



<AutomationML/>

**The Glue for Seamless
Automation Engineering**



White paper

**AutomationML and eCl@ss
integration**

Document Identifier: WP eClassAML, V 1.0.1

State: December 2017

Common Working Group of AutomationML e.V and eCl@ss e.V.

Contributer:

Olaf Graeser	PHOENIX CONTACT GmbH & Co. KG
Lorenz Hundt	inpro Innovationsgesellschaft für fortgeschrittene Produktionssysteme in der Fahrzeugindustrie mbH
Michael John	Siemens AG
Gerald Lobermeier	Weidmüller Interface GmbH & Co. KG
Arndt Lüder	Otto-von-Guericke-Universität Magdeburg
Stefan Mülhens	AmpereSoft GmbH
Nikolaus Ondracek	Paradine GmbH
Mario Thron	Institut für Automation und Kommunikation e.V.
Josef Schmelter	PHOENIX CONTACT GmbH & Co. KG

Version 1.0.1, December 2017

Contact: www.automationml.org
 www.eclass.de

Table of content

Table of content.....	3
List of figures	5
List of tables	7
1 Introduction and Scope	8
2 Terms, definitions and abbreviations	10
2.1 Terms and definitions.....	10
2.1.1 AutomationML	10
2.1.2 Automation object.....	10
2.1.3 AutomationML object	10
2.1.4 AutomationML class	10
2.1.5 AutomationML attribute	10
2.1.6 AutomationML document	10
2.1.7 AutomationML file.....	10
2.1.8 AutomationML interface	10
2.1.9 eCl@ss Classification Class	10
2.1.10 eCl@ss Property	11
2.1.11 Value	11
2.1.12 Unit	11
2.1.13 Keyword	11
2.1.14 Synonym	11
2.1.15 Value List.....	11
2.1.16 Application Class.....	11
2.1.17 Aspect	11
2.1.18 Block	11
2.2 AutomationML library	11
2.2.1 Instance	11
2.2.2 Instance hierarchy	11
2.2.3 Link	11
2.3 Abbreviations	11
2.4 Normative References	13
3 AutomationML	14
3.1 Basics.....	14
3.2 Capabilities for semantic integration.....	16
4 eCl@ss.....	17
4.1 Basic Representation.....	17
4.2 Advanced Representation.....	17
4.3 New structural elements of eCl@ss Advanced Representation	18
4.3.1 Block.....	18
4.3.2 Aspect	18
4.3.3 Cardinality	19
4.3.4 Polymorphism.....	19
4.3.5 Units	20
4.4 Export formats.....	20
4.4.1 CSV for Basic Representation	20
4.4.2 eCl@ss XML for Basic and Advanced Representation	20
4.5 Release – Update - Support	20

4.5.1	RUF	20
4.5.2	TUF	21
4.6	IRDI	21
5	Use cases and considered data exchange structures	23
5.1	Technical use cases	23
5.1.1	Use case – simple semantic identification	23
5.1.2	Use case – semantic identification of properties	24
5.1.3	Use case – semantic identification of structures	24
5.2	Exchange use cases	25
5.2.1	Assumptions for considered use cases	26
5.2.2	Use case - exchange of semantically unique instance data	27
5.2.3	Use case – encode data semantics	28
5.2.4	Use case – decode data semantics	30
5.2.5	Use case – eCl@ss dictionary exchange	31
5.2.6	Use case – receive eCl@ss dictionary	32
5.2.7	Use case – deliver eCl@ss dictionary	33
5.3	Application use cases	33
5.3.1	Use case - development and use of semantically unique component libraries	34
5.3.2	Use case - lossless exchange between system configurator and CAx tool	35
5.3.3	Use case - construction validation	37
5.4	Boundary Conditions for Application	39
6	Realization of technical use cases	40
6.1	Integration of attribute and object semantics	40
6.1.1	Concept	40
6.2	Generation process for eCl@ss AutomationML role model	44
6.2.1	Concept	44
6.2.2	Attribute transformation best practice	48
6.2.3	Example	50
6.3	Generation process for eCl@ss advanced AutomationML model	56
7	Literature	57

List of figures

Figure 1 – General engineering data exchange process between engineering tools	8
Figure 2 – AutomationML base structure.....	14
Figure 3 – AutomationML topology description architecture	15
Figure 4 – Means for data semantic modelling in AutomationML.....	16
Figure 5 – eCl@ss classification data structure.....	17
Figure 6 – Example block description.....	18
Figure 7 – Example for cardinality for properties	19
Figure 8 – Example for polymorphism in eCl@ss	19
Figure 9 – ISO / IEC 6523 Definition for use of IRDI	21
Figure 10 – General engineering activities embedded within production system engineering (subset)	25
Figure 11 – System of relevant use cases for semantically unique exchange of data points	26
Figure 12 – Activity structure of use case "exchange of semantically unique instance data"	28
Figure 13 – Activity structure of use case "Encode data semantics"	29
Figure 14 – Activity structure of use case "decode data semantics"	30
Figure 15 – Activity structure of use case "eCl@ss dictionary exchange"	31
Figure 16 – Activity structure of use case "receive eCl@ss dictionary"	32
Figure 17 – Activity structure of use case "deliver eCl@ss dictionary"	33
Figure 18 – System of relevant use cases for semantically unique exchange of data points	34
Figure 19 – Activity structure of use case "development and use of semantically unique component libraries"	35
Figure 20 – Structure of use case „lossless exchange between system configuration tool and CAx tool“	36
Figure 21 – Activity structure of use case „lossless exchange between system configuration tool and CAx tool“	36
Figure 22 – Structure of use case “validation of construction”	37
Figure 23 – Activity structure of use case "validation of construction"	38
Figure 24 – Applicable means for semantic reference for a single attribute	40
Figure 25 – Example RefSemantic for a single attribute	41
Figure 26 – XML description of the role class eClassClassSpecification	42
Figure 27 – Applicable means for semantic reference for AutomationML Object	43
Figure 28 – Example motor AutomationML object with semantic reference for a) AC and b) CC	43
Figure 29 – Attributes to the example motor AutomationML object with semantic reference for a) CC and b) AC	43
Figure 30 – XML View of the example motor AutomationML object with semantic reference (AC).....	44
Figure 31 – Applicable means for semantic representation used for eCl@ss Basic modelling	44
Figure 32 – Example Role Class	46
Figure 33 – Attributes within an Example Role Class.....	46
Figure 34 – Example Role Class XML View	47
Figure 35 – Example eClass classification role class library	50
Figure 36 – XML representation of example eClass classification role class library	52
Figure 37 – Example of AutomationML role class attributes for eCl@ss basic AC of 27-02- 25-01 DC engine (IEC)	53
Figure 38 – XML representation of example of attributes for the eClass classification role motor	53
Figure 39 – Example of system unit class library including the attributes for the SUCs	54

Figure 40 – XML representation of example of system unit class library	55
Figure 41 – Example of instance hierarchy	55
Figure 42 – XML representation of the example instance hierarchy	56

List of tables

Table 1 – Abbreviations	11
Table 2 – Code Space Identifier According To ISO/TS 29002-5 (Restricted)	22
Table 3 – Overview Use Cases	23
Table 4 – Definition eClassClassSpecification	42
Table 5 – Translation of the attributes of eCl@ss to AML	48
Table 6 – Translation of the data types from eCl@ss to AutomationML permitted data types	49

1 Introduction and Scope

Engineering processes of technical systems and its embedded automation systems have to be executed with increasing efficiency and quality. Especially the project duration has to be shortened while the complexity of the engineered system increases. To solve this problem the engineering process is more and more executed by exploiting software based engineering tools exchanging engineering information and artefacts along the engineering process related tool chain.

While the efficiency of the different engineering tools has been considered for a long time and important gains have been reached, the exchange of engineering data is still an important issue. Within this process engineering information developed within one engineering tool has to be transmitted to another engineering tool without any loss of information and without any misinterpretation.

Usually the engineering information of a sending engineering tool is stored in this tool using the tool internal data model as a tool internal project structure. To transmit this information to another tool the project data have to be transformed to data within a data exchange format and stored as exported data file. This data file has to be read by the information receiving engineering tool. The incoming data within the data exchange format have to be transformed to project data following the internal data model of the receiving tool. Thus a virtual mapping of the tool internal data models of the sending and the receiving tool is required as represented in Figure 1.

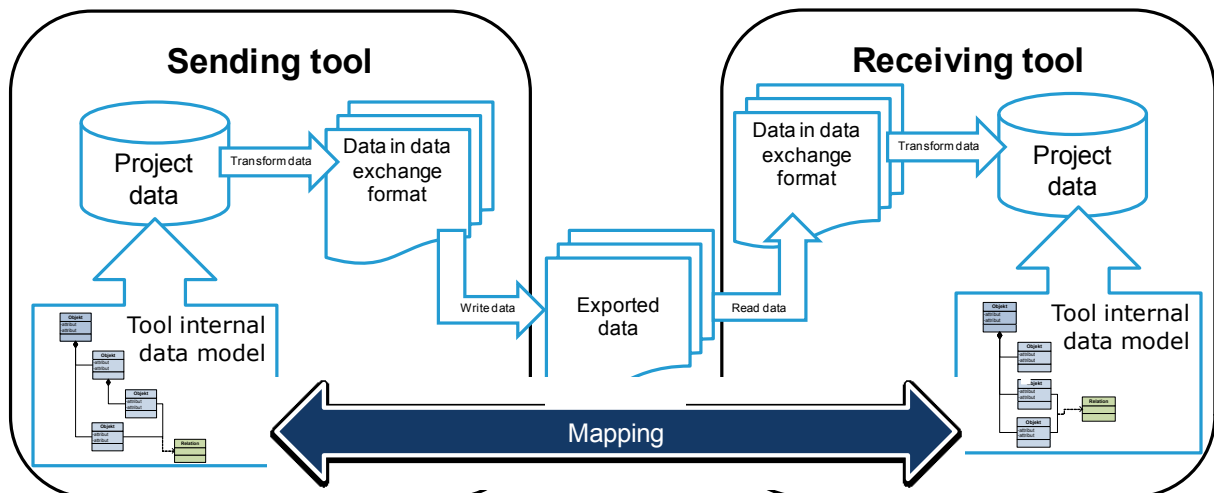


Figure 1 – General engineering data exchange process between engineering tools

The critical point in the described process is the correct interpretation of received data within the information receiving tool. For each incoming data point two main properties have to be given:

1. The incoming data have to be readable, i.e. they are provided in a syntax, which can be automatically read by the receiving tool.
2. The incoming data have to be correctly mappable to the tool internal data model, i.e. the semantics of each incoming data point need to be identifiable automatically by the receiving tool.

To ensure both properties the data exchange format specification can follow two approaches.

Within the first approach syntax and semantic are defined in combination for each data point expressible in the data exchange format, i.e. for each data point a dedicated expression is integrated in the data exchange format only useable for this data point. Such approaches are applied for example in STEP. They have the strong drawback, that they are not easily extendable or adaptable to different application cases.

Thus, the second approach avoids the fixed specification of data object semantic. It defines the syntax of data objects and enables the integration of a semantic specification, i.e. each object will carry its own semantics definition. This approach is applied in AutomationML for example.

Nevertheless, the semantic definition carried within the data object has to be unique. Therefore, appropriate means have to be available.

Within the industry, several catalogue standards are available defining object and property semantics uniquely. Usually they follow the IEC 61360 standard. One example is the eCI@ss catalogue standard. Within this standard a four layer object definition hierarchy with object properties is defined enabling the unique identification of object types like automation device types and its property types like vendor names.

Such catalogue standard will be valuable for the definition of object semantics within a data exchange format. This whitepaper will take up this idea. It will define a methodology how eCI@ss and similar catalogues could be applied to define unique data point semantics within AutomationML.

2 Terms, definitions and abbreviations

2.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62714-1 and the following apply.

2.1.1 AutomationML

XML based data exchange format for plant engineering data

2.1.2 Automation object

Entity in the automated system

Note: An example of an automation object is an automation component, a valve or a signal.

2.1.3 AutomationML object

Data representation of an automation object or a group of automation objects

Note 1: The AUTOMATIONML object is the core element of AUTOMATIONML. It may contain administration items, attributes, interfaces, relations and references. An AutomationML object is an individual instance and is derived from standard AutomationML classes.

Note 2: AutomationML objects have a relation to their corresponding AutomationML class.

Note 3: Examples for relations are type-instance-relations and copy-instance-relations. Single instances can be extended, e.g. by aggregated objects or attributes.

Note 4: AutomationML objects have a copy-instance-relation to their AutomationML class.

2.1.4 AutomationML class

Predefined AutomationML object type

Note 1: AutomationML classes are stored within AutomationML libraries.

Note 2: AutomationML classes define reusable sample solutions, characterized by attributes, interfaces and aggregated objects.

Note 3: AutomationML classes can be used for multiple instantiations.

2.1.5 AutomationML attribute

Attribute which belongs to an AutomationML object

Note: AutomationML attributes are described as XML element corresponding to IEC 62424:2008.

2.1.6 AutomationML document

Certain CAEX document following IEC 62714 including all referenced sub documents

Note: AutomationML documents may be stored as files, but also e.g. as string or data streams.

2.1.7 AutomationML file

Certain CAEX file following IEC 62714 with the extension .aml excluding all referenced sub files

2.1.8 AutomationML interface

Single connection point that belongs to an AutomationML object and can be linked with another interface

Note: Interfaces allow the description of relations between objects by the definition of CAEX InternalLinks. Examples are a signal interface, a product interface or a power interface.

2.1.9 eCl@ss Classification Class

Class for the classification of products into certain categories

2.1.10 eClass Property

Characteristic of a class

2.1.11 Value

A specification of a characteristic

2.1.12 Unit

Standardized unit of measure

2.1.13 Keyword

An alternative name of a class

2.1.14 Synonym

An alternative name of a property

2.1.15 Value List

A restrictive list of valid specifications of a property

2.1.16 Application Class

Class that comprises all characteristics described by properties

2.1.17 Aspect

Sub-class of an application class that comprises all properties describing a certain aspect of a product, not the product itself, e.g. packing information

2.1.18 Block

Sub-class of an application class that comprises all properties describing a certain part of a product

2.2 AutomationML library

Library containing AutomationML classes

2.2.1 Instance

Data representation of a specific real world item or concrete logical engineering item

2.2.2 Instance hierarchy

Hierarchy of AutomationML objects

2.2.3 Link

Connection between objects of the top-level format CAEX

Note: A link is modelled by means of CAEX InternalLink.

