Rotation Speed = 4370 [1/min]
• Torque = 117.4 [Nm]
Submodel
DOCUMENTATION
• Title = Operating Manual
• Digital File PDF → /aasx/OperatingManual.PDF
• Document Class ID = 03-02
• Document Class Name = Operation
Submodel
Submodel
Submodel

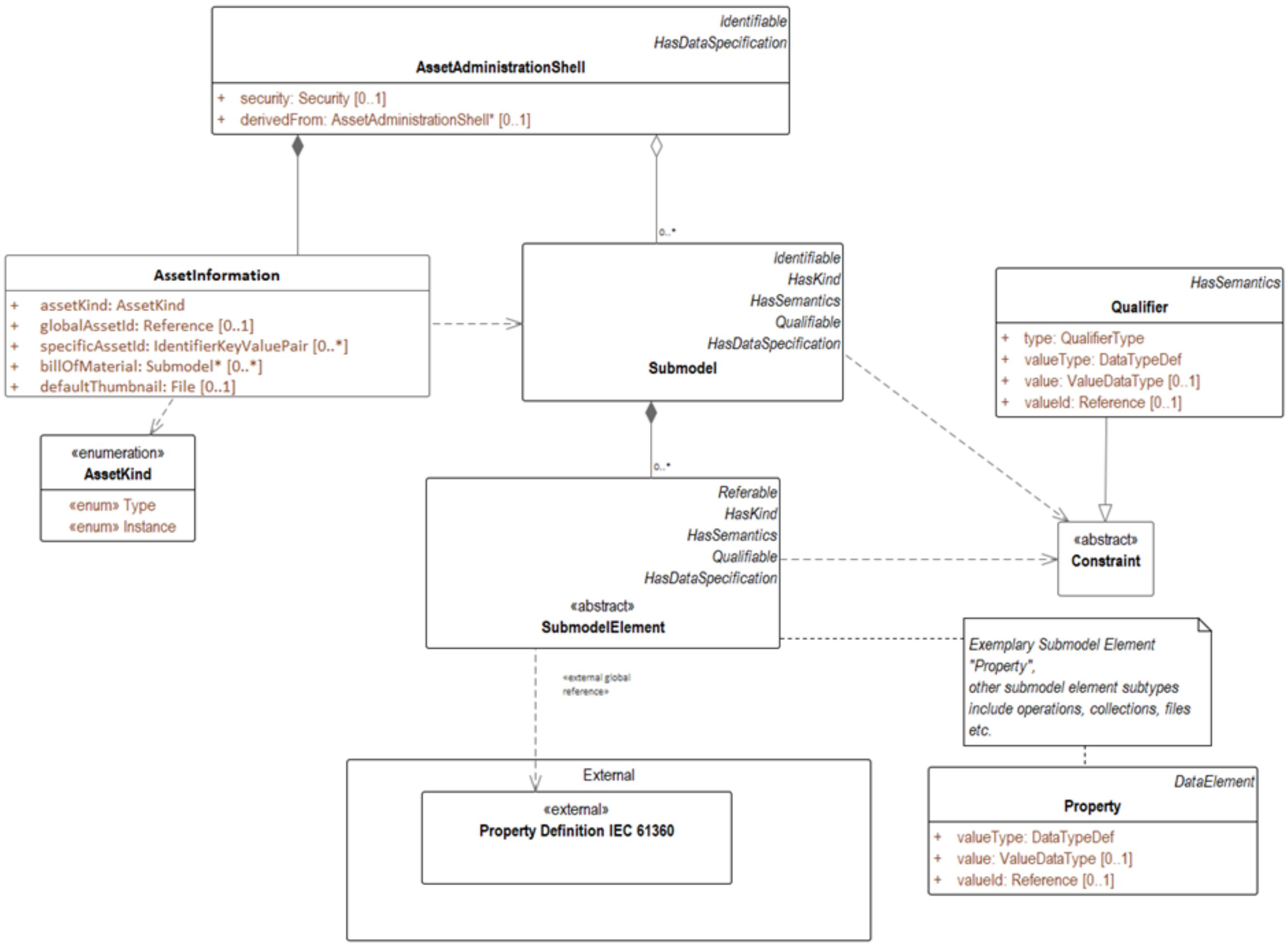
Figure 10: Asset Administration Shell and its Submodels (© Plattform Industrie 4.0)

For all Submodels, Submodel Element Collections and all other Submodel Elements, it is crucial that their semantic description is clearly defined and accessible to the interaction partners. This can be achieved by referencing the Concept Description defining the semantics of a specific Submodel or Submodel Element. These Concept Descriptions are contained in dictionaries. These dictionaries contain standardized data elements that follow semantic standards such as IEC 61360. Plattform Industrie 4.0 recommends ECLASS as the preferred dictionary of semantic definitions for the elements of AAS.

4.2 Meta Model of the Asset Administration Shell

The AAS Meta Model (Figure 11) is represented in the form of a UML class diagram. The boxes are the Classes. The relationships between the Classes are shown as lines with different end symbols. Inherited Classes not contained in the diagram are listed in the top right corner.

The main parts of an AAS are on the one hand the information about the asset it is representing (Class “AssetInformation”) and on the other hand its Submodels (Class “Submodel”).



Identifiable
HasDataSpecification
AssetAdministrationShell
+ security: Security [0..1]
+ derivedFrom: AssetAdministrationShell* [0..1]
0..*
AssetInformation
+ assetKind: AssetKind
+ globalAssetId: Reference [0..1]
+ specificAssetId: IdentifierKeyValuePair [0..*]
+ billOfMaterial: Submodel* [0..*]
+ defaultThumbnail: File [0..1]
Identifiable
HasKind
HasSemantics
Qualifiable
HasDataSpecification
Submodel
HasSemantics
Qualifier
+ type: QualifierType
+ valueType: DataTypeDef
+ value: ValueDataType [0..1]
+ valueId: Reference [0..1]
«enumeration»
AssetKind
«enum» Type
«enum» Instance
0..*
Referable
HasKind
HasSemantics
Qualifiable
HasDataSpecification
«abstract»
SubmodelElement
«abstract»
Constraint
Exemplary Submodel Element "Property", other submodel element subtypes include operations, collections, files etc.
«external global reference»
External
«external»
Property Definition IEC 61360
DataElement
Property
+ valueType: DataTypeDef
+ value: ValueDataType [0..1]
+ valueId: Reference [0..1]

Figure 11: AAS Meta Model in Form of an UML Class Diagram (© Plattform Industrie 4.0)

A Submodel consists of Submodel Elements (Class "SubmodelElement"). A typically Submodel Element is a property. But also, other Submodel Elements are supported like relationships, operations or files. For more details look up the document series “Asset Administration Shell in Detail” [1]. A Submodel or Submodel Element can be qualified (inherited Class “HasQualifier”).

4.3 Passive, Reactive and Proactive Asset Administration Shells

A digital twin can be implemented as a passive, reactive or proactive AAS (see Figure 12).

- **Passive** AAS as described in “Details of the Asset Administration Shell - Part 1” [1] can be serialized in XML, JSON, RDF, AML or OPC-UA Nodeset format and exchanged via a standardized file exchange format (AASX) between partners. They typically contain static product information.

- **Reactive** AAS as described in "Details of the Asset Administration Shell – Part 2" [2] offer access to its data via a standardized interface. APIs in different technologies are planned to be specified. The difference is that the inner structure of AAS (e.g. Submodels) is made available for higher-level enterprise information systems via an interface. The interface design depends on the chosen technology. Reactive AAS can be deployed in different ways, for example they may run on Edge, On-Premise or in a public cloud.
- The concept of **proactive** AAS follows the concepts of agents and is based on a certain autonomy or decision-making ability of the AAS. These serve to design decentralized processes and horizontal peer-to-peer interaction networks. (See also [3, 4])

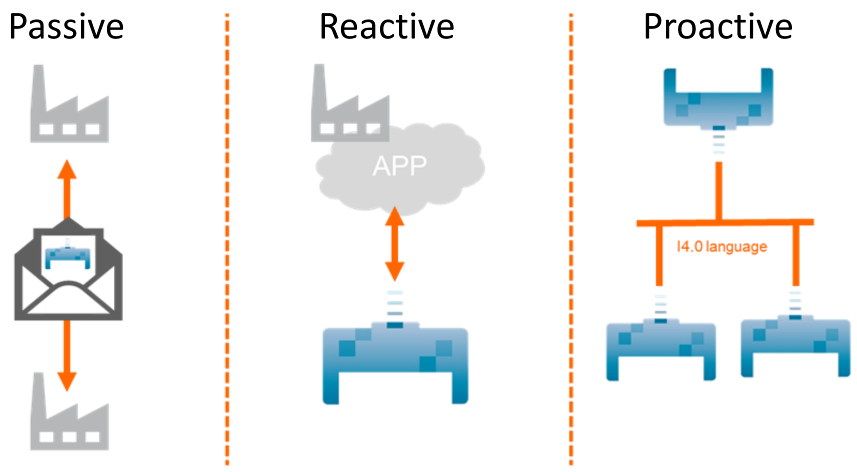
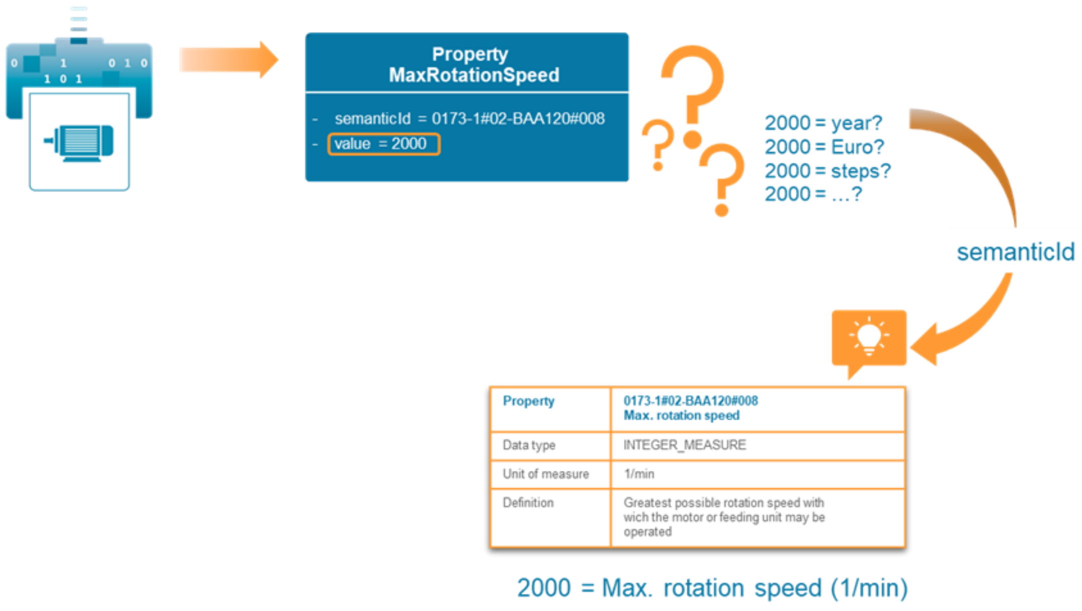


Figure 12: Three Types of AAS using a common AAS Meta Model (© Plattform Industrie 4.0)

4.4 AAS referencing Concept Descriptions

In the context of this document the most important attribute of elements within the meta model of the AAS is the one inherited by the Class "HasSemantics", the "semanticId".