2.3 Abbreviations

For the purpose of this document the abbreviations listed in Table 1 apply.

Table 1 – Abbreviations

Abbreviation	Meaning
AC	Application Class
ASN	Abstract Syntax Notation
AS	Aspect
AutomationML	Automation Markup Language
BL	Block

Abbreviation	Meaning
CAX	Computer aided X
CAEX	Computer Aided Engineering Exchange
CC	Classification Class
COLLADA	Collaborative Design Activity
CSI	Code Space Identifier
CSV	Comma Separates Values
DIN	Deutsche Industrienorm
ECAD (E-CAD)	Electrical engineering tool
EDS	Electronic Data Sheet
GUID	Global Unique Identifier
GSDML	Generic Station Description Markup Language
GSD	General Station Description
HMI	Human Machine Interface
ICD	International Code Designator
ID	Identifier
IEC	International Electrotechnical Commission
IE	InternalElements
IH	InstanceHierarchy
IRDI	International Registration Data Identifier
ISO	International Organization for Standardization
ISO/PAS	International Organization for Standardization Public Available Standard
KW	Keyword
MCAD (M-CAD)	Mechanical engineering tool
OCL	Object Constraint Language
OntoML	Ontology Markup Language
OSI	Open Systems Interconnection
P&I diagram	Plant and instrumentation diagram
P&ID tool	Plant and instrumentation diagram tool
PCE-CAE tool	Process Control Engineering Computer Aided Engineering tool
PLC	Programmable Logic Controller
PR	Property
RUF	Release-Update-File
STEP	STandard for the Exchange of Product model data
SUC	System unit classes
SUC LIB	System unit Class Library
SY	Synonym
TUF	Transaction-Update-file
URL	Uniform resource locator
UML	Unified Modeling Language
UN	Unit
URI	Uniform Resource Identifier
UUID	Universal Unique Identifier
XML	Extensible Markup Language

Abbreviation	Meaning
VA	Value
VL	Value List
W3C	World Wide Web Consortium

2.4 Normative References

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 60027 (all parts), *Letter symbols to be used in electrical technology*

IEC 61131-3:2013, *Programmable controllers – Part 3: Programming languages*

IEC 61158 (all parts), *Industrial communication networks – Fieldbus specifications*

IEC 61360 (all parts), *Standard data element types with associated classification scheme for electric components*

IEC 61784 (all parts), *Industrial communication networks – Profiles*

IEC 62424:2008, *Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools*

IEC 62714 (all parts), *Engineering data exchange format for use in industrial systems engineering – AutomationML*

ISO 13584-32:2010-12, *Industrial automation systems and integration - Parts library - Part 32: Implementation resources: OntoML: Product ontology markup language*

ISO 6523 (all parts), *Information technology - Structure for the identification of organizations and organization parts*

ISO 80000-1:2009-11, *Quantities and units – Part 1: General*

ISO/IEC 11179-6:2015-08, *Information technology – Metadata registries (MDR) – Part 6: Registration*

ISO/IEC 9834-8:2014-08, *Information technology - Open Systems Interconnection - Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components*

ISO/PAS 17506:2012, *Industrial automation systems and integration - COLLADA digital asset schema specification for 3D visualization of industrial data*

ISO/TS 29002-5:2009-02, *Industrial automation systems and integration - Exchange of characteristic data - Part 5: Identification scheme*

Extensible Markup Language (XML) 1.0 1.0:2004, W3C Recommendation
(available at <<http://www.w3.org/TR/2004/REC-xml-20040204/>>)

PLCopen XML 2.0:December 3rd 2008 and PLCopen XML 2.0.1:May 8th 2009, XML formats for IEC 61131-3
(available at <<http://www.plcopen.org/>>)

3 AutomationML

3.1 Basics

AutomationML is a solution for data exchange focusing on the domain of engineering of automation systems.

The AutomationML data format, developed by AutomationML e.V., is an open, neutral, XML-based, and free data exchange format which enables a domain and company spanning transfer of engineering data of production systems in a heterogeneous engineering tool landscape. The goal of AutomationML is to interconnect engineering tools in their different disciplines, e.g. plant planning, mechanical engineering, electrical engineering, process engineering, process control engineering, HMI development, PLC programming, robot programming, etc.

AutomationML stores engineering information following the object oriented paradigm and allows modelling of physical and logical plant components as data objects encapsulating different modelling aspects. An object may consist of other sub-objects, and may itself be part of a larger composition or aggregation. Typical objects in plant automation comprise information on topology, geometry, kinematics and logic, whereas logic comprises sequencing, behaviour and control.

In addition, AutomationML follows a modular structure by integrating and enhancing/adapting different already existing XML-based data formats combined under one roof the so called top level format (see Figure 2). These data formats are used on an “as-is” basis within their own specifications and are not branched for AutomationML needs. Logically AutomationML is partitioned in:

- description of the plant structure and communication systems expressed as a hierarchy of AutomationML objects and described by means of CAEX following IEC 62424,
- description of geometry and kinematics of the different AutomationML objects represented by means of COLLADA 1.4.1 and 1.5.0 (ISO/PAS 17506:2012),
- description of control related logic data of the different AutomationML objects represented means of PLCopen XML 2.0 and 2.0.1, and
- description of relations among AutomationML objects and references to information that is stored in documents outside the top level format.

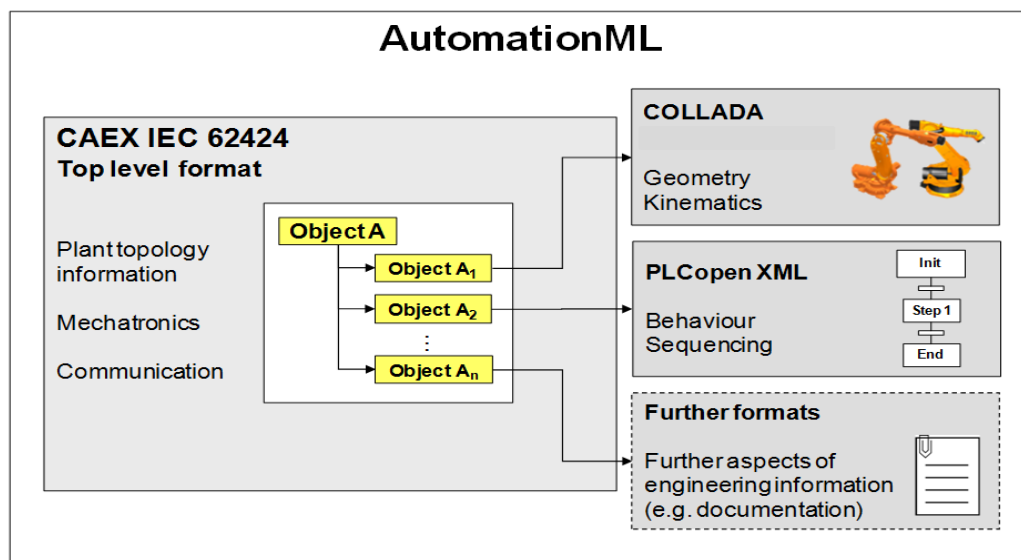


Figure 2 – AutomationML base structure

Due to the different aspects of AutomationML, the IEC 62714 series consists of different parts. Therby each part focuses on a different aspect of AutomationML:

- IEC 62714-1, Architecture and general requirements: This part specifies the general AutomationML architecture, the modeling of engineering data, classes, instances, relations, references, hierarchies, basic AutomationML libraries and extended AutomationML concepts. It is the basis of all future parts, and it provides mechanisms to reference other sub formats.
- IEC 62714-2, Role class libraries: This part is intended to specify additional AutomationML libraries.
- IEC 62714-3, Geometry and kinematics: This part is intended to specify the modeling of geometry and kinematics information.
- IEC 62714-4, Logic: This part is intended to specify the modeling of logics, sequencing, behavior and control related information.

Further parts may be added in the future in order to interconnect further data standards to AutomationML.

The basis of AutomationML is the application of CAEX as top level format and the definition of an appropriate CAEX profile fulfilling all relevant needs of AutomationML to model engineering information of production systems, to integrate the three data formats CAEX, COLLADA, and PLCopenXML, and to enable an extension if necessary in the future.

CAEX enables an object oriented approach (see Figure 3) where

- semantic of system objects can be specified using roles defined and collected in role class libraries,
- interfaces between system objects can be specified using interface classes defined and collected in interface class libraries,
- classes of system objects can be specified using system unit classes (SUC) defined and collected in system unit class libraries, and
- individual objects are modeled in an InstanceHierarchy (IH) as a hierarchy of InternalElements (IE) referencing
 - system unit classes they are derived from and
 - role classes defining its semantics and interface objects used to interlink objects among each other or with externally modeled information (e.g. COLLADA and PLCopenXML files).

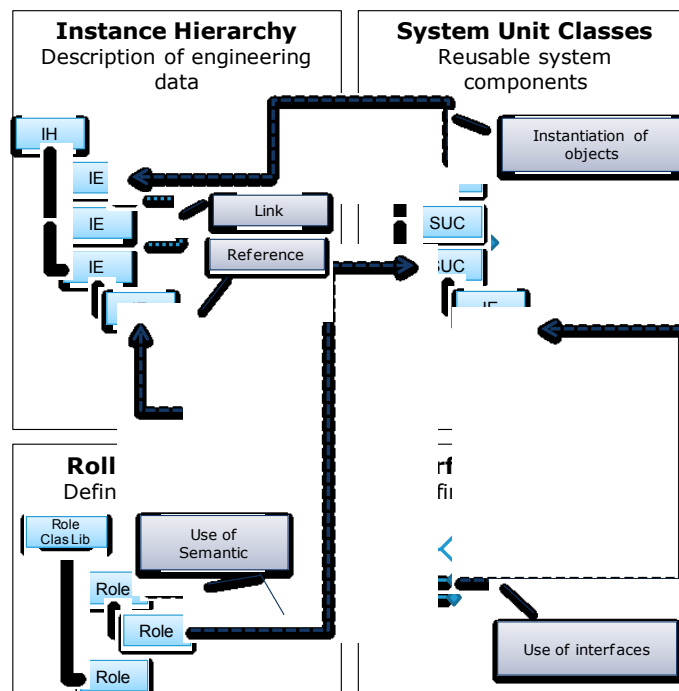


Figure 3 – AutomationML topology description architecture

The use of CAEX within AutomationML is described in detail in [1].

Pre-developed role class libraries applicable to derive necessary roles for semantic definition in individual application cases are provided in [2].

The integration of geometry and kinematic models stored in COLLADA and appropriate interface classes are detailed in [3].

The integration of behaviour models stored in PLCopen XML and appropriate interface classes are detailed in [4].

3.2 Capabilities for semantic integration

The mapping of incoming data points to the data model of the importing tool is a critical point during the import of data sets to be exchanged within engineering tools. Therefore the semantics of each data point related to the importing tool has to be specified.

Within the data import process of AutomationML based data the incoming data points are given within the InstanceHierarchy either as InternalElements or as attributes.

To identify the semantics of InternalElements AutomationML provides two main mechanisms: referencing of roles and referencing of SystemUnitClasses. For referencing of roles the sub-objects RoleRequirements and SupportedRoleClass are provided. They can contain the complete role name including the role path. For referencing a SystemUnitClass the attribute RefBaseSystemUnitPath can be used. This contains the complete name and path of the SystemUnitClass in an arbitrary SystemUnitClassLib.

For the representation of attribute semantics the RefSemantik sub-attribute of an attribute object can be used. It is given for each AutomationML attribute.

Fehler! Verweisquelle konnte nicht gefunden werden. shows all these means for semantic representation.

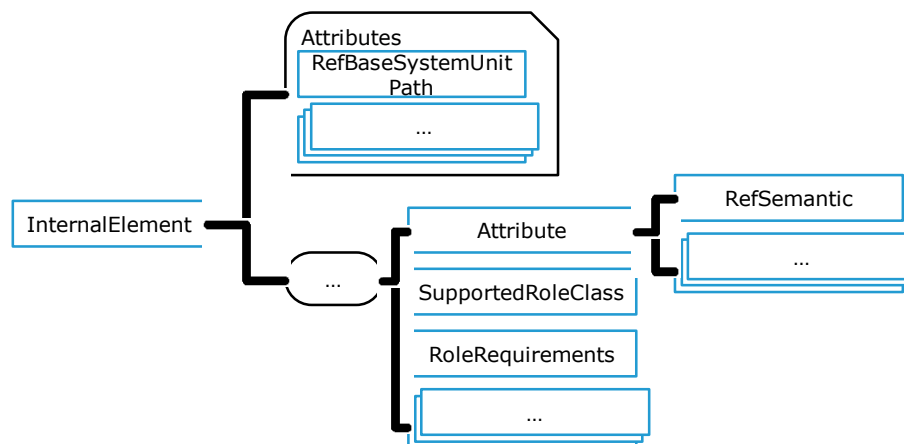


Figure 4 – Means for data semantic modelling in AutomationML

4 eCl@ss

eCl@ss is a hierarchical and semantic system for grouping materials, products and services according to a logical structure with a level detail that corresponds to the product-specific properties that can be described using properties according to ISO 13584 and IEC 61360. Products and services can be allocated to the four-stage, numeric eCl@ss class structure. Search terms and synonyms permit targeted sourcing of products and services within the classification. Feature lists with standardized properties and value tables enable accurate description and subsequent identification of products and services.

Several committees inspect each change request for correctness with regard to form and content to ensure the high quality of eCl@ss.

The content is inspected with regard to correctness and conformity to the underlying standards (e.g. ISO 13584, IEC 61360, DIN 4002).

eCl@ss is offered for download in several languages from <http://www.eclassdownload.com>

Detailed definitions of the structure elements can be found in <http://wiki.eclass.eu>.

4.1 Basic Representation

eCl@ss basic representation presents a version variant with all essential new contents in plain CSV format. In addition, the basic version is downloadable as an XML variant.

The new structural elements described in 4.3 are not part of the basic representation. A reference list with the structural elements of the Advanced Representation maps the attributes of the basic representation.

4.2 Advanced Representation

By introducing eCl@ss Advanced Representations (since eCl@ss 7.0), the eCl@ss-Association offers an ISO13584-32:2010 (ontoML) compliant data structure. The data structure contains the structural elements described under point 2. The result is the following structural arrangement:

The 4-tier class structure is associated with the new application class (AC). This, in turn, contains all relevant structural elements such as blocks and aspects.

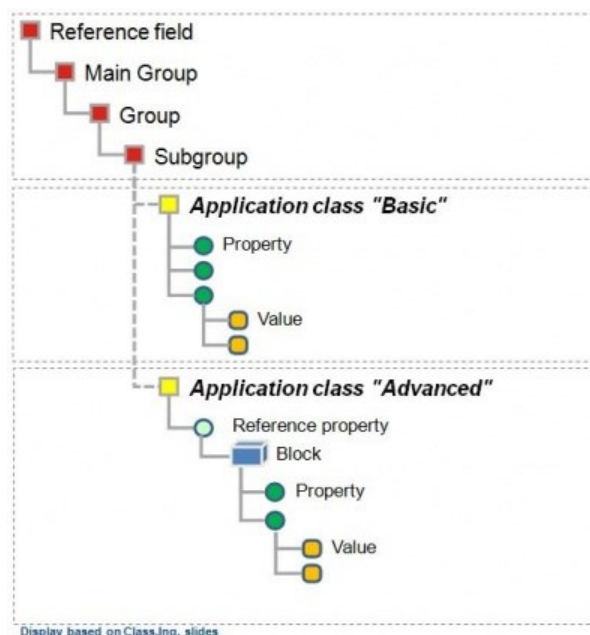


Figure 5 – eCl@ss classification data structure

Integration of both the PROLIST standard and CAx elements results in additional important data model extensions. Using “Cardinality” multiplication elements (refer to 4.3.3) as well as variant access of special blocks by means of “Polymorphism” (refer to 0) results in significant simplification.

Implementation of additional data type extensions (Level type and axis type) results in further simplifications. Consequently, a single data type combines physical-technical links originating from various attributes.

It should therefore be possible to transfer the complex data model in a data structure configured for that purpose.

That’s why eCI@ss uses OntoML rules defined in ISO 13584-32. These OntoML rules describe the form and arrangement of the structural elements and predefine how to interpret the OntoML in order to be able to process eCI@ss Advanced Representation from a data-technical point of view.

4.3 New structural elements of eCI@ss Advanced Representation

4.3.1 Block

A block is defined as the total of various properties under a single name and can be reused in different areas of the classification system. A structure like this may be used for detailed descriptions of components.

Using a car as an example, the block “wheel” would contain all attributes to describe the wheel: Rim diameter, tire size etc. Another block would then describe the “axle” properties. In addition, apart from attributes, a block can also contain additional sub-blocks with attributes and more blocks: In this example the “wheel” block is subdivided into “rim” and “tire”. In order to create a block, a so-called reference property has to be created in the Advanced Representation.

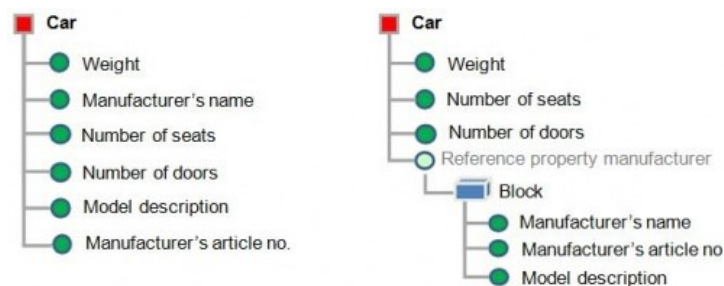


Figure 6 – Example block description

4.3.2 Aspect

An aspect is a special block variant that cannot be represented by cardinality or polymorphism and is located in the top level of a class. The advantage of an aspect as opposed to a block is the aspect’s universal use in a random number of classes without being a direct part of a class. Consequently, it is not subject to any product specific restrictions.

An aspect, therefore, does not describe the product specific property itself (like ordinary blocks and class attributes), but it includes attributes for that class under certain conditions or additional attributes for a class under certain view points.

An example is the aspect “Operating conditions”. It describes under which conditions the car is to be operated: Central Europe, the Arctic or desert. The resulting properties, however, such as minimum starting temperature or operational altitude are attributes of the “car” class and aspect elements.

4.3.3 Cardinality

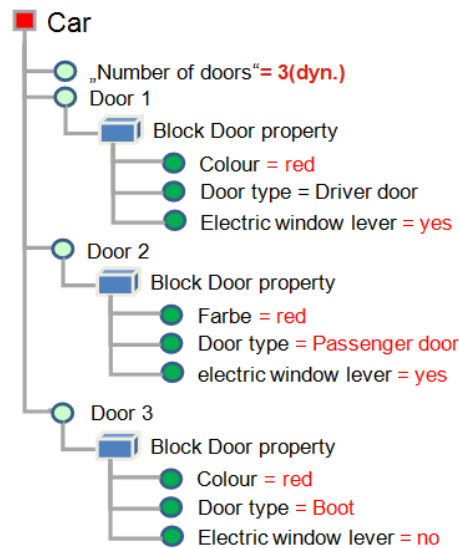


Figure 7 – Example for cardinality for properties

For structuring class attributes one can also use, apart from blocks, a so-called cardinality. “Cardinality” refers to the property allowing dynamic multiplication of a block within the scope of the property values to be managed.

In case of the “car” example, one could use “cardinality” to describe the doors. For instance, the attributes color, door type, and electric window levers describe the doors. A „door attributes” block combines these attributes, which can be accessed at random using the reference property “number of doors”. To describe a 3-door vehicle, the value “3” has to be set for the reference property “number of doors”. As a result, the “door attributes” block is accessed 3 times.

Within the contents of data description, cardinality is therefore a possibility to determine the number of identical blocks.

4.3.3.1 Polymorphism

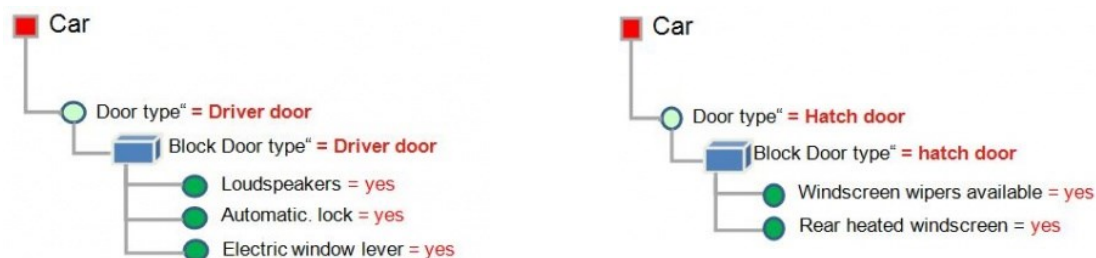


Figure 8 – Example for polymorphism in eCl@ss

Polymorphism implies that the block content is not assigned within a class but that only after an allocation of values to the attributes it is dynamically decided, which block content is actually required (only at this stage it is determined, data-technically, which block is to be selected from a number of blocks).

In the case of our “car” example, one could use polymorphism to describe the type of doors. The driver’s door attributes differ compared to the hatch attributes. For instance, the driver’s door could have the attributes, electric window lever, automatic lock, loudspeakers etc., whereas the hatch door would sooner have attributes such as windscreen wipers available, rear heated windscreen etc.

In the “wheel” example of the “car” class, you will find that the attributes for a wheel with double tires will be quite different from a wheel with single tires. Therefore, if values are to be allocated to a “wheel” block one will first have to decide which type of wheel is to be described: Single tire or double tire. Following the decision for “double tires”, the block “wheel with double tires” is accessed and related attributes made available (from the Data-model point of view, the block “wheel” is replaced by a specialization “wheel with double tires”).

The properties of this multiple (“poly”) substitutability (“morphism”) resulting from this special variant are therefore used to describe various details of a product structure keeping the number of total attributes manageable and free of redundancies.

4.3.4 Units

Each unit-associated property may express values (numbers) with regard to its basic unit and dosage unit.

Consequently, properties are used quite universally and do not require multiple definitions should the scale of data volumes differ considerably. In addition, common usage units can be used.

Convertibility of figures to various units is a prerequisite for using different units for a property. A property cannot have the unit kg/m³ (density) and kg/m² (surface density).

4.4 Export formats

Following the release of eCI@ss 7.0, the eCI@ss standard is available in two different export formats. Internationalization required amendments of ISO standards. Significant amendments involved both a distinction between ISO-standard conformant “closed” value lists and “open” proposal lists and the introduction of standard compliant measurement units.

4.4.1 CSV for Basic Representation

The usual CSV export format of the eCI@ss-standard includes eight CSV files, which Office programs such as MS-Excel and MS-Access can import and process. The attached ReadMe file describes the amendments, structure, contents, and associations of the CSV files.

4.4.2 eCI@ss XML for Basic and Advanced Representation

With introduction of eCI@ss 7.0, an automated eCI@ss XML format representation/generation of eCI@ss data is available. This export format bases on an ISO-standardized XML format for product data exchange. Related specifications are published in ISO 13584-32:2010 (ontoML). These specifications offer a uniform and comparable data structure for communications between machines. Consequently, two export formats are available for processing and implementing Basic Representation. Export of Advanced Representation is only available in eCI@ss XML-format.

4.5 Release – Update - Support

By means of eCI@ss 7.0, eCI@ss e.V. supports standard users with additional information making it possible to update from eCI@ss Release to Release more or less automatically. The result, a significant saving in data processing costs. Updating from 6.2 to 7.0 will make this possible for the first time. Below please find a description of the Release-Update-Files and Transaction-Update-Files made available by eCI@ss:

4.5.1 RUF

The Release-Update-File, abbreviated RUF, corresponds with common mapping tables but is far more complete and of a higher quality. Made available in CSV format it contains information with regard to release changes made in respect of classes and attributes. Move/Split/Join as well as previous and successive associations are retraceable by means of these files.

4.5.2 TUF

The Transaction-Update-file, abbreviated TUF, made available in XML-format, provides information as to whether previous attributes are still valid when using successive attributes. This allows a (partly) automatic update of evaluated data files.

4.6 IRDI

eCl@ss uses globally unique identifiers for every object included in the eCl@ss standard. This IRDI (International Registration Data Identifier) is based on the international standards ISO/IEC 11179-6, ISO/TS 29002 and ISO 6523(see Figure 9).

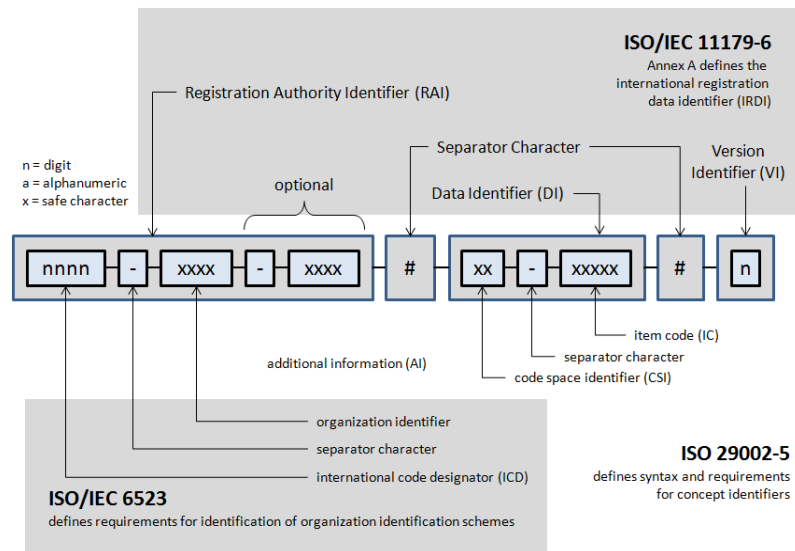


Figure 9 – ISO / IEC 6523 Definition for use of IRDI

The IRDI consists of several parts:

- an International Code Designator (ICD) according to ISO 6523, followed by an Organization Identifier (OI) that globally identifies eCl@ss as the publishing organization (eCl@ss: 0173, other registered organizations include ISO (0112), ODETTE (0177), SIEMENS (0175), GTIN (0160))
- a Code Space Identifier (CSI) that identifies the type of object (e.g. 01 for classification class, 02 for property etc.)
- a Concept Code (6-digit identifier, e.g. AAA123)
- a Version identifier (e.g. 001 for the first version of an object)

Table 2 defines the Code Space Identifiers (CSIs) used in eCI@ss XML:

Table 2 – Code Space Identifier According To ISO/TS 29002-5 (Restricted)

CSI	Category of administered item	Used in eCI@ss XML
CT	Concept type	-
TM	Term	-
DF	Definition	-
IM	Image	-
AB	Abbreviation	-
GS	Graphical symbol	-
TS	Textual symbol	--
LG	Language	-
OG	Organization	-
01	Class	YES
02	Property	YES
05	Unit of measure	YES
07	Property Value	YES
08	Currency	-
09	Data type	YES
11	Ontology	YES
Z2	Aspect of conversion	YES
Z3	Template	YES
Z4	Quantity	YES

Company Specific Identifiers

Certain ranges of (6 digit) identifiers (i.e. the concept code) are blocked for company-specific use:

- concept codes starting with X are blocked for user-specific structural elements
- concept codes starting with Y are blocked for manufacturer-specific structural elements
- concept codes starting with Z are blocked for eCI@ss-internal structural elements and shall not be used by eCI@ss users

eCI@ss will not use identifiers starting with X or Y. This guarantees that these company-specific structural elements will not be replaced by eCI@ss elements. Apart from the blocked concept codes, a company may simply use a different ICD, so that all concept codes can be used, but they are distinct with the help of an ICD that is not 0173 (for eCI@ss).

5 Use cases and considered data exchange structures

Within the engineering of production systems several activities require a unique identification of data objects. Nevertheless, not all possible cases are relevant for the application of data exchange formats. Within the following section the relevant use cases necessary for the application of unique identification of data objects during the data exchange between engineering tools in the production system engineering process as well as the relevant information sets within them are named.