An example is given in Figure 13. An AAS Property "MaxRotationSpeed" consists of a pair "Value" being "2000" and a semanticId. The value of the semanticId attribute should be the IRDI of a Concept Description in ECLASS. The AAS Property without this information of the ECLASS Concept Description does not reveal anything about the meaning or the physical Unit of the value "2000". When looking up the Concept Description in ECLASS the meaning gets clear: It is the greatest possible rotation speed with which the motor or feeding Unit may be operated. It is specified in rotations per minute (1/min). The type is INTEGER_MEASURE.



Property	0173-1#02-BAA120#008 Max. rotation speed
Data type	INTEGER_MEASURE
Unit of measure	1/min
Definition	Greatest possible rotation speed with wich the motor or feeding unit may be operated

Figure 13: Example how to Reference an External Concept Description

The semantics of an AAS Property or other elements can either be defined by referencing an external dictionary entry like ECLASS or by referencing a Concept Description defined in the scope of the AAS. The AAS Concept Description may be proprietary or a copy from a dictionary entry of an external dictionary, like ECLASS.

The description of the concept follows a standardized schema and is realized as a Data Specification Template in the AAS. Currently a predefined data specification template for Concept Descriptions for properties is available supporting IEC 61360.

5 ECLASS Structure Elements for the Elements of the AAS Meta Model

This chapter describes which ECLASS Structure Elements of the ECLASS Conceptual Data Model could be used as Concept Descriptions to define Submodels, different Submodel Elements and other elements of the AAS for which a semanticId can be defined. To have the complete AAS Meta Model with all its Elements on hand please use the document [1].

5.1 Introduction

The essential functional elements of the AAS are Submodels, which comprise descriptive properties, configuration parameters, variables, files, offered capabilities and operations of the asset in a machine-interpretable form. Submodels are information models that describe a specific aspect of an asset. In the best case these Submodels follow a standard. Submodels consist of a structured set of Submodel Elements. Figure 14 provides an overview of the possible Submodel Elements. The following sections explain how the individual Submodel Elements can be represented with Structure Elements of the ECLASS Conceptual Data Model. In addition, Submodels and Submodel Elements can be qualified. For Qualifier it is explained how to define a Structure Element in ECLASS. In some cases, necessary elements of the AAS Meta Model are not yet supported by either IEC 61360 or ECLASS. In these cases, a possible proposal for extending the ECLASS Conceptual Data Model is given. *Proposals are written in cursive letters* and are specially marked in corresponding figures or UML´s.

Each Submodel Element has a semanticId. This semanticId, which is an attribute of the Class "HasSemantics", is inherited by Submodels, Submodel Elements, Qualifiers and Views and is a reference to a unique identifier. The identified entity is the Concept Description. In [1] extensions and external asset IDs were introduced. These elements do also have a semanticId. For simplicity, in this document semanticId and HasSemantics but also semantic reference is used interchangeably.

ECLASS is the recommended standard for this semanticId. This whitepaper is analyzing all Submodel Elements of an AAS and suggests which elements in ECLASS should be referenced. In other words, how the Concept Description of an AAS Element can be defined with elements of ECLASS. In general, the principle followed is to make semantics explicitly by using standardized ECLASS Elements.

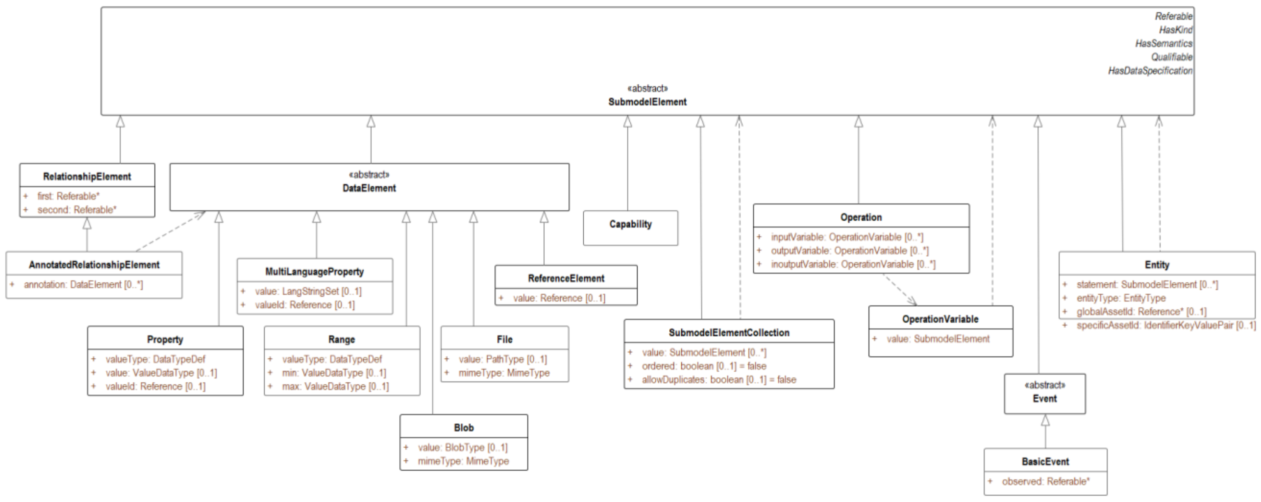

Referable
HasKind
HasSemantics
Qualifiable
HasDataSpecification
«abstract» SubmodelElement
RelationshipElement
+ first: Referable*
+ second: Referable*
«abstract» DataElement
Capability
Operation
+ inputVariable: OperationVariable [0..*]
+ outputVariable: OperationVariable [0..*]
+ inoutputVariable: OperationVariable [0..*]
AnnotatedRelationshipElement
+ annotation: DataElement [0..*]
MultiLanguageProperty
+ value: LangStringSet [0..1]
+ valueId: Reference [0..1]
ReferenceElement
+ value: Reference [0..1]
Entity
+ statement: SubmodelElement [0..*]
+ entityType: EntityType
+ globalAssetId: Reference* [0..1]
+ specificAssetId: IdentifierKeyValuePair [0..1]
OperationVariable
+ value: SubmodelElement
Property
+ valueType: DataTypeDef
+ value: ValueDataType [0..1]
+ valueId: Reference [0..1]
Range
+ valueType: DataTypeDef
+ min: ValueDataType [0..1]
+ max: ValueDataType [0..1]
File
+ value: PathType [0..1]
+ mimeType: MimeType
SubmodelElementCollection
+ value: SubmodelElement [0..*]
+ ordered: boolean [0..1] = false
+ allowDuplicates: boolean [0..1] = false
«abstract» Event
Blob
+ value: BlobType [0..1]
+ mimeType: MimeType
BasicEvent
+ observed: Referable*

Figure 14: AAS Submodel Elements [1]

5.2 AAS Submodel

Although there are several elements in ECLASS like Classification Classes, Application Classes, Aspects or Blocks that have similarities with Submodels, the recommendation is to explicitly support the standardization of Submodels via a separate dictionary in ECLASS. The Application Class is considered to represent a Submodel Template. In case ECLASS Dictionaries (Basic and Advanced) provide properties or other Structure Elements like Blocks, Aspects, etc., that might be used in Submodels, a Semantic Mapping should be defined. In Figure 15 the concept is sketched.

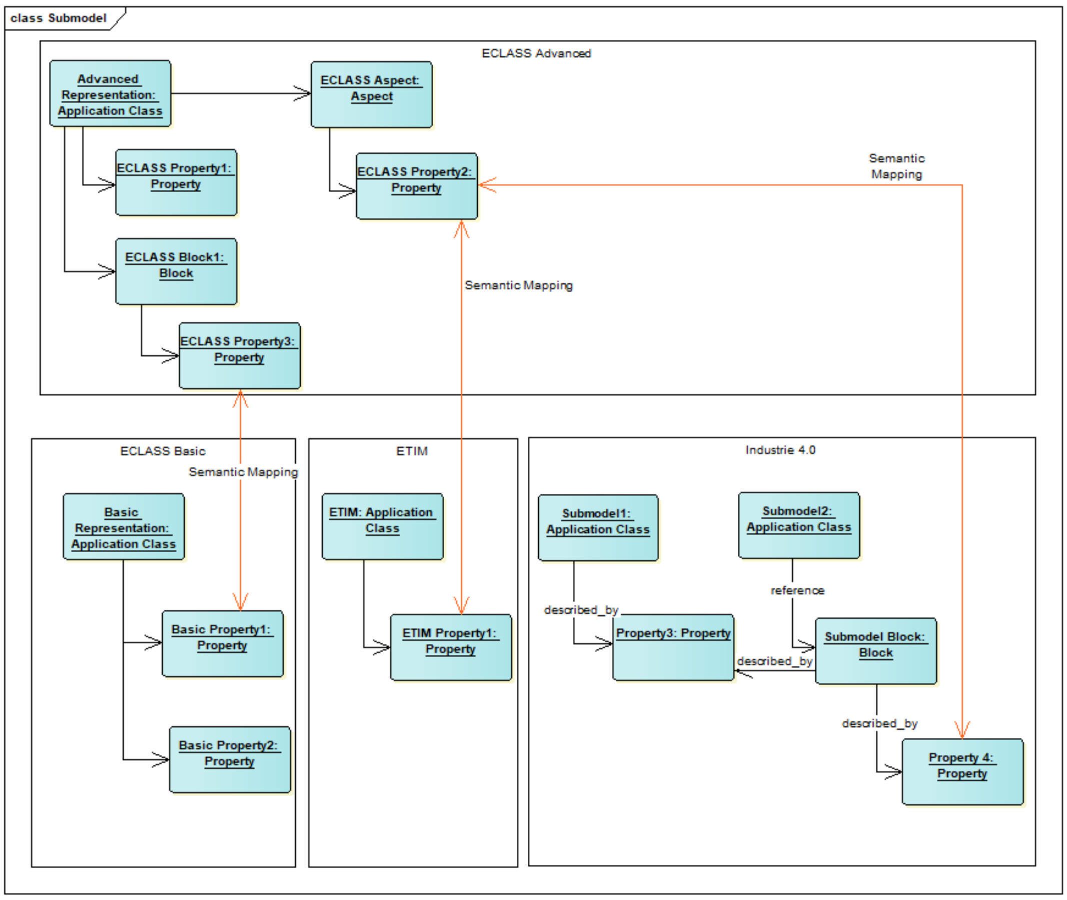


Figure 15: Modelling Concept Descriptions of Submodels as ECLASS Application Classes

Proposal:

The Structure Elements of Submodels are represented within ECLASS in a special Dictionary named "Industrie 4.0". This Dictionary contains Application Classes "Submodel-AC". The Submodel-AC's are semantically linked and semantically mapped to the ECLASS Advanced Application Classes. Properties are mapped in context of their Characterization Classes (similar to the Advanced Basic Mapping explained in [10]).