Table 3 – Overview Use Cases

Name	Category	Short description
Simple semantic identification	technical	Enrich AutomationML objects or attributes with semantic based on eCI@ss catalogue or properties.
Semantic identification including property guarantees	technical	Enrich AutomationML objects with semantic based on eCI@ss catalogue and guarantee existence of property information.
Semantic identification including structure guarantees	technical	Enrich AutomationML objects with semantic based on eCI@ss catalogue and guarantee existence of object substructures.
Exchange of semantically unique instance data	exchange	Semantically unique exchange of engineering data between two tools
Encode data semantics	exchange	Semantically unique encoding of engineering information in case of information export from a tool
Decode data Semantics	exchange	Semantically unique decoding of engineering information in case of information import to a tool
ECl@ss dictionary exchange	exchange	Create, maintain and provide AutomationML libraries encoding ECl@ss dictionary required to define/identify/ validate the semantic of AutomationML objects
Receive eCI@ss dictionary	exchange	Receiving ECl@ss dictionary and creation of AutomationML libraries encoding ECl@ss dictionary
Deliver eCI@ss dictionary	exchange	Enrich the ECl@ss dictionary based on request
Development and use of semantically unique component libraries	application	Creation and use of SystemUnitClass libraries useable within different engineering processes containing AutomationML objects with a unique object and property semantics
Lossless exchange between system configurator and CAX tool	application	Exchange of engineering results between engineering tools containing semantically unique identifiable AutomationML objects
Construction validation	application	Validation of applicability of selected system components based on uniquely identifiable component properties

5.1 Technical use cases

The first set of use cases is technically motivated. It considers the degree of semantic identification of objects resulting in different necessary technical solutions for the semantic representation.

5.1.1 Use case – simple semantic identification

Following Figure 1 basic requirement for the data exchange is the capability to identify the semantics of an object to enable the correct mapping of the object to object types of the data model of the data receiving tool. This semantic identification simply has to describe the meaning of the object within a data model, i.e. the object type. Examples of such a simple semantic identification are the indication, that a data object represents an inductive sensor, a special drive type or a complete industrial robot or a chemical reactor.

As there are several semantic definitions available in the different industries as well as industry crossing semantic definitions it shall not be the responsibility of AutomationML to define a new semantic catalogue. Instead existing catalogues shall be applied for the semantic representation.

This use case will provide the following requirements to AutomationML:

- AutomationML shall enable the integration of a semantic reference to an external defined semantic definition.
- AutomationML shall enable the identification of a used semantic catalogue.
- AutomationML shall enable the unique identification of the catalogue object relevant for the semantic identification of the exchanged data object.

5.1.2 Use case – semantic identification of properties

In several cases of semantic identification it is not sufficient to simply represent the object type but to provide guarantees about available information within the provided data object. The data receiving tool has to map the information provided within a data object in a structured way to the internal data model including the mapping of properties, attributes, etc. Therefore, it is necessary to extend the simple object identification by guarantees about available information within the provided data object as well as its representation (naming, data type, etc.) within the provided data object.

As an example an inductive sensor or a special drive type can have special object class related information like a vendor identification, a material description, an energy consumption etc.. The industrial robot can provide information about its possible work space and the chemical reactor information about the maximal volume.

As in the case of a simple semantic identification there are catalogues available specifying relevant properties and attributes for the object classes. These catalogues shall be applicable for semantic identification including property guarantees in AutomationML.

This use case will provide the following requirements to AutomationML:

- AutomationML shall enable the integration of a semantic reference to an external defined semantic definition.
- AutomationML shall enable the identification of a used semantic catalogue.
- AutomationML shall enable the unique identification of the catalogue object relevant for the semantic identification of the exchanged data object.
- AutomationML shall enable the definition and identification of object properties (attributes) by referencing the relevant properties (attributes) in the used catalogue.

5.1.3 Use case – semantic identification of structures

In addition to the use cases identified above the semantic representation may provide guarantees about the provided data substructure available within the provided data object and the relations between the data objects. The data receiving tool has to map the substructure and its relations in a structured way to the internal data model. Therefore, it is necessary to extend the object identification by guarantees about available substructure and relations within the provided data object as well as its representation (naming, identification, etc.) within the provided data object.

As an example a special drive type can contain a break with its own properties. The industrial robot usually consists of a set of axis all interrelated in a special way. Finally the chemical reactor consists of a set of mechanical parts and automation / process equipments like electrical drive driven stirrers, heaters, temperature sensors, etc.. In addition their data objects may be interrelated. For example, if a chemical reactor will have a heater it usually will also have a temperature sensor. This needs to be represented in the guarantees.

As in the case of a simple semantic identification there are catalogues available specifying relevant structures and its properties / interrelations for the object classes. These catalogues shall be applicable for semantic identification including structure guarantees in AutomationML.

This use case will provide the following requirements to AutomationML:

- AutomationML shall enable the integration of a semantic reference to an external defined semantic definition.

- AutomationML shall enable the identification of a used semantic catalogue.
- AutomationML shall enable the unique identification of the catalogue object relevant for the semantic identification of the exchanged data object.
- AutomationML shall enable the definition and identification of object properties (attributes) by referencing the relevant properties (attributes) in the used catalogue.
- AutomationML shall enable the definition and identification of object structures (subobject and its dependencies to properties) by referencing the relevant structures in the used catalogue.

5.2 Exchange use cases

The second set of use cases is motivated by the engineering process. The engineering of production systems contains several engineering activities from different engineering domains. Figure 10 represents an example set of engineering activities relevant within the general engineering process of production systems.

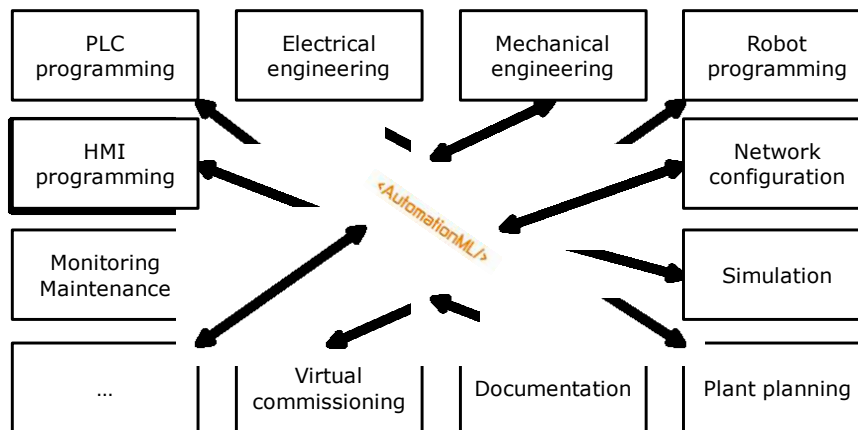


Figure 10 – General engineering activities embedded within production system engineering (subset)

Within the named engineering activities engineering tools like the following (but not limited to) will have a relevant impact:

- Plant planning tool
- Mechanical engineering tool (MCAD)
- Electrical engineering tool (ECAD)
- PLC programming tool
- Robot programming tool
- HMI programming tool
- Network configuration tool
- Simulation tool
- Virtual commissioning tool
- Monitoring tool
- Maintenance tool
- Documentation tool

For the data exchange between these tools a system of use cases is relevant providing the capabilities to semantically unique exchange data points.

The central use case is the use case “exchange of semantically unique instance data”. This use case covers the exchange of information between two engineering tools. This use case can be extended to use cases like “development and use of semantically unique component libraries”, “lossless exchange between system configurator and CAx tool”, and “construction validation”.

Within the use case “exchange of semantically unique instance data” three supporting use cases are embedded. These are the use case “encode data semantics” for syntactically and semantically unique encoding of data objects exchanged, the use case “decode data semantics” for syntactically and semantically unique decoding of data objects exchanged, and the use case “eCl@ss dictionary exchange” required to enable the semantically unique identification of data points.

The later use case consists of two sub use cases for delivering and receiving an eCl@ss dictionary.

Within the use cases three main actors are involved. These are the sending and the receiving engineer involved and driving the use case “exchange of semantically unique instance data” and the eCl@ss association involved in the use case “deliver eCl@ss dictionary”.

The named use case structure is depicted in Figure 11.

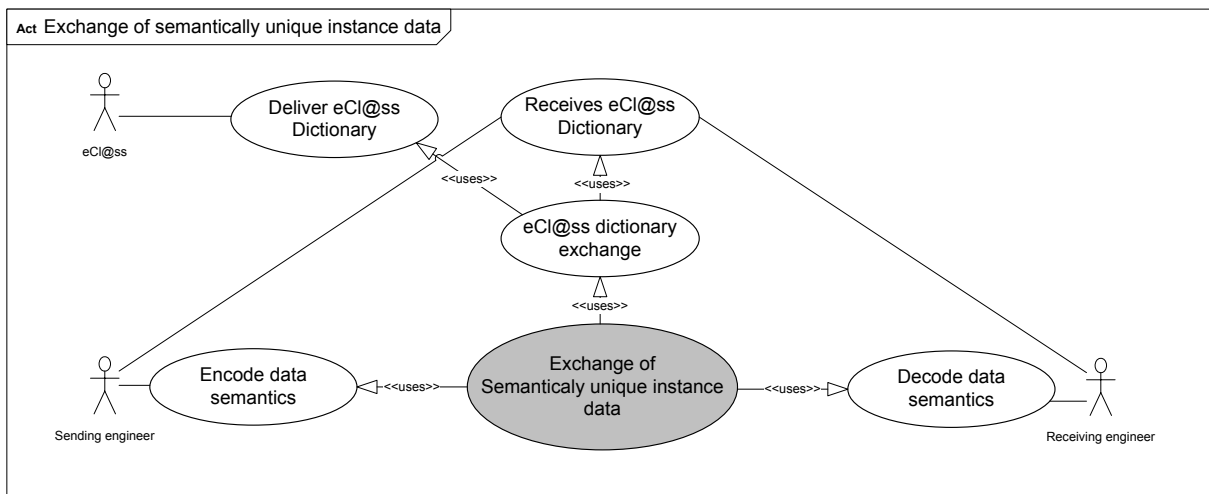


Figure 11 – System of relevant use cases for semantically unique exchange of data points

In the following the different named use cases are characterized in detail. Prior some main assumptions are named.

5.2.1 Assumptions for considered use cases

In the following only the actors sending and the receiving engineer and eCl@ss association are considered. Within the other AutomationML and eCl@ss specifications mainly further actors are identified. Without definition of completeness it is assumed, that the actors sending and the receiving engineer can be engineers of different engineering disciplines. For example these engineers can be:

- Standard Plant Planner
- New production planner (Mechanics)
- New production planner (Electrics)
- Series production planner (Mechanics)
- Hall layout planner
- Mechanical Simulation Engineers / Offline Programmer
- Robot Programmer
- Mechanical Design Engineer
- Electrical Design Engineer
- PLC Programmer
- HMI Programmer

The actor eCI@ss association subsumes all engineers, companies, and other legal entities involved in the standardization process of eCI@ss dictionaries. It is out of scope of this document to further distinguish the different roles and actions of them.

In addition the following use cases will consider only a sending and a receiving tool. Without definition of completeness it is assumed, that the sending and the receiving tool can be engineering tools of different engineering disciplines. For example these tools can be:

- Computer Aided Mechanical Design tools (M-CAD)
- Simulation tools
- Computer Aided Electrical Engineering tools (CAE) and
- Control Programming tools.

5.2.2 Use case - exchange of semantically unique instance data

The main goal of this use case is the semantically unique exchange of engineering data between two engineering tools. Thereby, two engineers shall be able to select a part of the project data available in a sending engineering tool, assign them with a unique semantic identification, transfer them to a receiving engineering tool, identify the semantics of data objects, and, finally, integrate them in the project data of the receiving tool.

The critical point within this use case is the implementation of the necessary mapping of the data models of the sending and receiving tool as indicated in Section 1. This mapping shall be established by utilization of the same semantic dictionary within the sending and the receiving tool.

Within this use case only a sending and a receiving engineer as well as a sending and a receiving engineering tool are involved. They will execute the following normal flow of activities also given in Figure 12.

1. The sending engineer will select the subset of the project data of the sending engineering tool for data exchange.
2. The sending engineer will utilize the set of AutomationML semantic libraries establishing the eCI@ss dictionary for data object encoding (assigning a unique object semantics to each relevant data object).
3. The sending engineer will store the decoded data within an AutomationML project.
4. The receiving engineer will read the stored AutomationML project.
5. The receiving engineer will utilize the set of AutomationML semantic libraries establishing the eCI@ss dictionary for data object decoding (identifying unique object semantics to each relevant data object).
6. The receiving engineer will integrate the read data objects within the project data of the receiving tool.

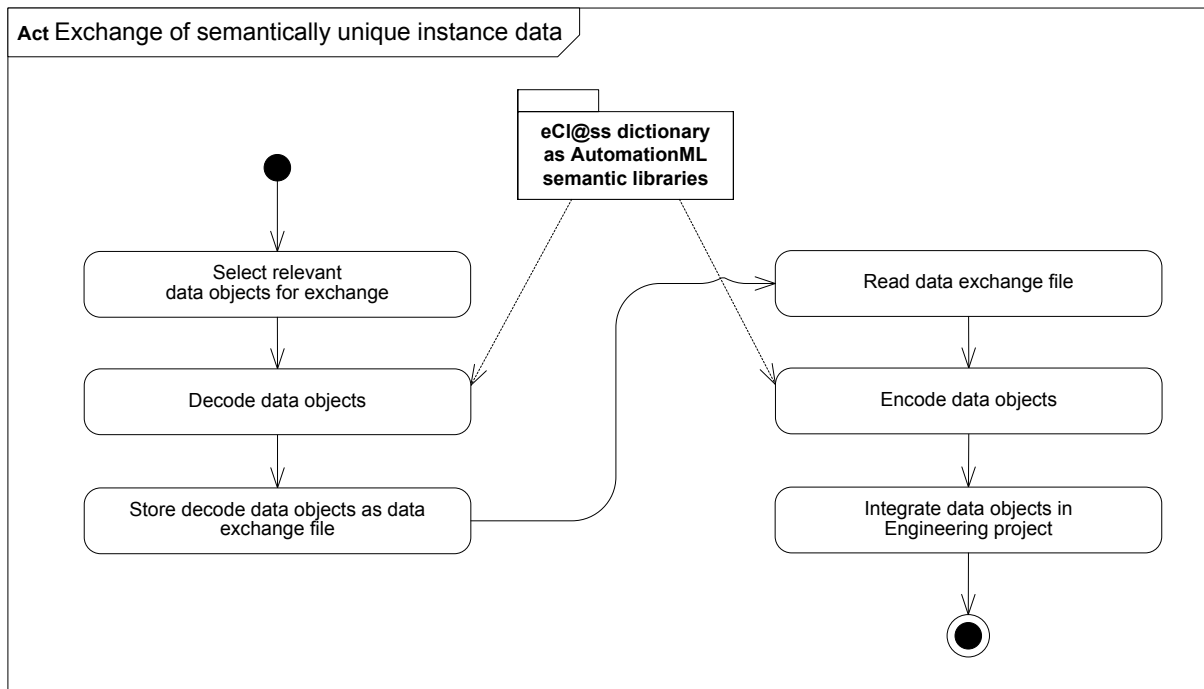


Figure 12 – Activity structure of use case "exchange of semantically unique instance data"

Within this use case the project data of the sending engineering tool as well as the set of AutomationML semantic libraries establishing the eCl@ss dictionary have to be seen as input data while the project data of the receiving engineering tool are output data.