This proposal allows both the mapping of a Submodel-AC to many ECLASS Advanced AC's and ECLASS Advanced AC may mapped to many Submodel-AC's.

Properties which are describing the Submodel-AC may be

- Imported (unmodified)
- Copied in the Submodel namespace getting a new IRDI (allowing modification)
- or new properties created, if not existing in ECLASS (following ECLASS Rules, see ECLASS Accelerated [17])

The ECLASS Advanced AC will not change if a Submodel-AC is attached or changed, since the Submodel-AC has its own lifecycle and version management.

Since Submodel-AC’s are in ECLASS selfstanding AC’s an export or REST access to a given Submodel-AC should be possible.

Domain experts are currently actively working on the development of standardized Submodel Templates. The Submodel Templates for the “Digital Nameplate for Industrial Equipment” [18] and the “Generic Frame for Technical Data for Industrial Equipment in Manufacturing” [19] are already available in the library of the Plattform Industrie 4.0. Here, ECLASS semantic is used already. (See also [20])

Figure 16 describes the possible concept of the future handling of Concept Descriptions for Submodel-AC’s in ECLASS:

- Domain experts are developing the Submodels ideally based on an existing or future standard.
- Standardization (assignment IRDI), search, distribution (e.g. via web services), maintenance in the life cycle takes place by ECLASS.

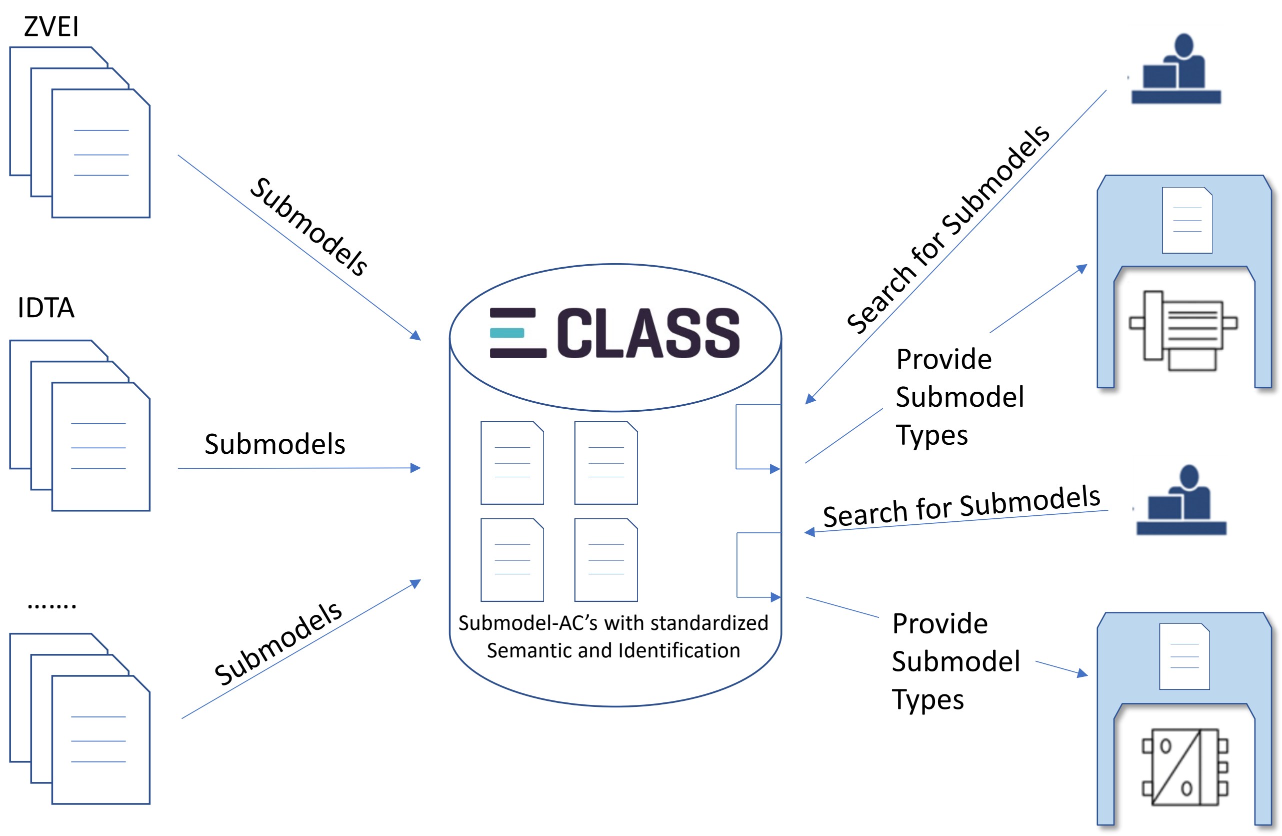


Figure 16: Concept of the Future Handling of Submodel-AC's in ECLASS

5.3 AAS Property

The Concept Description of a Property in the AAS corresponds to a Property in ECLASS with data types not equal to "STRING_TRANSLATABLE" and having no level type.

5.3.1 Category

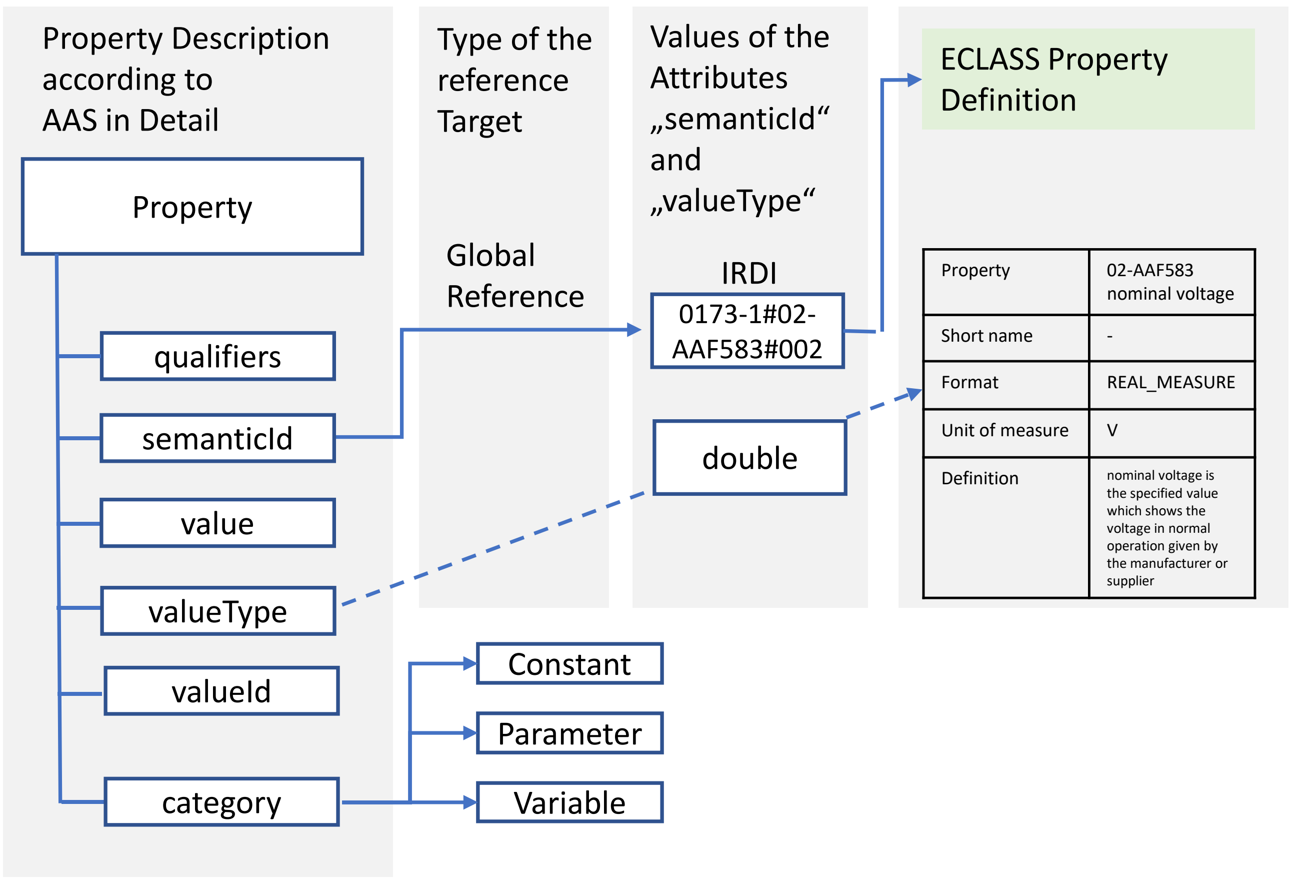
The Property-Value-Pairs of the AAS have a category assigned to the property. Table 1 provides an overview and short explanation of the specified and possible categories of a property.

Table 1: Categories according to the AAS Meta Information Model

Category	**Explanation**
CONSTANT	A Constant Property is a property with a value that does not change over time.
PARAMETER	A Parameter Property is a property that is once set and then typically does not change over time. This is for example the case for configuration parameters.
VARIABLE	A Variable Property is a property that changes of Property-Value-Pairs over time or respectively is calculated during runtime, e.g. its value is a runtime value.

The category can be stated as an attribute of an AAS Property (Figure 17). There is no need to define the category of a Property in the Concept Description, respectively in the semantic Property description provided by ECLASS Dictionary. The ECLASS semantic definitions referenced by its IRDI can be used for all categories of AAS Property-Value-Pairs.

In general, all ECLASS Properties can be interpreted as parameters. In addition, it is possible to declare ECLASS Properties as constants. If the Property should behave like a variable, the definition must be defined in the Application using ECLASS Properties, in this case the AAS.


Property Description according to AAS in Detail
Property
qualifiers
semanticId
value
valueType
valueId
category
Type of the reference Target
Global Reference
Values of the Attributes „semanticId“ and „valueType“
IRDI
0173-1#02-AAF583#002
double
Constant
Parameter
Variable
ECLASS Property Definition

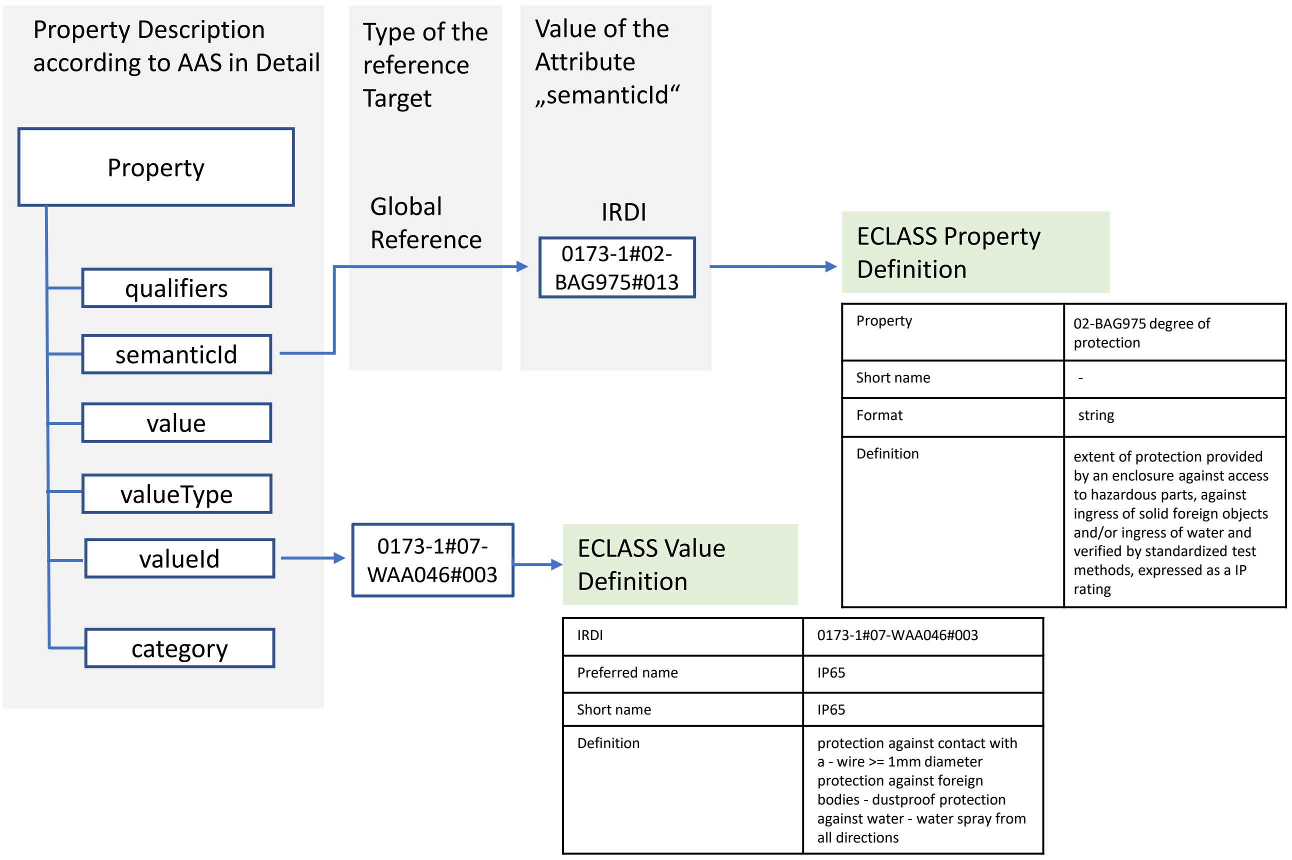
Property	02-AAF583 nominal voltage
Short name	-
Format	REAL_MEASURE
Unit of measure	V
Definition	nominal voltage is the specified value which shows the voltage in normal operation given by the manufacturer or supplier

Figure 17: Mapping of a Property with different Categories to an ECLASS Property

5.3.2 Value

Note, that the semantic of a Property-Value-Pair is defined by a reference of an AAS Property to an ECLASS Property definition. Additionally, AAS offers semantic of the possible elements of the value by the so-called valueId. This can proceed with ECLASS as follows:

- By a reference to the ECLASS Coded Value (via IRDI), e.g., IP65 (Figure 18)

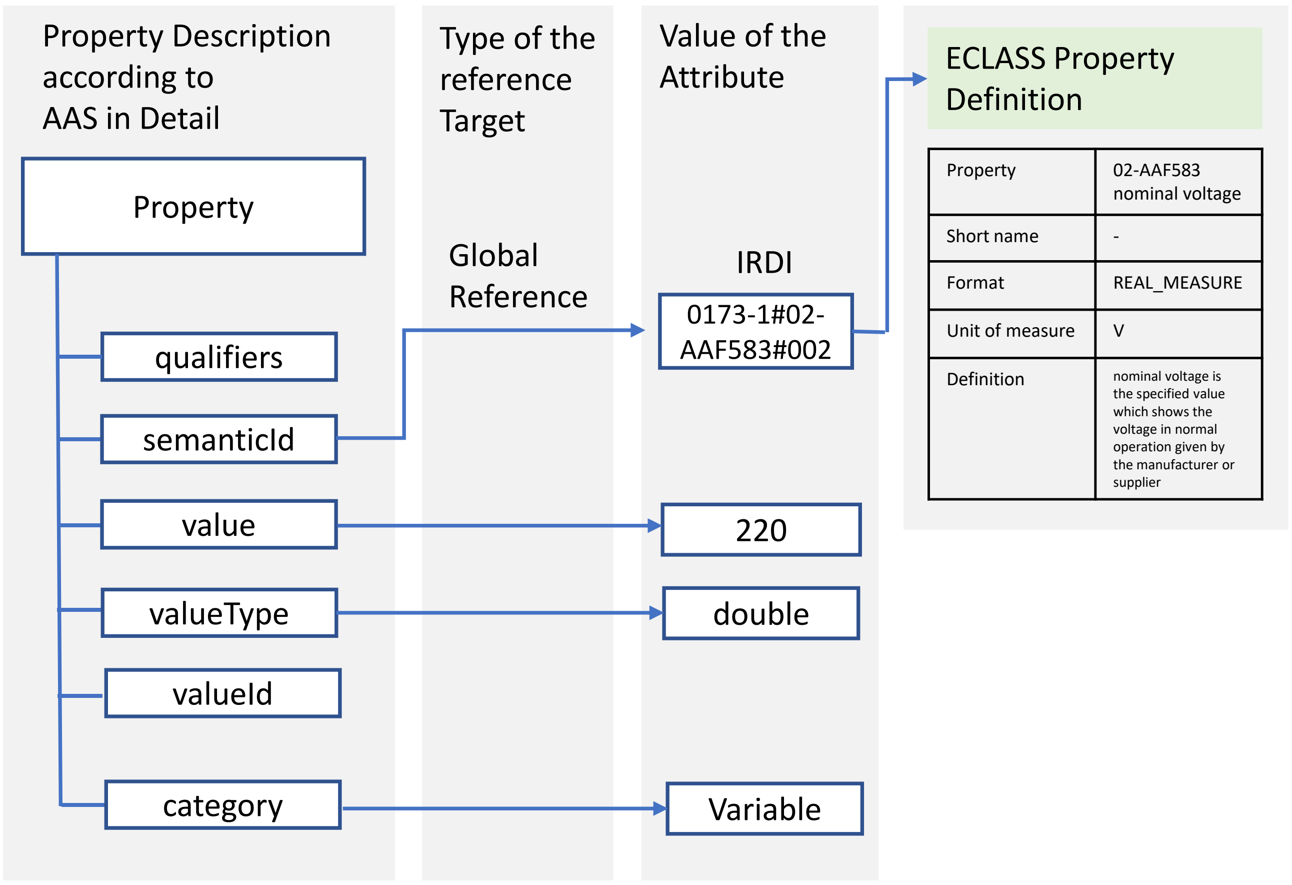


Property	02-BAG975 degree of protection
Short name	-
Format	string
Definition	extent of protection provided by an enclosure against access to hazardous parts, against ingress of solid foreign objects and/or ingress of water and verified by standardized test methods, expressed as a IP rating

IRDI	0173-1#07-WAA046#003
Preferred name	IP65
Short name	IP65
Definition	protection against contact with a - wire >= 1mm diameter protection against foreign bodies - dustproof protection against water - water spray from all directions

Figure 18: Referencing an ECLASS Property with ValueId and Value (IP65)

- Or to an ECLASS Property with an explicit value e.g., 220 with unit V (Figure 19)



Property	02-AAF583 nominal voltage
Short name	-
Format	REAL_MEASURE
Unit of measure	V
Definition	nominal voltage is the specified value which shows the voltage in normal operation given by the manufacturer or supplier

Figure 19: Referencing an ECLASS Property with explicit Value 220

5.3.3 Dependent Property

ECLASS offers the possibility to define so-called Dependent Properties. ECLASS Dependent Properties declare that they depend on other properties (conditions).

Example: Power of motor by a given frequency and voltage. The Property power is depending on conditions frequency 50/60Hz and voltage 110/220V.

Note for the AAS modelling: By using the ECLASS Dependent Properties, the information model instance of the AAS shall

- Define a Property (Dependent Property: power) as well as properties for the conditions (frequency and voltage) as part of a Submodel.
- Provide semantic references to the ECLASS Property and each of its conditions as usual.
- Provide semantic references to ECLASS Values if needed.

5.3.4 Multi Value – Single Value

In ECLASS all properties are treated as multi-valued properties with only few exceptions, e.g., BOOLEAN.

In the AAS the opposite is true: Typically, a property is single valued. Multi-values Properties are not supported by the AAS except for ranges and multi-language Properties.

Proposal:

For supporting the single-valued Property of the AAS the ECLASS Property definition should be enhanced by a corresponding ECLASS Attribute (multi-value=true/false).

5.3.5 Open Value List – Closed Value List

In ECLASS all Value Lists are treated as open as default (except Boolean). Value Lists may be extended by any value in the Property-Value-Pairs, but these values are NOT in (should not be a part of) the ECLASS Property definition.

In the AAS the opposite is true: Since the AAS is meant to enable interoperability Value Lists are closed.