This use case will provide the following requirements to AutomationML:

- AutomationML shall define a methodology to attach a unique identification of object semantics to each relevant data object (InternalElements, SystemUnitClasses, Interfaces, Attributes).
- AutomationML shall define a methodology for creation of a set of AutomationML semantic libraries establishing the eCl@ss dictionary based on the eCl@ss dictionary specifications.

In case of application an eCl@ss dictionary following of eCl@ss basics this use case requires the implementation of technical use cases

- Simple semantic identification and
- Semantic identification including property guarantees.

In case of application of an eCl@ss dictionary following eCl@ss advanced this use case requires the implementation of technical use cases

- Simple semantic identification,
- Semantic identification including property guarantees, and
- Semantic identification including structure guarantees.

5.2.3 Use case – encode data semantics

One part of the use case "exchange of semantically unique instance data" is the encoding of selected engineering data within the sending engineering tool. The execution of this encoding process is the main goal of this use case. Thus, this use case covers the automatic / semi-automatic association of a unique semantics to each relevant data object (InternalElements, SystemUnitClasses, Interfaces, Attributes) within the sending engineering tool.

The critical point within this use case is the assignment process of semantics to data objects and, thereby, the mapping of the data model of the sending tool to the semantic dictionary.

Within this use case only the sending engineer and the sending tool are involved. They will execute the following normal flow of activities also given in Figure 13. This normal flow of activities assumes, that the set of engineering data to be exchanged has been selected previously.

1. The sending engineer will select the relevant AutomationML semantic library out of the set of AutomationML semantic libraries establishing the eCI@ss dictionary for semantic mapping.
2. The sending engineering tool executes for each data object to be exchanged:
 - a. Identify the semantics of the data object within the data model of the sending tool.
 - b. Select the corresponding unique semantic identification out of the selected AutomationML semantic library.
 - c. Assign selected unique semantic identification to selected data object.

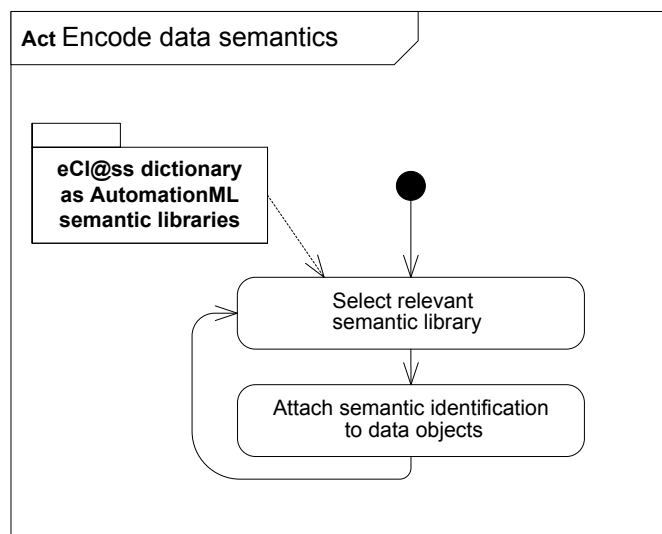


Figure 13 – Activity structure of use case "Encode data semantics"

Within this use case the selected part of project data of the sending engineering tool to be exchanged as well as the set of AutomationML semantic libraries establishing the eCI@ss dictionary have to be seen as input data while selected part of project data of the sending engineering tool to be exchanged with assigned AutomationML semantic references are output data.

This use case will provide the following requirements to AutomationML:

- AutomationML shall define a methodology to enable a one to one mapping of sending tool data object semantics to a unique semantic identification of the AutomationML semantic libraries establishing the eCI@ss dictionary.

In case of application of an eCI@ss dictionary following eCI@ss basics this use case requires the implementation of technical use cases

- Simple semantic identification and
- Semantic identification including property guarantees.

In case of application of an eCI@ss dictionary following eCI@ss advanced this use case requires the implementation of technical use cases

- Simple semantic identification,
- Semantic identification including property guarantees, and

- Semantic identification including structure guarantees.

5.2.4 Use case – decode data semantics

One part of the use case "Exchange of semantically unique instance data" is the decoding of exchanged engineering data within the receiving engineering tool. The execution of this decoding process is the main goal of this use case. Thus, this use case covers the automatic / semi-automatic exploitation of the assigned unique semantics to each relevant data object (InternalElements, SystemUnitClasses, Interfaces, Attributes) within the receiving engineering tool.

The critical point within this use case is the evaluation process of semantics of received data objects and, thereby, the mapping of the semantic dictionary to the data model of the receiving tool.

Within this use case only the receiving engineer and the receiving tool are involved. They will execute the following normal flow of activities also given in Figure 14. This normal flow of activities assumes, that the set of engineering data to be exchanged has been stored as an AutomationML project previously and can be read by the receiving tool.

1. The receiving engineer will select the relevant AutomationML semantic library out of the set of AutomationML semantic libraries establishing the eCI@ss dictionary for semantic mapping.
2. The receiving engineering tool executes for each data object read out of the provided AutomationML project:
 - a. Identify the semantics of the data object within the received AutomationML project following the selected AutomationML semantic library.
 - b. Select the corresponding object semantic within the data model of the receiving engineering tool.
 - c. Assign selected semantics to selected data object.

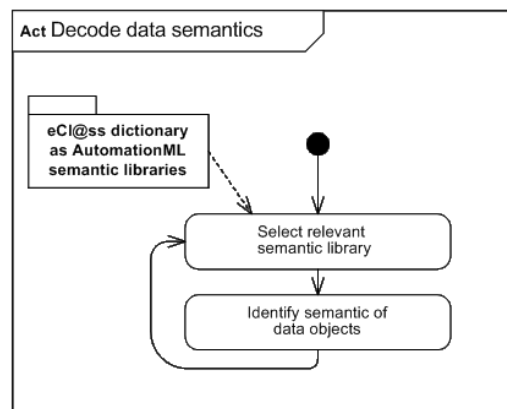


Figure 14 – Activity structure of use case "decode data semantics"

Within this use case the AutomationML project with the exchanged data objects as well as the set of AutomationML semantic libraries establishing the eCI@ss dictionary have to be seen as input data while exchanged data objects with assigned object semantics of the receiving engineering tool are output data.

This use case will provide the following requirements to AutomationML:

- AutomationML shall define a methodology to enable a one to one mapping of receiving tool data object semantics to a unique semantic identification of the AutomationML semantic libraries establishing the eCI@ss dictionary.

In case of application of an eCI@ss dictionary following eCI@ss basics this use case requires the implementation of technical use cases

- Simple semantic identification and
- Semantic identification including property guarantees.

In case of application an eCI@ss dictionary following of eCI@ss advanced this use case requires the implementation of technical use cases

- Simple semantic identification,
- Semantic identification including property guarantees, and
- Semantic identification including structure guaranties.

5.2.5 Use case – eCI@ss dictionary exchange

One part of the use case "exchange of semantically unique instance data" is the exchange of the eCI@ss dictionary required for the creation of the set of AutomationML semantic libraries establishing the eCI@ss dictionary. Thus, this use case covers the process of acquiring the necessary eCI@ss dictionary, its automatic translation to the set of AutomationML semantic libraries, and its provision to the sending and receiving engineer for use within the data exchange process for semantic mapping.

Within this use case the eCI@ss actors, the sending and the receiving engineer and the sending and the receiving tools are involved. They will execute the following normal flow of activities also given in Figure 15.

1. Based on request the eCI@ss actor is creating/maintaining and providing the eCI@ss dictionary.
2. The sending and/or receiving engineer is translating the eCI@ss dictionary to the set of AutomationML semantic libraries establishing the eCI@ss dictionary for semantic mapping.

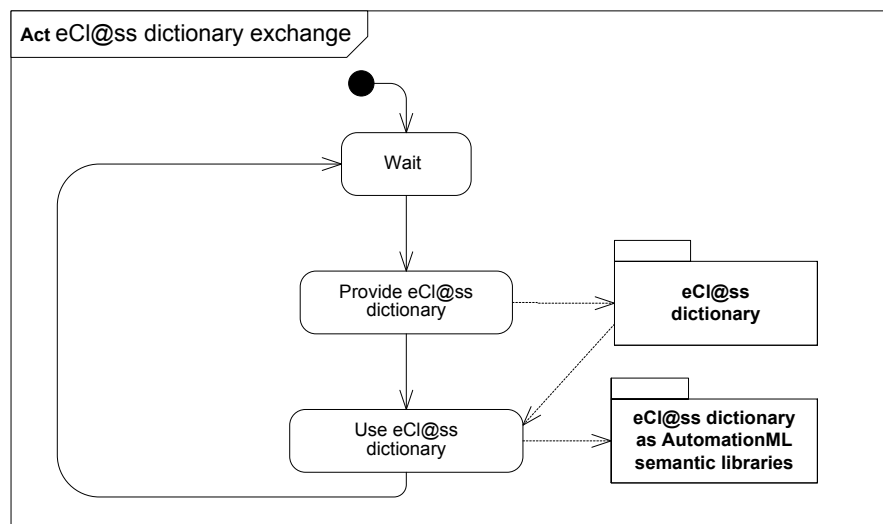


Figure 15 – Activity structure of use case "eCI@ss dictionary exchange"

Within this use case the request for eCI@ss dictionary provision is seen as input data while the provided set of AutomationML semantic libraries establishing the eCI@ss dictionary is considered as output data.

This use case will provide the following requirements to AutomationML:

- AutomationML shall define a methodology for automatic creation of a set of AutomationML semantic libraries establishing the eCI@ss dictionary based on the eCI@ss dictionary specifications.

In case of application of an eCI@ss dictionary following eCI@ss basics this use case requires the implementation of technical use cases

- Simple semantic identification and
- Semantic identification including property guarantees.

In case of application an eCI@ss dictionary following of eCI@ss advanced this use case requires the implementation of technical use cases

- Simple semantic identification,
- Semantic identification including property guarantees, and
- Semantic identification including structure guarantees.

5.2.6 Use case – receive eCI@ss dictionary

One part of the use case "eCI@ss dictionary exchange" is the use of the eCI@ss dictionary for creation of the set of AutomationML semantic libraries establishing the eCI@ss dictionary. Thus, this use case covers the process of automatic translation of the eCI@ss dictionary to the set of AutomationML semantic libraries, and its storing as AutomationML data objects.

Within this use case the sending and the receiving engineer and the sending and the receiving tools are involved. They will execute the following normal flow of activities also given in Figure 16. This use case, assumes that the eCI@ss dictionary is available.

1. The published eCI@ss dictionary is read.
2. The sending and/or receiving engineer is translating the eCI@ss dictionary to the set of AutomationML semantic libraries establishing the eCI@ss dictionary for semantic mapping.

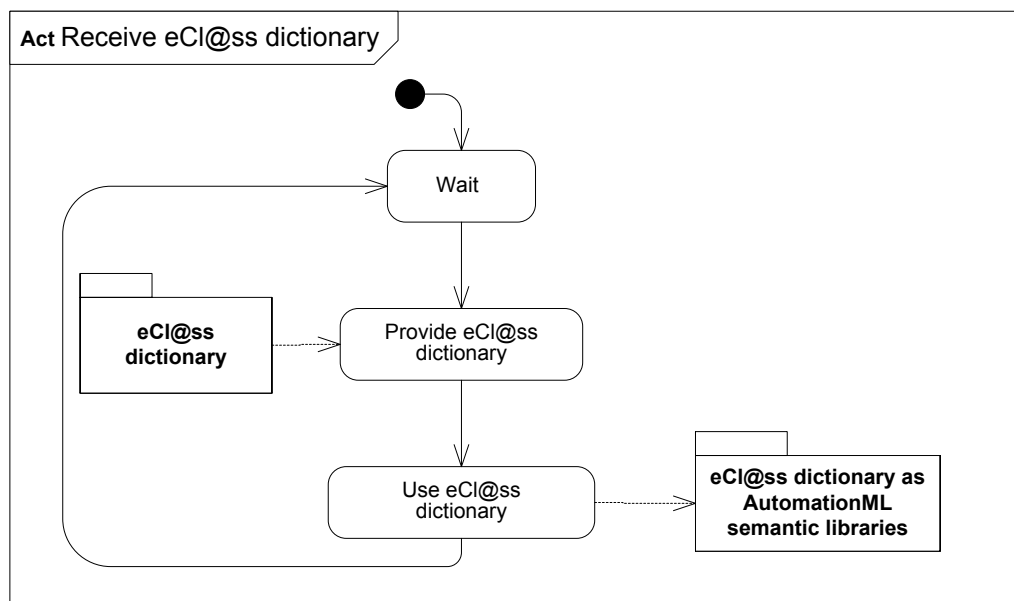


Figure 16 – Activity structure of use case "receive eCI@ss dictionary"

Within this use case the published eCI@ss dictionary is seen as input data while the provided set of AutomationML semantic libraries establishing the eCI@ss dictionary is considered as output data.

This use case will provide the following requirements to AutomationML:

- AutomationML shall define a methodology for automatic creation of a set of AutomationML semantic libraries establishing the eCI@ss dictionary based on the eCI@ss dictionary specifications.

In case of application of an eCI@ss dictionary following eCI@ss basics this use case requires the implementation of technical use cases

- Simple semantic identification and
- Semantic identification including property guarantees.

In case of application of an eCI@ss dictionary following eCI@ss advanced this use case requires the implementation of technical use cases

- Simple semantic identification,
- Semantic identification including property guarantees, and
- Semantic identification including structure guarantees.

5.2.7 Use case – deliver eCI@ss dictionary

One part of the use case "eCI@ss dictionary exchange" is the provision of the eCI@ss dictionary by the eCI@ss actor. Thus, this use case covers this provision process.

Within this use case only the eCI@ss actor is involved. He will execute the following normal flow of activities also given in Figure 17.

1. Based on request the eCI@ss actor is either defining new components of the eCI@ss dictionary or maintains existing ones.
2. The eCI@ss actor is publishing the eCI@ss dictionary following the usual publication process of eCI@ss association.

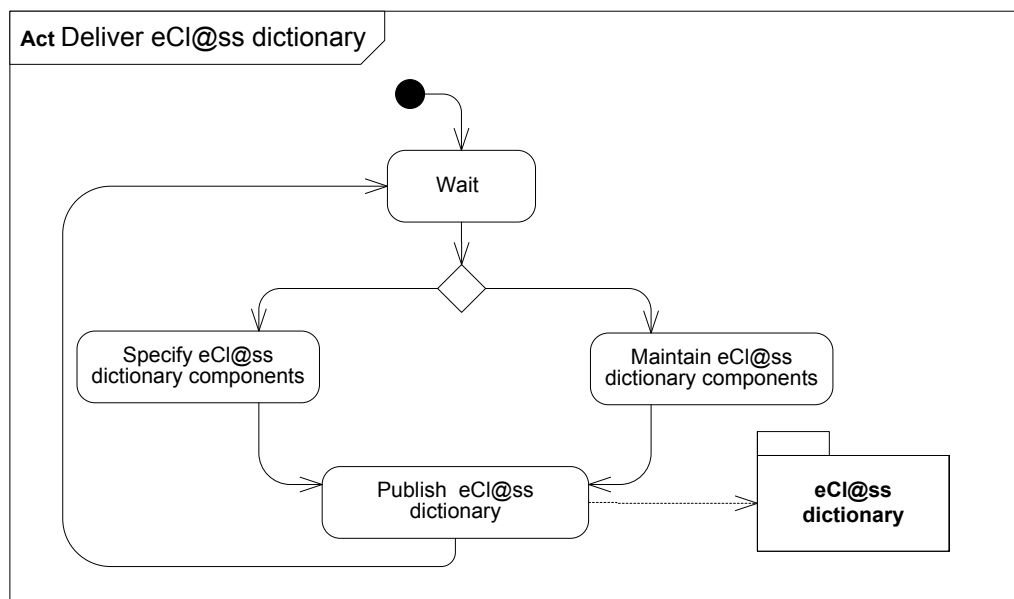


Figure 17 – Activity structure of use case "deliver eCI@ss dictionary"

Within this use case the published eCI@ss dictionary is seen as output data while there are no input data.

This use case will provide no requirements to AutomationML.

Note: AutomationML will **not** automatically provide access to eCI@ss dictionaries. The access and use rights for eCI@ss dictionaries and all other properties of eCI@ss association have to be purchased or acquired in further ways from eCI@ss association.

5.3 Application use cases

The use cases of the third set are motivated by tools, task and application, that uses the technical and exchange use cases.

Act Semantically unique exchange of data points

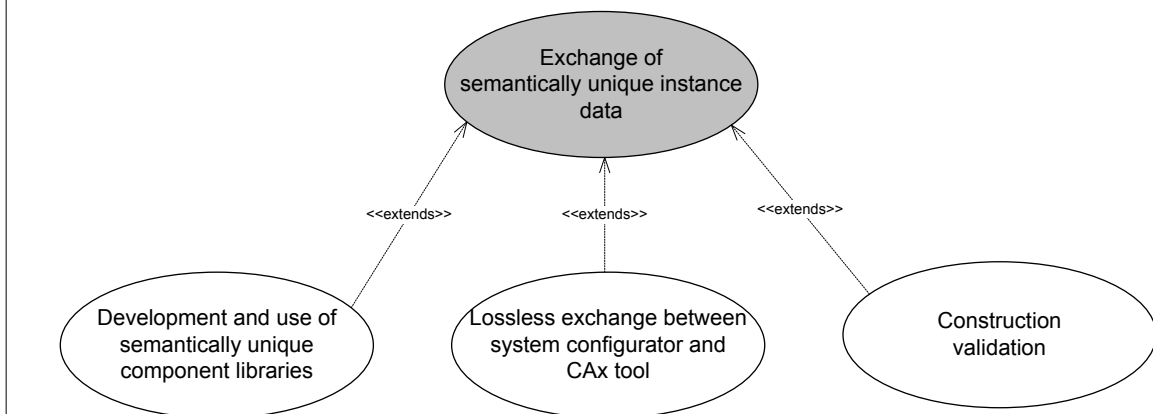


Figure 18 – System of relevant use cases for semantically unique exchange of data points

5.3.1 Use case - development and use of semantically unique component libraries

Within the engineering process of production systems it shall be possible to uniquely identify the semantics of data points given within a library of plant components. This unique identification of the semantics is required on different levels of detail including the semantics of a plant component (special type of drive, special type of robot, etc.), the semantics of its internal engineering artifacts (mechanical construction, electrical wiring, ...) and the semantics of properties / parameters / attributes (size in 3D coordinates, weight, ...). The unique identification of semantics has to be created by the library element creator and shall be exploited by all other engineers.

Hence, the use case “development and use of semantically unique component libraries” is an extension of the use case “exchange of semantically unique instance data”. Here the exchanged instance data are plant components without/out of a component library.

Within this use case the sending engineer and the receiving engineer are involved while the sending engineer is usually a component designer and while the receiving engineers can be new production planner (Mechanics), new production planner (Electrics), series production planner (Mechanics), hall layout planner, mechanical simulation engineers / offline programmer, robot programmer, mechanical design engineer, electrical design engineer, PLC programmer, HMI programmer and others. The involved tools will cover the complete range of tool named in section 5.2.1. They will execute the following normal flow of activities.

1. A library element is created and attached with a semantic identification.
2. An sending engineer is creating and engineering artifacts related to the library element. It adds the engineering artifact semantics.
3. An sending engineer is specifying a property / parameter / attribute related to the library element. It adds the property / parameter / attribute related semantics.
4. Repeat 2. and 3 as often as necessary.
5. The library element is stored in the library.
6. A receiving engineer is reading the library element from the library and is able to identify semantics of the library element, its engineering artifacts and its properties / parameters/ attributes.

The described activity sequence is depicted in Figure 19.

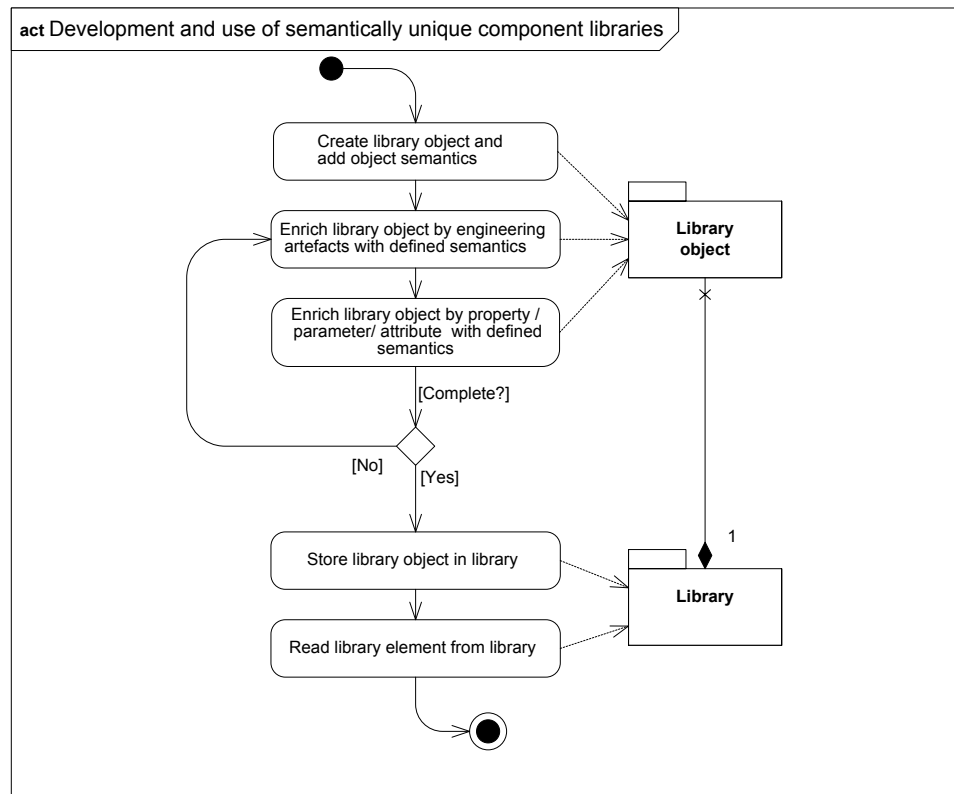


Figure 19 – Activity structure of use case "development and use of semantically unique component libraries"

5.3.2 Use case - lossless exchange between system configurator and CAx tool

Within the early engineering phases of production systems system components are designed by combining sets of sub-components out of component libraries by configuration tools and the provision of this configuration to a subsequent CAx tool. Examples of such engineering activities are

- the combination of manufacturing resources like welding cells and mounting stations following a given assembling sequence for body work in car manufacturing industry and the transmission of the developed structure to a plant simulation tool,
- the combination of electrical wiring components on mounting rails for control cabinet manufacturing and the transmission of the developed structure to a ECAD tool, and
- the combination of IEC 6113-3 Function blocks following a behaviour specification for PLC programming and its subsequent transfer to a PLC programming software.

The critical points in this process are the unique identification of objects selected in the configuration step, the modelling and transfer of relation information expressing the dependencies / orders / sequences / etc. between the selected components, and the expression of overall information characterising the complete set of combined objects as production system components.

Figure 20 provides an overview over this use case and its relation to the other use cases described.

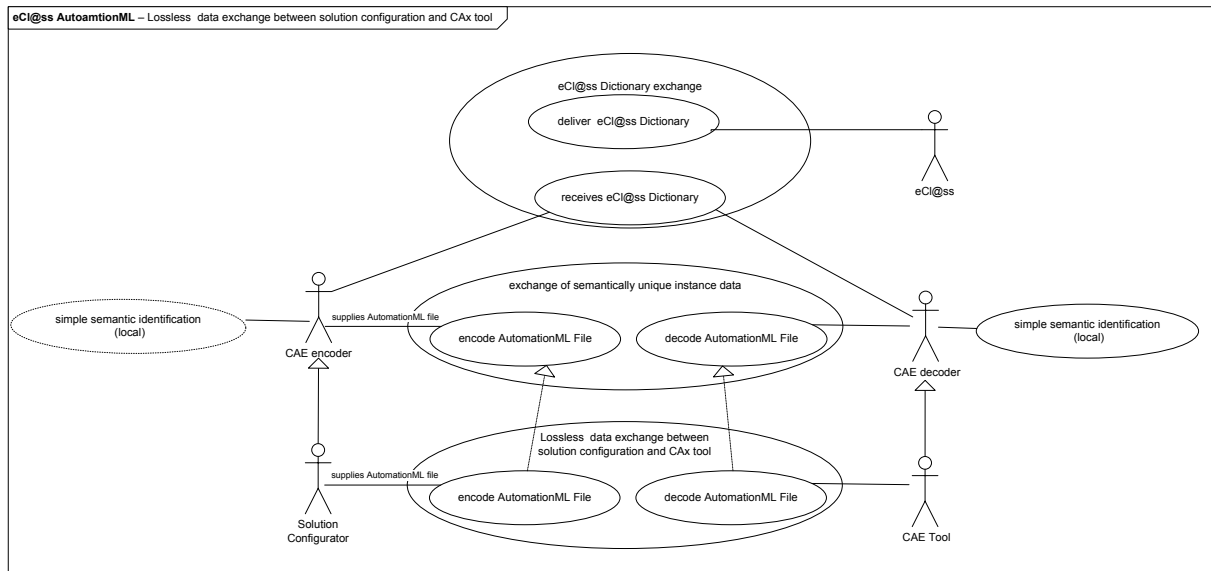


Figure 20 – Structure of use case „lossless exchange between system configuration tool and CAx tool“

Within this use case the sending engineer usually in the role as solution configurator and the receiving engineer in the role as CAX engineer as well as the configuration tool and the CAX tool are involved. They will execute the following normal flow of activities.

1. The solution configurator is collecting all necessary components for the solution
2. The solution configurator specifies the overall system structure out of the components and its describing properties
3. The solution configurator encodes the description to a data transfer file.
4. The CAX engineer decodes the description from the data transfer file.
5. The CAX engineer exploits the system description within his work.

The described activity sequence is depicted in Figure 21.

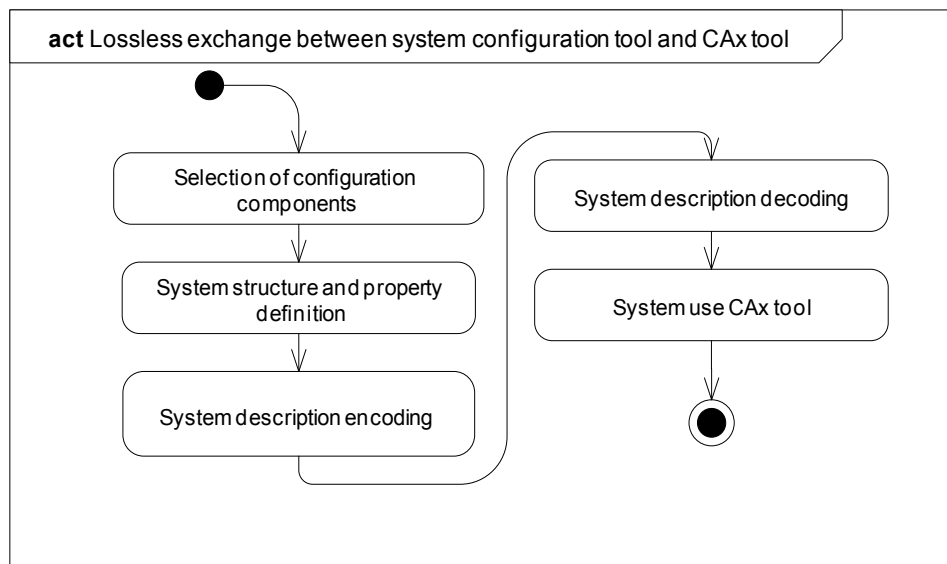


Figure 21 – Activity structure of use case „lossless exchange between system configuration tool and CAx tool“

Within this use case the set of described components with its properties is seen as input data while the complete system configuration is seen as output data.

This use case will provide the following requirements to AutomationML:

- Objects shall have a reference to eCI@ss dictionary (IRDI)
- Properties shall have a reference to eCI@ss dictionary (IRDI)

5.3.3 Use case - construction validation

Within engineering processes of production systems in various situations engineers have to select appropriate system components / devices / etc. following given requirements or to decide if a given design / construction fulfills a set of requirements. Examples of such engineering activities can be

- the selection of an appropriate drive for a drive chain from a drive vendor library,
- the identification of appropriate process parameter sets for a chemical reaction from a given reaction process library, or
- the validation of mechanical properties of an assembly.