If additional values in the Value List of the ECLASS definition are needed, then there is the following possibility how to proceed:

- Copy the Value List of ECLASS in the local Concept Dictionary of the AAS.
- Extend the Value Range locally. Create a new IRDI for the Concept Description in AAS and reference the ECLASS Property via the isCaseOf reference.

Proposal:

For AAS support the ECLASS Property definition should be enhanced by a corresponding ECLASS attribute (open-value-list=true/false).

5.3.6 Level Type

In ECLASS Dictionary the properties may be modelled as Level Type (e.g., a vector of values: min, max, nom, type).

In AAS Meta Models the properties of this kind should be treated like this except for range, see chapter 5.5:

- Copy of ECLASS Property description in the local Concept Dictionary of the AAS and create (up to) four properties locally.
- Use a new IRDI for each of these copies for the Concept Descriptions in AAS and reference the ECLASS Property via the isCaseOf reference.

Proposal:

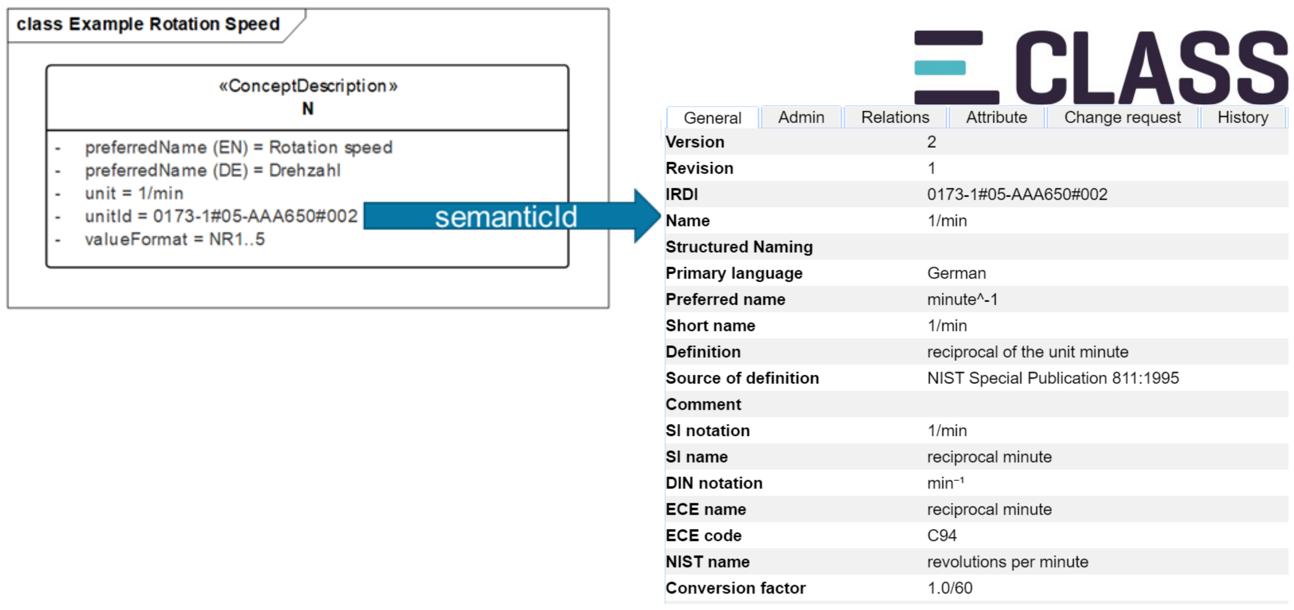
Three proposals:

1. *Remove the Level Type. All properties that have min, max, nom and typ are created as Individual properties. In case min and max shall describe a range, then one Property is to be created with a range constraint according to ISO 13584.*
2. *Extend IRDI by level syntax and extend Level Type by actual value. Level Types can be extended by more levels.*
3. *Extend ECLASS to support collection of properties that share the same data type, unit of measure, etc.*

5.3.7 Physical Unit

In AAS it is possible to define or copy Concept Descriptions for Physical Units if needed.

The Units are defined in the ECLASS Dictionary, so the IRDI as defined in ECLASS can be used directly for the identification of the Units (see below).


class Example Rotation Speed
«ConceptDescription»
N
- preferredName (EN) = Rotation speed
- preferredName (DE) = Drehzahl
- unit = 1/min
- unitId = 0173-1#05-AAA650#002
- valueFormat = NR1..5
semanticId
ECLASS
General
Admin
Relations
Attribute
Change request
History

Version	2
Revision	1
IRDI	0173-1#05-AAA650#002
Name	1/min
Structured Naming	
Primary language	German
Preferred name	minute^-1
Short name	1/min
Definition	reciprocal of the unit minute
Source of definition	NIST Special Publication 811:1995
Comment	
SI notation	1/min
SI name	reciprocal minute
DIN notation	min^{-1}
ECE name	reciprocal minute
ECE code	C94
NIST name	revolutions per minute
Conversion factor	1.0/60

Figure 20: Use of ECLASS IRDI for the Identification of the Unit (UoM)

5.3.8 Data Type Mapping

The ECLASS Data Type Mapping in ECLASS Wiki [21] is envisioned for the mapping of Attribute ValueType of AAS Properties. In XSD (XML Schema Definition) more specializations are possible.

5.4 AAS Multi Language Property

The Concept Description of a MultiLanguageProperty in the AAS corresponds to a Property in ECLASS with Data Type “STRING_TRANSLATEABLE”.

AAS Multilanguage Properties can be represented in ECLASS. As ECLASS is multilingual, properties are described with the following translatable Attributes: Short Name, Definition and Preferred Name.

5.5 AAS Range

The Concept Description of an AAS Range corresponds to an ECLASS Property with Property Levels min and max.

Note: Property Levels are sometimes referred as Level Type.

Proposal:

How to deal with Level Type, see chapter 5.3.

5.6 AAS Entity

The Concept Description of an AAS Entity corresponds to an ECLASS Block. The handling is identical as for the Concept Description of an AAS Submodel Element Collection with Attribute “allowDuplicates=false”.

5.7 AAS Reference Element

The term Reference Element used in the specification of the AAS is similar to the term Reference Property used by ECLASS, which can reference only an ECLASS Block.

In the AAS, however, any Referable Element – like Submodel Element, View, Asset – can be referenced but it is also possible to add an external logical reference.

The references that are specified in AAS are not supported and cannot be modelled by the Elements of the ECLASS Dictionary. ECLASS can only provide a semantic description for the AAS Elements being referenced.

References can reference anything, for this reason a standardized semantic is not applicable or reasonable, so it is not senseful to have References in ECLASS.

Proposal:

No support, it is not senseful for the mentioned reason to have Concept Descriptions for AAS Reference Elements in ECLASS.

5.8 AAS File and BLOB

AAS File and AAS Blob are not explicitly supported in ECLASS, because ECLASS is not supporting the exchange of instance data.

Proposal:

For AAS support the allowed list of ECLASS data types should be extended with the data types FILE and BLOB.

5.9 AAS Submodel Element Collection

Another Submodel Element Type is the Submodel Element Collection, which is used to bundle Submodel Elements and to enable structuring and nesting. The AAS distinguishes between an AAS Submodel Element Collection containing duplicates and those that do not contain duplicates. Duplicates are elements with the “same” semanticId. This is comparable with e.g., blocks in ECLASS. ECLASS in general allows only a one-time use of a property in a specific context (Application Class, Block, Aspect). Please compare to chapter 3.3.

Note: AAS Elements with the same semanticId in the same context mean more than one instance of an ECLASS Property. This is to be distinguished from multiple Value assignment, in which multiple values are assigned to a property in a list for instance. Please compare to chapter 5.3.

Therefore, two cases must be distinguished:

1. **Duplicates = false**

In case no duplicates are allowed, the Concept Descriptions of a Submodel Element Collection correspond to the block concept in ECLASS.

2. **Duplicates = true**

In case duplicates are allowed, the Concept Descriptions of a Submodel Element Collection correspond to the pattern of Cardinality in ECLASS. This allows the reuse of Blocks and thus the properties of the Blocks multiple times (see [22]). To bring a Submodel Element Collection with duplicates = true in the ECLASS Dictionary a Cardinality must be modelled.

However, the resulting IRDI structure shall be used in the AAS as semanticIds. Since this is a complex case, the application will be described here for better understanding. In order not to have to reproduce the multi-level nesting of ECLASS Structure Elements as Submodel Elements in the AAS, the following concept in accordance with the context information is proposed:

To point to the specific Block in ECLASS the semanticId of the Submodel Element Collection contains the IRDI of the referenced properties as indicated in chapter 3.3. In the figure below the focused pattern of Cardinality is nested again. Here, it is proposed to use several keys in order of the nesting in ECLASS to represent the IRDI-Path inside of ECLASS. Therefore, in the shown case two keys are necessary to point to the right position in the ECLASS Structure.

Furthermore, the two properties A1 and A2 should have the same meaning. Therefore, in ECLASS a property must be nested in a Block which again is in a context of Cardinality. To realize this in ECLASS a Reference Property and a referenced Block has be created, and the pattern of Cardinality must be used.

Bringing the standardized semantic back to the Submodel Element and to overcome the nesting problem, it is proposed to use an IRDI-Path out of several keys in the semanticId. **This IRDI-Path is a relative path of the current context in the AAS and not an absolute path.** On instance level like in the AAS it is important to have a semantically unambiguous addressing. Therefore, it is proposed to extend the key with a counter information like *n. Please compare chapter 3.3.

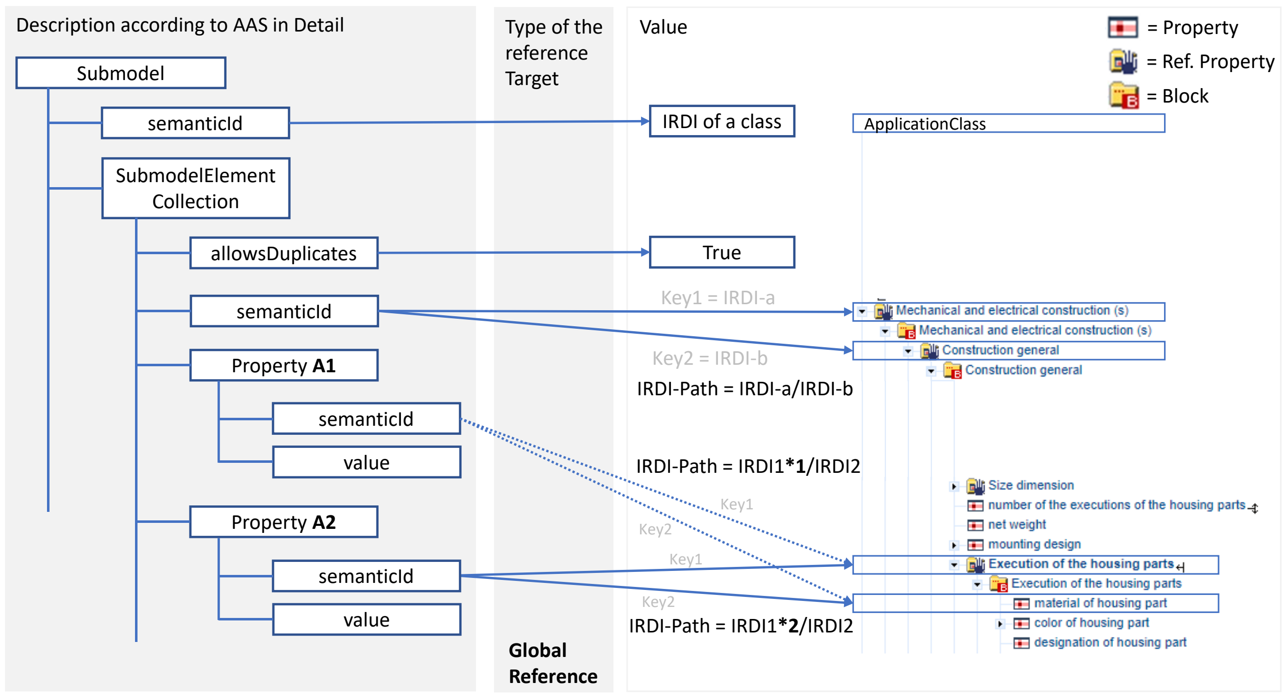

Description according to AAS in Detail
Submodel
semanticId
SubmodelElement Collection
allowsDuplicates
semanticId
Property A1
semanticId
value
Property A2
semanticId
value
Type of the reference Target
Global Reference
Value
= Property
= Ref. Property
= Block
IRDI of a class
ApplicationClass
True
Key1 = IRDI-a
Key2 = IRDI-b
IRDI-Path = IRDI-a/IRDI-b
IRDI-Path = IRDI1*1/IRDI2
Key1
Key2
Key1
Key2
IRDI-Path = IRDI1*2/IRDI2
Mechanical and electrical construction (s)
Mechanical and electrical construction (s)
Construction general
Construction general
Size dimension
number of the executions of the housing parts
net weight
mounting design
Execution of the housing parts
Execution of the housing parts
material of housing part
color of housing part
designation of housing part

Figure 21: AAS Submodel Element Collection Duplicates = true

The Figure 21 shows a Submodel Collection which collects all general construction-relevant properties. It is assumed that the housing of the product consists of two parts. In this example there are two materials used for the different housing parts. The different housing parts are separated by an index (e.g., *1) in the IRDI-Path. The specific materials of the parts are described by the properties A1 and A2. The color for part 1 in this case would be a property A3 with a path IRDI1*1/IRDI3.

Note: Additionally, ordered, and non-ordered Submodel Element Collections are distinguished. The ECLASS Dictionary does not define any order of elements. Only in the interpretation it is ordered via index (e.g., *1).

5.10 AAS Relationship Element

An AAS Relationship Element defines the relationship between two referable elements. The AAS does not restrict the relation between two properties. It is also possible to specify the relationship between an Asset and a Property or the relationship between two Assets or between two AAS. The AAS only requires that the relationship participants can be referenced.

In ECLASS Free Relations could be used to create a relationship between two Structure Elements, where the relationship is not intrinsically predefined in the ECLASS Conceptual Data Model. However, ECLASS Free Relations allow to configure relationship knowledge which may be used in customer specific business logic, workflows, etc.

AAS annotated Relationship Elements are relationships that can additionally be annotated with Data Elements. AAS Data Elements are AAS Properties, AAS Multilanguage Properties, AAS Ranges, AAS Reference Elements as well as AAS Files or AAS Blobs.

Proposal:

For AAS support of AAS relationship elements use the already foreseen but not yet used ECLASS Structure Element "Free Relation Type" and extend this Free Relation Type so that it may be annotated with a Characterization Class. See Figure 22 below.

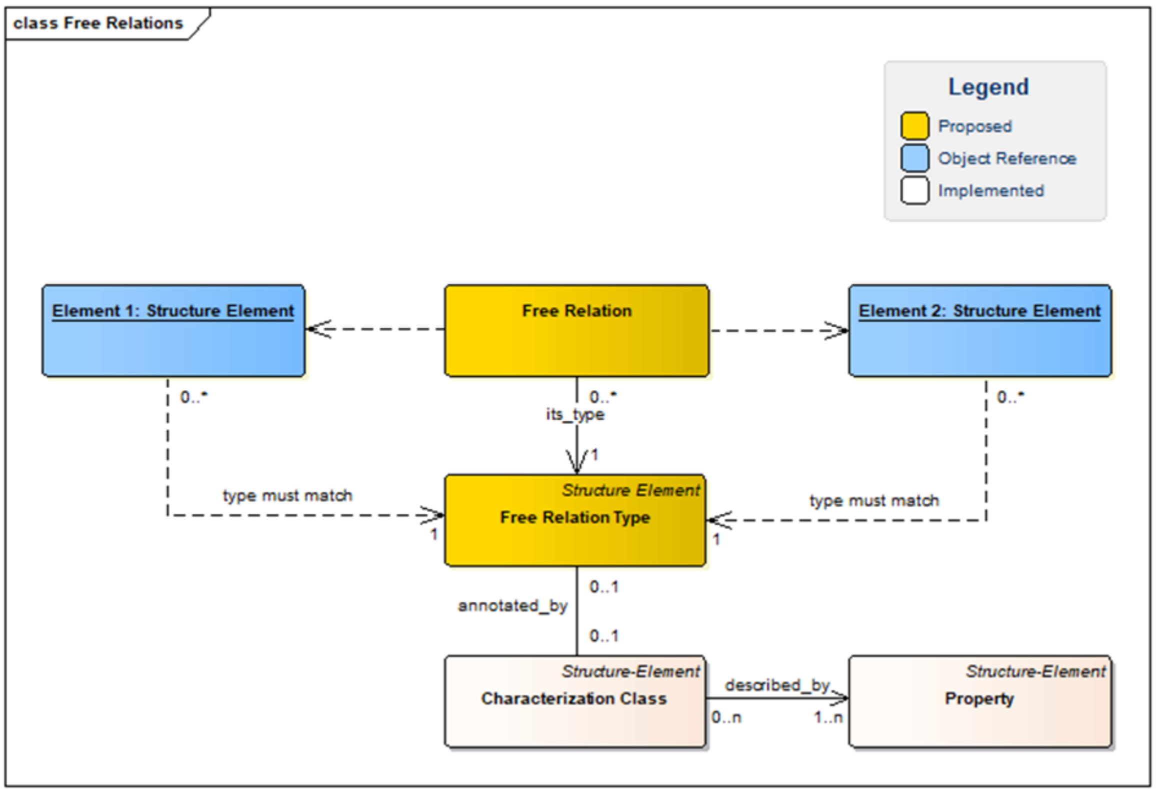


Figure 22: Proposal for the possible Extension of ECLASS Conceptual Data Model to support Concept Descriptions for AAS Relationship Elements

A Free Relation is a Structure Element which is used to define relations between Structure Elements or Instances (e.g., products) which are described with ECLASS. The Free Relation Type defines the characteristics of the Free Relation.

They are part of the ECLASS Conceptual Data Model. However, they are not used in the current ECLASS Releases yet.

With ECLASS Free Relations it is possible to express any relation that is defining any relation-specific logic such as:

- A is cheaper than B
- A sees B

5.11 AAS Capability

Capabilities describe the technical capabilities of assets. In the specification of the AAS a Capability is the implementation-independent description of the potential of an asset to achieve a certain effect in the physical or virtual world.

The Capabilities in the meaning of the AAS specifications are currently not supported by ECLASS, no pre-defined capabilities can be found in the ECLASS Dictionary yet.

Proposal:

For AAS support of AAS Capability elements the ECLASS Conceptual Data Model could be extended by a new Characterization Class “Capability Class”. See Figure 23.

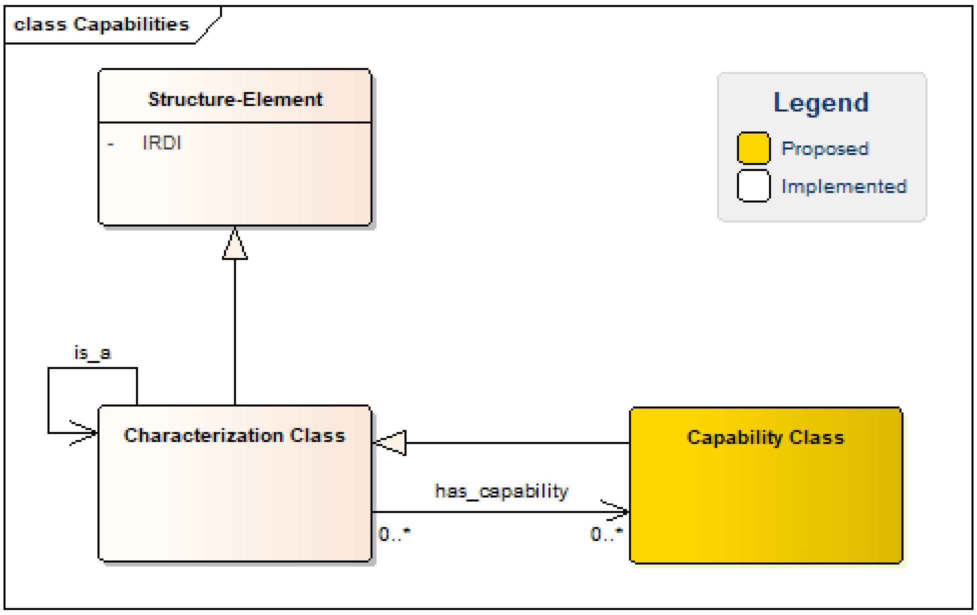


Figure 23: Possible Extension of the ECLASS Conceptual Data Model to support AAS Capabilities

The ECLASS Capability Class (see Figure 23) would be assigned to the corresponding ECLASS Characterization Class. The Capabilities shall be semantically identifiable, each ECLASS Capability therefore receives a unique IRDI.

Example: A "washing machine" (Application Class) can "wash" (Capability). Capability "wash" is a representation of the ability of a washing machine to wash.