Assuming a given list of requirements and a system description, then in all cases an automatic or manual validation procedure shall entail exactly one of the two propositions: “the described system fulfills the requirements” or “the described system does not fulfill the requirements”.

For this purpose the structured, static and deterministic properties are exploited. They are defined as requirements prior to the system design / engineering process and validated based on system descriptions after the system design / engineering process.

To evaluate the requirement properties the different information necessary to evaluate them have to be uniquely identifiable and comparable. Those identifiers have to be used in order to link construction elements to requirements. Thus each requirement may be considered within validation procedures, which have the same semantics on the requirements and construction side.

Figure 22 provides an overview for this use case and its relation to the other use cases described.

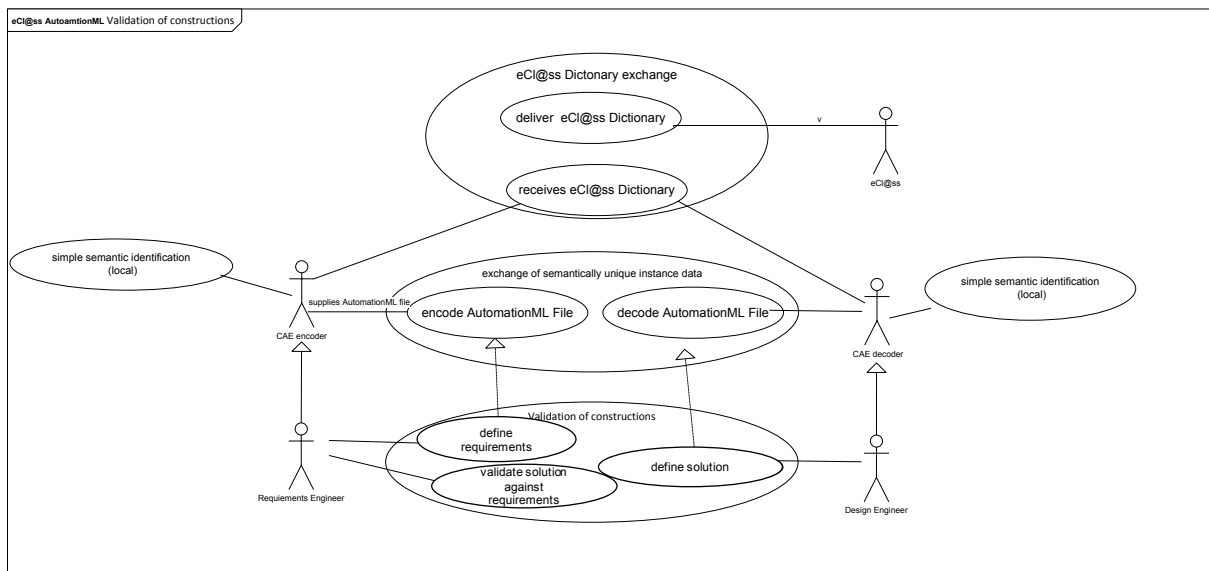


Figure 22 – Structure of use case “validation of construction”

Within this use case the requirements engineer, the design engineer, the validation engineer and the validation system are involved. They will execute the following normal flow of activities also given in Figure 22.

1. A requirements engineer defines types of system components and supplements those type descriptions with required attributes and sometimes with sub-component

descriptions. For each attribute one or more combinations of attribute value and equality/inequality operator are added. Those required type-related component descriptions are sometimes called “typicals”. In the following they are called “role classes” in order to match the AutomationML name space.

2. The requirements engineer or the design engineer defines the structure of a system by creation of a tree or network like structure of objects, each representing one instance element of the future real world system. The whole data structure is called “instance hierarchy” in the following.
3. The requirements engineer or the design engineer creates relations between instance elements of the instance hierarchy and role classes.
4. The design engineer attaches type representatives of real-world components (devices, machines, etc.) to the instance elements or enriches instance elements with attribute values.
5. The validation engineer compares all attribute values of instance elements with the role class attribute values of the related role class for all instance elements with attached role classes. Base for the comparison are the equality/inequality operators attached to the role class attributes. If all comparisons entailed true, then the validation engineer signals “the system construction fulfills the requirements” otherwise she signals “the system construction does not fulfill the requirements”. Additionally a list of comparison cases are reported, which entailed “false”.

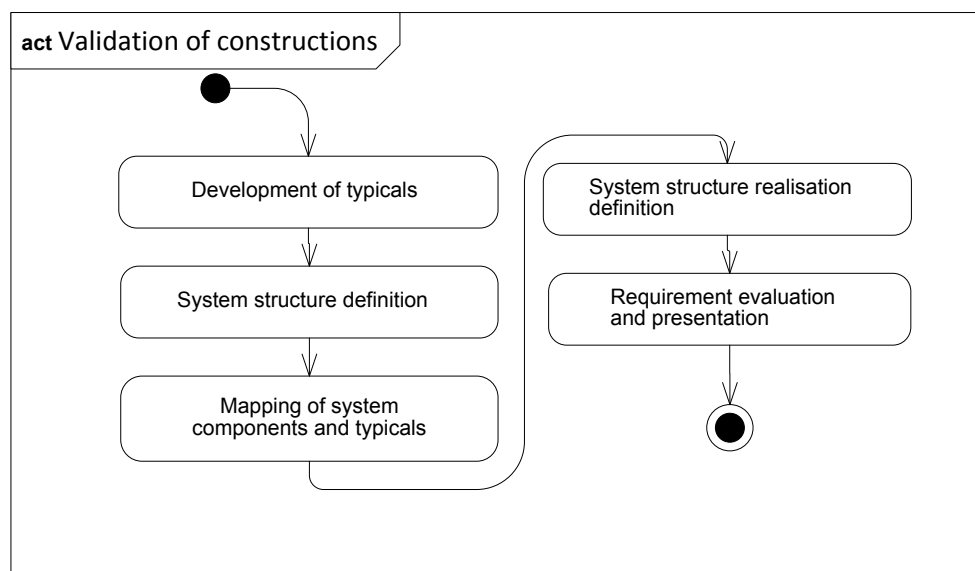


Figure 23 – Activity structure of use case "validation of construction"

Within this use case the requirements data and libraries typically coming from planning and construction tools as well as libraries of system components coming from product data bases are seen as input data. Output data is the evaluation result as a Boolean value.

This use case will provide the following requirements to AutomationML:

- Role classes shall be applicable to model uniquely required system characteristics
- Role class attributes should be enriched with fixed relation between ID and semantics
- Role class attribute values shall be combinable by equality/inequality operators to model requirements
- Instance elements shall contain references to role classes and thus implicitly to namespaces of attribute names
- Instance element shall contain attribute values

The previously mentioned methods for the construction validation are related to attribute values. There may be further requirements specified regarding the parent, sibling or sub-instance hierarchy elements of an element to be validated. Those validations may be processed by involving further requirements specification facilities such as the introduction of Object Constraint Language (OCL) expressions (see [5]). Nevertheless the clear semantic definition of attributes would be a necessity for those validation procedures too.

5.4 Boundary Conditions for Application

Within the execution of the use cases some technical and legal bordering conditions shall be considered.

The developed eCl@ss classification is a very encyclopaedic classification. It contains a huge amount of classes. Not all of them are relevant within an application case. Therefore, in advance to an application case it shall be defined which of the classification classes will be applied in the AutomationML based modelling, i.e. the relevant part of the eCl@ss classification is identified. For example the consideration of automation related systems, devices, and components can be made only reflecting classes with the classification 27-*-*.*.

It may happen that in an application case not all relevant elements can be classified by using eCl@ss classification. In this case other classification standards might be identified for application. It shall be ensured, that also these classification standards will follow the international standards ISO/IEC 11179-6, ISO/TS 29002, and ISO 6532 and provide an IRDI for identification of the classification classes.

If the relevant part of the eCl@ss classification is identified it has to be ensured, that the application of this part is legally possible. eCl@ss is protected by licensing rights of different types. For each application case the right license conditions shall be identified and used. The different eCl@ss license models are available under http://www.eclassdownload.com/catalog/eclass_categories.php.

In any way in each application case the licensing rights shall be respected.

6 Realization of technical use cases

6.1 Integration of attribute and object semantics

To realize the use case “simple semantic identification” as specified in 5.1.1 two cases have to be distinguished. First the specification of the semantics for single attributes and second the semantic description of AutomationML objects. Within the first case the IRDI for a property is used to define the attribute semantic. In contrast, the second case requires the use of the eCI@ss classification layers to specify the semantics of AutomationML objects.

6.1.1 Concept

6.1.1.1 Attribute semantics via IRDI

The AutomationML top level format CAEX includes a concept to reference a semantic definition for attributes. This concept shall be used to define the semantics of AutomationML attributes by using eCI@ss properties. For these definitions, the following provisions apply:

- The CAEX schema element “RefSemantic” shall be used for the semantic definition for a single attribute, see Figure 24.
- The value of the XML attribute “CorrespondingAttributePath” of the CAEX schema element “RefSemantic” element shall be assembled as the following string:
 - “ECLASS:” + IRDI of the eCI@ss property defining the semantics of the AutomationML attribute
- The IRDI shall be used as defined within the eCI@ss specification.
- If the AutomationML Unit attribute is not provided the measurement unit is defined by the referenced eCI@ss property.
- If the AutomationML Unit attribute is provided it shall be selected from the eCI@ss measurement unit table.
- If the AutomationML AttributeDataType attribute is not provided the data type is defined by the referenced eCI@ss property.
- If the AutomationML AttributeDataType attribute is provided it shall be converted according to Table 6.

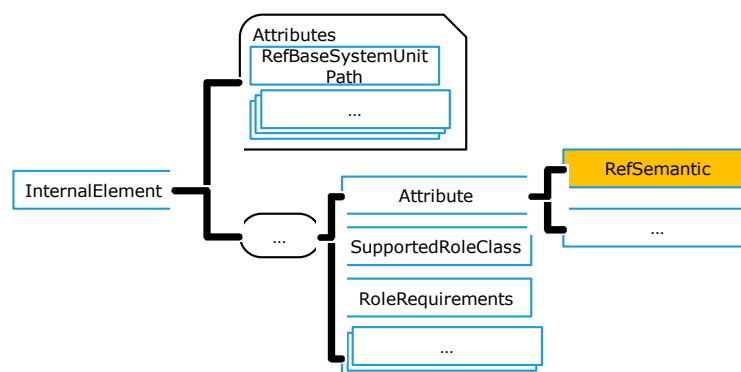
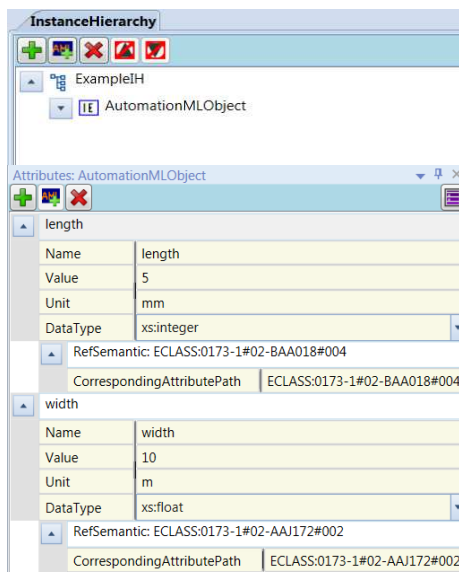


Figure 24 – Applicable means for semantic reference for a single attribute

Example

Figure 25 depicts an example for the use of the CAEX element “RefSemantic” to specify the semantics of an AutomationML attribute. Within this example one AutomationML object with the name “AutomationMLObject” exists. This Object has the two AutomationML attributes “length” and “width”. Each attribute is semantically defined by an IRDI according to the provision above.

For the AutomationML attribute “length” the IRDI has the value “0173-1#02-BAA018#004” and for the AutomationML attribute “width” the value of the corresponding IRDI is “0173-1#02-AAJ172#002”.



The screenshot shows the 'InstanceHierarchy' window. Under 'ExampleIH', there is an 'AutomationMLObject'. Its attributes are listed below. The 'length' attribute has a value of 5, unit mm, and is linked to ECLASS:0173-1#02-BAA018#004. The 'width' attribute has a value of 10, unit m, and is linked to ECLASS:0173-1#02-AAJ172#002.

```
<InstanceHierarchy Name="ExampleIH">
  <InternalElement Name="AutomationMLObject" ID="de88f618-6802-40cc-9dd3-2d73546e6280">
    <Attribute Name="length" AttributeDataType="xs:float" Unit="mm">
      <Description>Attribute with an eClass RefSemantic</Description>
      <DefaultValue>5</ DefaultValue >
      <Value>5</Value>
      <RefSemantic CorrespondingAttributePath="ECLASS:0173-1#02-BAA018#004" />
    </Attribute>
    <Attribute Name="width" AttributeDataType="xs:float" Unit="m">
      <Description>Attribute with an eClass RefSemantic</Description>
      <DefaultValue>10</ DefaultValue >
      <Value>10</Value>
      <RefSemantic CorrespondingAttributePath="ECLASS:4 0173-1#02-AAJ172#002" />
    </Attribute>
    <SupportedRoleClass
      RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/>
      <RoleRequirements
        RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/>
      </RoleRequirements>
    </InternalElement>
  </InstanceHierarchy>
```

Figure 25 – Example RefSemantic for a single attribute

6.1.1.2 Object semantic via eClassClassSpecification RoleClass

To define the semantics of an AutomationML object unambiguously with the help of a semantic specification the AutomationML concept of role classes can be used. Therefore a new AutomationML role class has to be defined. The detailed specification of this role class with the name “*eClassClassSpecification*” can be found in Table 4.