5.12 ASS Operation

Operations describe an activity related to an asset that can be executed by a client of the (reactive) AAS. For example, a machine or system has the capability to manufacture a product or to perform a defined process step in manufacturing. Such a capability is realized by an Operation (the skill). Let us call this Operation “produce”. In the AAS the Operation “produce” can be defined to trigger a machine to produce a product. With ECLASS this produce could get a unique semantic. However, the machine shall also receive a description of the product - the production order. To transfer the product description to the manufacturing system, the Operation shall be equipped with an input parameter. Since it is possible to use any Submodel Element as input or output-variable, you can define the semantics of these variables as described in the chapters for the corresponding Submodel Element.

The operations in the meaning of the AAS specifications are currently not supported by ECLASS, no pre-defined Operations can be found in the ECLASS content yet.

Proposal:

For AAS support of AAS Operation elements the ECLASS Conceptual Data Model could be extended by a Structure Element "Operation". Its input, output and inoutput parameters are realized as Characterization Classes. Additionally, allow any Characterization Class to have associated operations. See figure below.

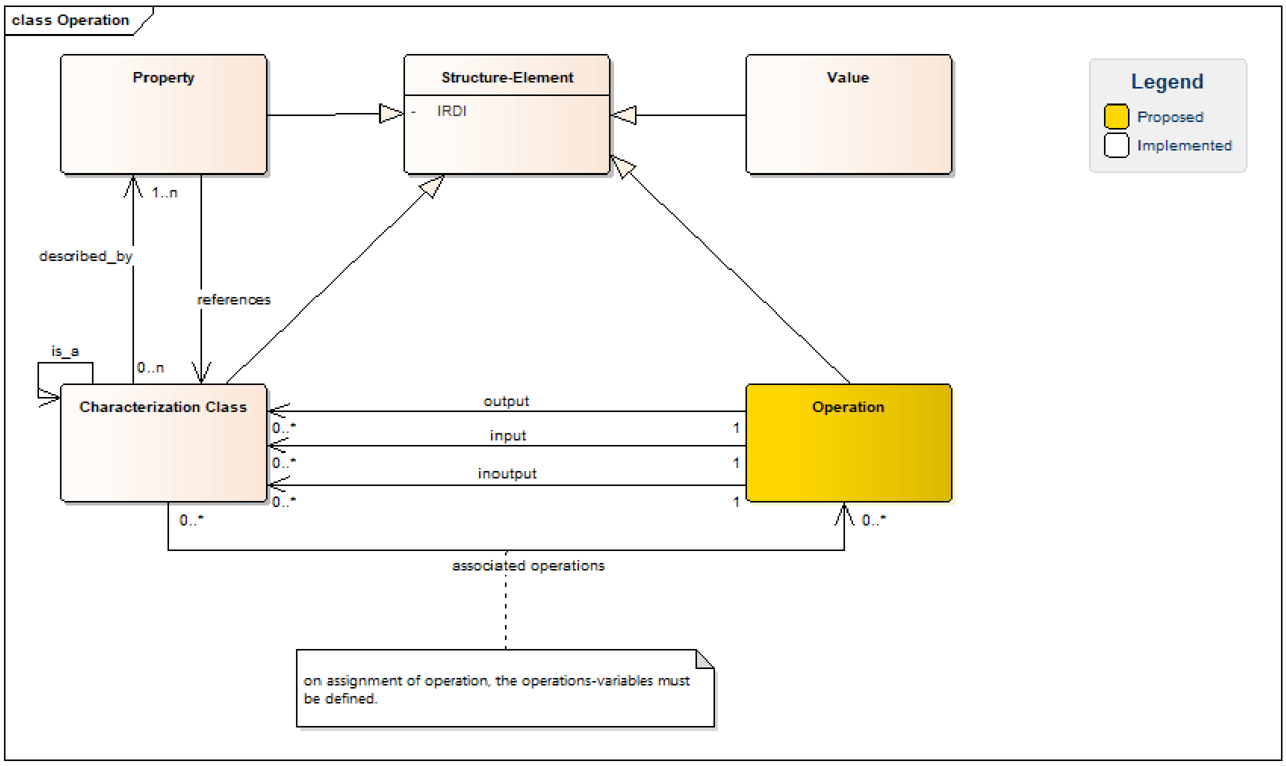


Figure 24: Possible Extension of the ECLASS Conceptual Data Model to support AAS Operations

The Operation Class is defined by a list of elements, typically properties, that serve as input, output or inoutput parameters (so called Operations variables in the AAS).

Example

Use case to be modelled: A lamp that can be turned on, turned off and dimmed.

The following steps would have to be done:

- Model the Operations “turn_on()”, “turn_off()” and “dim(brightness)” as ECLASS Structure bElements “Operation”.
- The operation “dim” has the input variable “brightness”. To model this input variable in ECLASS a specific Characterization Class described by a property “brightness” has to be defined and linked as input to the operation “dim”.
- The defined Operations then are assigned to the Application Class Lamp.

As a result, you have an Application Class “Lamp” which offers the following operations:

- Lamp – turn_on()
- Lamp – turn_off()
- Lamp – dim(30%)

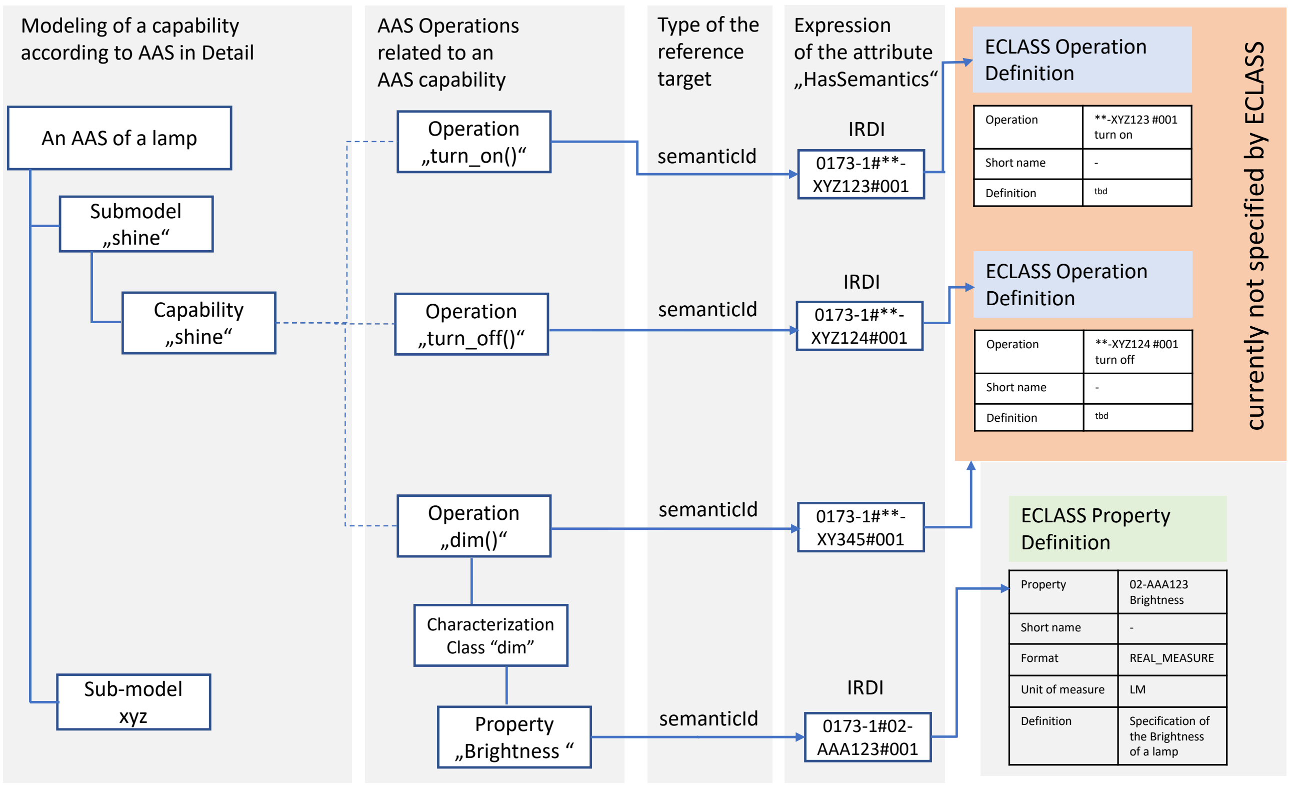


Figure 25: Modelling of AAS Operations as ECLASS Operation Classes

5.13 AAS Event

AAS Events are discrete events such as an alarm or samples of a time series that are pushed regularly. An event is defined by a list of properties that serve as Event Variables. The Event Variables present the observed elements.

Events in the meaning of the AAS specification are currently not supported by ECLASS, no pre-defined event semantics can be found in the ECLASS Dictionary yet.

Proposal:

For AAS support of AAS Event elements the ECLASS Conceptual Data Model could be extended by a Structure Element “Event”. For each event references to its observed elements represented by Characterization Classes can be defined. See below.

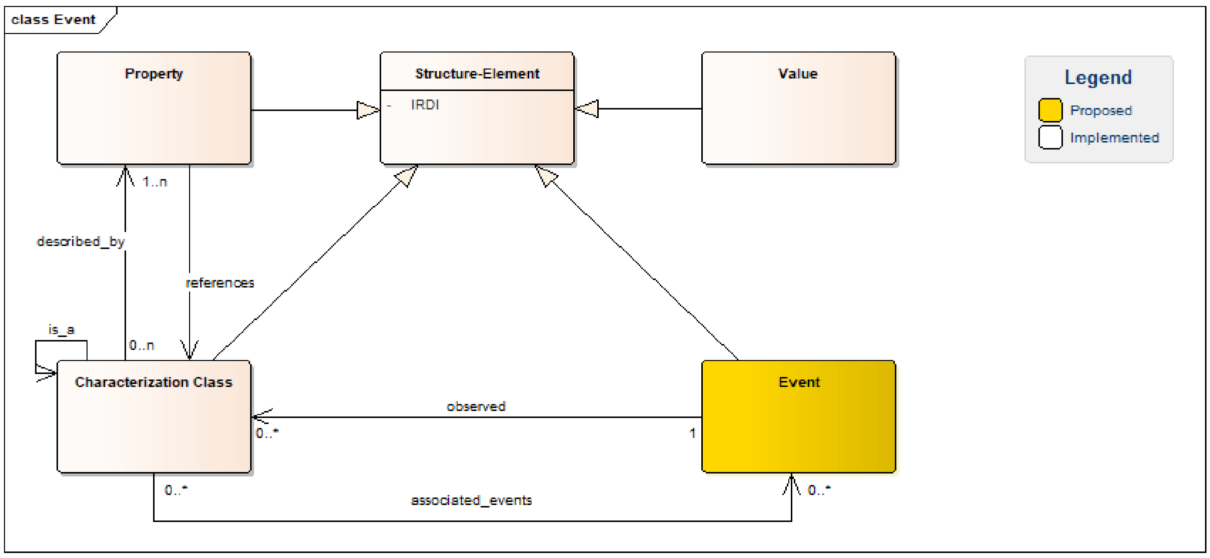


Figure 26: Possible Extension of the ECLASS Conceptual Data Model to support AAS Events

Example 1

Use case to be modelled: A lamp that can send notifications about the status and the brightness.

- Model the Events “status” and “brightness change”
- The Event “status” has the Event Property “is_on” (is_on=true when the lamp is switched on)
- The Event “brightness change” has the Property “brightness”.
- Event Variables are Property-Value-Pair and can be modelled as a AAS Property.
- Application Class: Lamp
- Assign the corresponding Event to the Submodel Application Class “Lamp”

As a result, an Application Class “Lamp” which offers the following Events can be modelled:

- Lamp sends Event: “status” with Property is_on=true
- Lamp sends Event: “brightness” with Property brightness = 29%

Example 2

Use case to be modelled: A washing machine that can send notifications about the completion of the washing process and in case of error the failure notification with the failure code.

- Model the Events “status” and “error”.

- The Event “status” has the event Property “wash_is_finished”
- The Event “error” has the Property “error_code”.
- Event Variables are Property-Value-Pair and can be modelled as AAS Properties.
- Application Class: washing machine
- Assign the corresponding Event to the Submodel Application Class “washing machine”

As a result, an Application Class “washing machine” which offers the following Events can be modelled:

- Washing machine sends Event: “status” with Property wash_is_finished = true
- Washing machine sends Event: “error” with Property “error_code” = xyz

5.14 AAS Qualifier

The AAS Qualifier is a defined element associated with a Property Instance or Submodel Element, restricting the Value Statement to a certain life cycle time (example: as planned) or use case.

The Qualifiers in the meaning of the AAS specifications are currently not supported by ECLASS, e.g., no pre-defined Qualifiers can be found in the ECLASS Classification yet.

Proposal:

For AAS support of AAS Qualifiers the ECLASS Conceptual Data Model could be extended by a new Structure Element “QualifierType” and “Qualifier”, the ECLASS Qualifiers being the possible values of a Qualifier-Type. See Figure 27 below.

The AAS Qualifier corresponds to a pair of Qualifier Type and Qualifier Value. Therefore, the ECLASS Qualifier should be modelled similar to a property (with closed Value List). Both, the ECLASS Qualifier Type and the ECLASS Qualifier Value should get unique IRDI's.

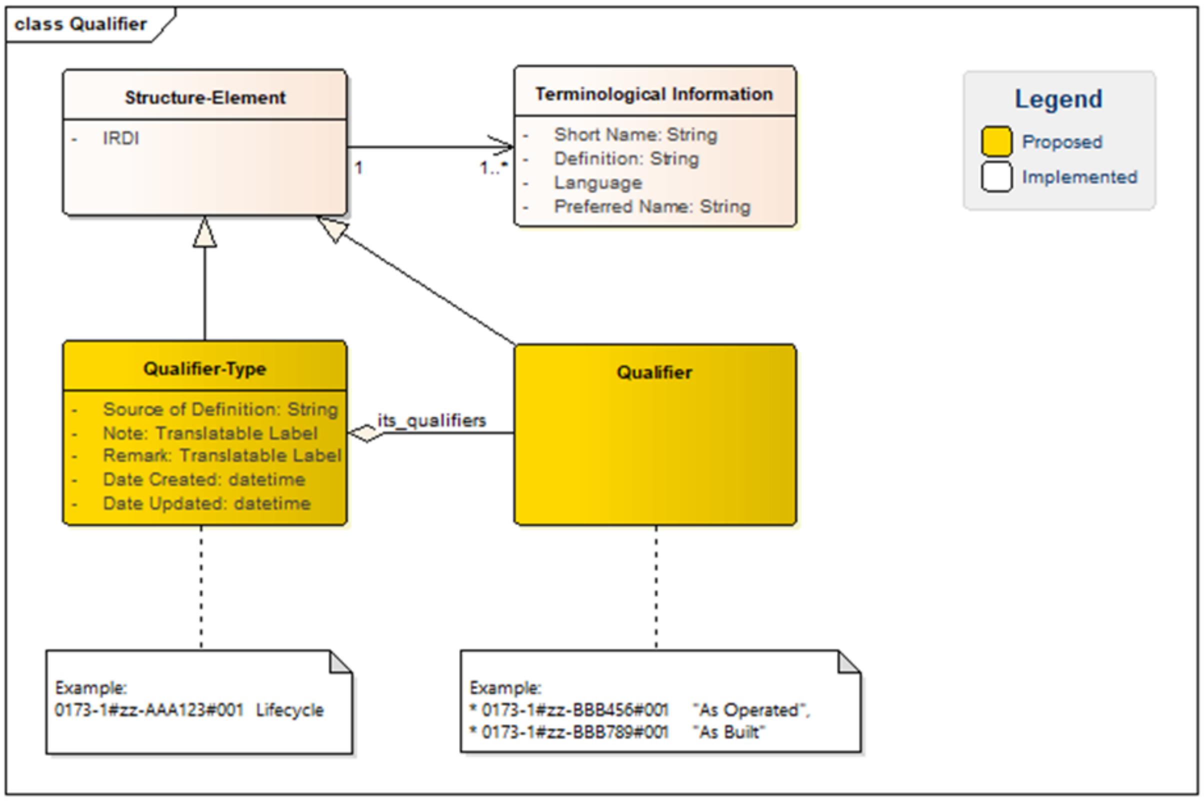


Figure 27: Possible Extension of ECLASS Conceptual Data Model for Semantics of AAS Qualifier

6 Summary and Outlook

In this document all Submodel elements of the Asset Administration Shell were examined how to define a suitable concept description of each of them in ECLASS. Additionally, concept descriptions for Qualifiers were examined.

For some Submodels Elements an existing ECLASS concept could be directly used, for example for the widely used Properties. For others like Relationship Elements there are solutions already foreseen, but not used so far. For Capabilities, Operations, and some other elements the ECLASS Conceptual Data Model would need extensions to provide a standardized semantic to define complete Digital Twins. For each of these extensions an explicit proposal of how to extend the ECLASS Conceptual Data Model is introduced.

ECLASS provides already today a beneficial basis for describing the semantics of data for the Asset Administration Shell or Digital Twin. With the proposed extensions implemented in ECLASS a big step forward to describe also enhanced semantics in an unambiguous way would be possible.

ECLASS e.V. and Plattform Industrie 4.0 together are willing to continue the path to make standardized Digital Twins become real and to contribute to the digitalization of the manufacturing industry.

References

[1] Bundesministerium für Wirtschaft und Energie (BMWi): Details of the Administration Shell - Part 1: The exchange of information between partners in the value chain of Industrie 4.0, Version 3.0RC01. 2020. Online: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html

[2] Bundesministerium für Wirtschaft und Energie (BMWi): Details of the Asset Administration Shell. Part 2 - Interoperability at Runtime - Exchanging Information via Application Programming Interfaces. Version 1.0RC01. 2020. https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part2_V1.html

[3] Plattform Industrie 4.0 in Kooperation mit VDI/VDE-GMA Fachausschuss 7.20: I4.0-Sprache. Vokabular, Nachrichtenstruktur und semantische Interaktionsprotokolle der I4.0-Sprache (German). 2018. https://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/hm-2018-sprache.html

[4] VDI/VDE 2193 Blatt 2: Language for I4.0 components -Interaction protocol for bid-ding procedures. Beuth Verlag. 2020. https://www.beuth.de/en/technical-rule/vdi-vde-2193-blatt-2/3141143

[5] ECLASS Wiki: https://wiki.eclass.eu/wiki/Main_Page

[6] ECLASS Wiki – ECLASS Conceptual Data Model: https://wiki.eclass.eu/wiki/Conceptual_data_model

[7] ECLASS e.V.: Technical specification 11. Conceptual Data Model, 2020. https://www.eclass.eu/static/documents/wiki/Technical_Specifications/eCl@ss_Technical-Specification_11_Conceptual-Data-Model_v_1.0.pdf

[8] ECLASS Wiki – IRDI: https://wiki.eclass.eu/wiki/IRDI

[9] ECLASS Wiki – ECLASS Update: https://wiki.eclass.eu/wiki/ECLASS-Update

[10] ECLASS Wiki – Mapping Basic-Advanced: https://wiki.eclass.eu/wiki/Basic-Advanced_Mapping

[11] ECLASS Wiki – Transaction: https://wiki.eclass.eu/wiki/Transaction_Update_File

[12] ECLASS Wiki – Classification: https://wiki.eclass.eu/wiki/Classification_Update_File

[13] eCl@ssXML 3.0 – XML Schema: https://www.eclass.eu/static/eClassXML/3.0/eCl@ssXML/mapping.xsd

[14] ECLASS Wiki – References: https://wiki.eclass.eu/wiki/Reference_algorithm

[15] ECLASS Wiki – IRDI-Path: https://wiki.eclass.eu/wiki/IRDI-Path

[16] Bundesministerium für Wirtschaft und Energie (BMWi): Relationships between I4.0 components - composite components and intelligent production. 2017.

https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/hm-2018-relationship.html

[17] Ondracek, N. & Treitinger, G.: ECLASS Fast track. ECLASS Congress, 2019

[18] Bundesministerium für Wirtschaft und Energie (BMWi): Submodel Templates of the Asset Administration Shell - ZVEI Digital Nameplate for industrial equipment (Version 1.0). 2020. https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Submodel_templates-Asset_Administration_Shell-digital_nameplate.html

[19] Bundesministerium für Wirtschaft und Energie (BMWi): Submodel Templates of the Asset Administration Shell - Generic Frame for Technical Data for Industrial Equipment in Manufacturing (Version 1.1). 2020. https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Submodel_Templates-Asset_Administration_Shell-Technical_Data.html

[20] Eichberger, D. & Kämper, B. & Müller, J.: Integration von Industrie 4.0 Teilmodellen in ECLASS und OPC UA Companion Specifications. Automation 2020, Baden-Baden.

[21] ECLASS Wiki – Datatype to XSD mapping: https://wiki.eclass.eu/wiki/Datatype_to_XSD_mapping

[22] ECLASS Wiki – Cardinality: http://wiki.eclass.eu/wiki/Cardinality#Cardinality