Table 4 – Definition eClassClassSpecification

Role class name	eClassClassSpecification	
Description	The role class “eClassClassSpecification” shall be used in order to reference the corresponding eCI@ss Classification Class for the AutomationML object.	
Parent Class	AutomationMLBaseRole	
Attributes	“eClassVersion” (AttributeDataType=“xs:string”)	The attribute “eClassVersion” shall define the version of the eCI@ss catalogue, which the eCI@ss category is related to. The values of the attribute shall be described as followed: ECLASS-“major release”.“release” Example: ECLASS-8.1 or ECLASS-9.0 “major release” is the number of the major release of the eCI@ss version. “release” is the number of the subrelease of the eCI@ss version.
	“eClassClassificationClass” (AttributeDataType =“xs:string”)	The attribute “eClassClassificationClass” shall define the eCI@ss Classification Class for the AutomationML object. The value shall be the coded name of the classification class as 8-digit integer. Thereby two digits shall be used for each hierarchical level of the class structure. Example: 27242201 <ul style="list-style-type: none"> • 27 for “segment” (Electric engineering, automation, process control engineering) • 24 for „Main group“ (Control) • 22 for “Group” (Programmable logic control) • 01 for Sub-group or commodity class-product group (PLC analouge input/output module)
	“eClassIRDI” (AttributeDataType =“xs:string”)	The attribute “eClassIRDI” shall contain the IRDI for the class. To define ACs the IRDI of the AC shall be used. To define CCs the IRDI of the CC shall be used. The value shall be coded according to eCI@ss definition. Example: 0173-1---ADVANCED_1_1#01-ADN862#003

Figure 26 shows the XML description of the AutomationML role class “eClassClassSpecification”.

```

<RoleClass Name="eClassClassSpecification" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole">
  <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
    <Description>Specifies the IRDI for the application or classification class</Description>
  </Attribute>
  <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
    <Description>Specifies the eClass Classification Class for the AutomationML object coded as 8-digit integer with two digits for each hierarchical level of the class structure.</Description>
  </Attribute>
  <Attribute Name="eClassVersion" AttributeDataType="xs:string">
    <Description>Specifies the version of the eClass catalogue, which the eClass category is related to</Description>
  </Attribute>
</RoleClass>

```

Figure 26 – XML description of the role class eClassClassSpecification

To assign a semantic definition to an AutomationML object with the help of the role class eClassClassSpecification the CAEX elements “SupportedRoleClass” and “RoleRequiements” shall be used according to the AutomationML specification, see Figure 27.

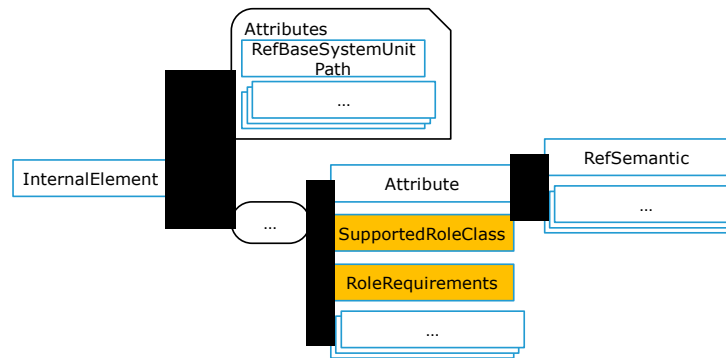


Figure 27 – Applicable means for semantic reference for AutomationML Object

Example

Figure 28 and Figure 29 depict an example for the use of the semantic reference for AutomationML objects. Within this example the AutomationML object is defined as a DC motor with the AutomationML Object name “InstanceMyMotor” for a concrete instance of a motor and “PlaceholderClassificationMotor” for an unspecific object.

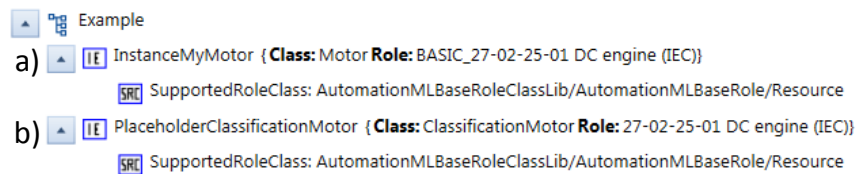


Figure 28 – Example motor AutomationML object with semantic reference for a) AC and b) CC

a)	b)																																																																																				
<table border="1"> <tr><td colspan="2">eClassVersion</td></tr> <tr><td>Name</td><td>eClassVersion</td></tr> <tr><td>Description</td><td>Specifies the version of the eClass catalogue, which the eClass category is related to</td></tr> <tr><td>Value</td><td>9.0</td></tr> <tr><td>Unit</td><td></td></tr> <tr><td>DataType</td><td>xs:string</td></tr> <tr><td colspan="2">eClassClassificationClass</td></tr> <tr><td>Name</td><td>eClassClassificationClass</td></tr> <tr><td>Description</td><td>Specifies the eClass Classification Class for the AutomationML object coded as 8-digit integer with two digits for each hierarchical level of the class structure.</td></tr> <tr><td>Value</td><td>27022501</td></tr> <tr><td>Unit</td><td></td></tr> <tr><td>DataType</td><td>xs:string</td></tr> <tr><td colspan="2">eClassIRDI</td></tr> <tr><td>Name</td><td>eClassIRDI</td></tr> <tr><td>Description</td><td>Specifies the IRDI for the application class</td></tr> <tr><td>Value</td><td>0173-1---BASIC_1_1#01-ABW077#009</td></tr> <tr><td>Unit</td><td></td></tr> <tr><td>DataType</td><td>xs:string</td></tr> <tr><td colspan="2">Construction form of DC motor</td></tr> <tr><td colspan="2">Construction size DC-Motor</td></tr> <tr><td colspan="2">Cooling type</td></tr> <tr><td colspan="2">Field rotation speed</td></tr> <tr><td colspan="2">GTIN</td></tr> <tr><td colspan="2">Impact load</td></tr> </table>	eClassVersion		Name	eClassVersion	Description	Specifies the version of the eClass catalogue, which the eClass category is related to	Value	9.0	Unit		DataType	xs:string	eClassClassificationClass		Name	eClassClassificationClass	Description	Specifies the eClass Classification Class for the AutomationML object coded as 8-digit integer with two digits for each hierarchical level of the class structure.	Value	27022501	Unit		DataType	xs:string	eClassIRDI		Name	eClassIRDI	Description	Specifies the IRDI for the application class	Value	0173-1---BASIC_1_1#01-ABW077#009	Unit		DataType	xs:string	Construction form of DC motor		Construction size DC-Motor		Cooling type		Field rotation speed		GTIN		Impact load		<table border="1"> <tr><td colspan="2">eClassVersion</td></tr> <tr><td>Name</td><td>eClassVersion</td></tr> <tr><td>Description</td><td>Specifies the version of the eClass catalogue, which the eClass category is related to</td></tr> <tr><td>Value</td><td>9.0</td></tr> <tr><td>Unit</td><td></td></tr> <tr><td>DataType</td><td>xs:string</td></tr> <tr><td colspan="2">eClassClassificationClass</td></tr> <tr><td>Name</td><td>eClassClassificationClass</td></tr> <tr><td>Description</td><td>Specifies the eClass Classification Class for the AutomationML object coded as 8-digit integer with two digits for each hierarchical level of the class structure.</td></tr> <tr><td>Value</td><td>27022501</td></tr> <tr><td>Unit</td><td></td></tr> <tr><td>DataType</td><td>xs:string</td></tr> <tr><td colspan="2">eClassIRDI</td></tr> <tr><td>Name</td><td>eClassIRDI</td></tr> <tr><td>Description</td><td>Specifies the IRDI for the application class</td></tr> <tr><td>Value</td><td>0173-1#01-AKE162#011</td></tr> <tr><td>Unit</td><td></td></tr> <tr><td>DataType</td><td>xs:string</td></tr> </table>	eClassVersion		Name	eClassVersion	Description	Specifies the version of the eClass catalogue, which the eClass category is related to	Value	9.0	Unit		DataType	xs:string	eClassClassificationClass		Name	eClassClassificationClass	Description	Specifies the eClass Classification Class for the AutomationML object coded as 8-digit integer with two digits for each hierarchical level of the class structure.	Value	27022501	Unit		DataType	xs:string	eClassIRDI		Name	eClassIRDI	Description	Specifies the IRDI for the application class	Value	0173-1#01-AKE162#011	Unit		DataType	xs:string
eClassVersion																																																																																					
Name	eClassVersion																																																																																				
Description	Specifies the version of the eClass catalogue, which the eClass category is related to																																																																																				
Value	9.0																																																																																				
Unit																																																																																					
DataType	xs:string																																																																																				
eClassClassificationClass																																																																																					
Name	eClassClassificationClass																																																																																				
Description	Specifies the eClass Classification Class for the AutomationML object coded as 8-digit integer with two digits for each hierarchical level of the class structure.																																																																																				
Value	27022501																																																																																				
Unit																																																																																					
DataType	xs:string																																																																																				
eClassIRDI																																																																																					
Name	eClassIRDI																																																																																				
Description	Specifies the IRDI for the application class																																																																																				
Value	0173-1---BASIC_1_1#01-ABW077#009																																																																																				
Unit																																																																																					
DataType	xs:string																																																																																				
Construction form of DC motor																																																																																					
Construction size DC-Motor																																																																																					
Cooling type																																																																																					
Field rotation speed																																																																																					
GTIN																																																																																					
Impact load																																																																																					
eClassVersion																																																																																					
Name	eClassVersion																																																																																				
Description	Specifies the version of the eClass catalogue, which the eClass category is related to																																																																																				
Value	9.0																																																																																				
Unit																																																																																					
DataType	xs:string																																																																																				
eClassClassificationClass																																																																																					
Name	eClassClassificationClass																																																																																				
Description	Specifies the eClass Classification Class for the AutomationML object coded as 8-digit integer with two digits for each hierarchical level of the class structure.																																																																																				
Value	27022501																																																																																				
Unit																																																																																					
DataType	xs:string																																																																																				
eClassIRDI																																																																																					
Name	eClassIRDI																																																																																				
Description	Specifies the IRDI for the application class																																																																																				
Value	0173-1#01-AKE162#011																																																																																				
Unit																																																																																					
DataType	xs:string																																																																																				

Figure 29 – Attributes to the example motor AutomationML object with semantic reference for a) CC and b) AC

Figure 30 depicts the XML view of example a), which includes the object “InstanceMyMotor”.

```
<InternalElement Name="InstanceMyMotor" RefBaseSystemUnitPath="ExampleLib/Motor" ID="9e64d613-3b95-4af5-b826-072a7732b9a0">
  <Attribute Name="eClassVersion" AttributeDataType="xs:string">
    <Description>Specifies the version of the eClass catalogue, which the eClass category is related to</Description>
    <Value>9.0</Value>
  </Attribute>
  <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
    <Description>Specifies the eClass Classification Class for the AutomationML object coded as 8-digit integer with two digits for each hierarchical level of the class structure.
    </Description>
    <Value>27022501</Value>
  </Attribute>
  <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
    <Description>Specifies the IRDI for the application class</Description>
    <Value>0173-1---BASIC_1_1#01-ABW077#009</Value>
  </Attribute>
  <Attribute Name="Construction form of DC motor" AttributeDataType="xs:string">
    <Description>Arrangement of machine parts in reference to anchorage, arrangement of bearings and shafts</Description>
    <RefSemantic CorrespondingAttributePath="ECLASS:0173-1#02-BAE069#007" />
  </Attribute>
  <Attribute Name="Construction size DC-Motor" AttributeDataType="xs:string">
    <Description>Housing size of a DC motor, from which the attachment and anchorage dimensions are derived</Description>
    <RefSemantic CorrespondingAttributePath="ECLASS:0173-1#02-BAE072#005" />
  </Attribute>
  <Attribute Name="Cooling type" AttributeDataType="xs:string">
    <Description>Summary of various types of cooling, for use as search criteria that limit a selection</Description>
    <RefSemantic CorrespondingAttributePath="ECLASS:0173-1#02-BAE122#006" />
  </Attribute>
  ...
  <SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
  <SupportedRoleClass RefRoleClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/27 Electric engineering, automation, process control engineering /27-02 Electrical drive/27-02-25 DC engine/27-02-25-01 DC engine (IEC)/BASIC_27-02-25-01 DC engine (IEC)" />
</InternalElement>
```

Figure 30 – XML View of the example motor AutomationML object with semantic reference (AC)

6.2 Generation process for eCI@ss AutomationML role model

To realize the use case “Semantic identification including property guarantees” as specified in section 5.1.2 it is necessary to go beyond simple referencing of a catalogue object. In contrast it is necessary to integrate the semantic representation by using the standardized semantic representation using roles. Following **Fehler! Verweisquelle konnte nicht gefunden werden.** in this case the data elements “SupportedRoleClass” and “RoleRequirement” are applied as depicted in Figure 31.

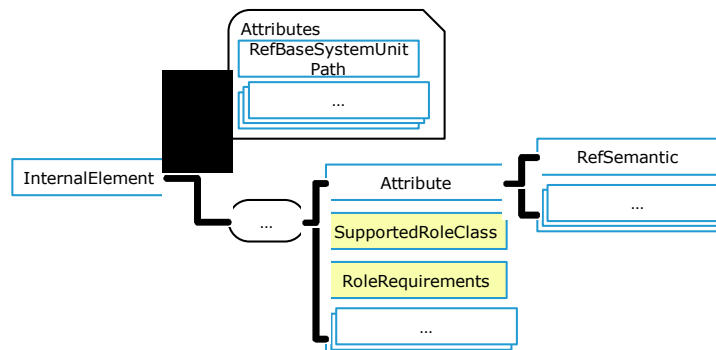


Figure 31 – Applicable means for semantic representation used for eCI@ss Basic modelling

6.2.1 Concept

The semantic identification including property guarantees is based on the generation of a role class library covering the object identification and the attribute definition and its referencing using the data elements “SupportedRoleClass” and “RoleRequirement” as well as the use of appropriate role related attributes.

6.2.1.1 Role class library generation process

At first a role class library has to be developed modelling the used catalogue. Without loss of generality and based on IEC 61360 in the following it is assumed, that

- the catalogue is structured as a hierarchical tree,
- each tree node represents a class of objects to be categorized,
- each tree node may have but need not to have a set of attributes,
- each sub node of a node represents a more detailed class of objects to be categorized,
- each sub node may have but need not to have additional attributes, and
- each leave node of the tree has at least one attribute.

Currently, the only available, vendor-independent classification standard for devices exploitable in factory automation is eCI@ss.

Following these assumptions the following algorithm shall be applied to create the role class library for a given catalogue.

Step 1. Create a new RoleClassLibrary with

- a. a unique name following the AutomationML rules,
- b. a unique version number following the AutomationML rules, and
- c. a meaningful description following the AutomationML rules.

Step 2. Select one node of the catalogue class tree where all super nodes of this node have been modeled as role.

Step 3. For the selected node do:

- a. Create a new role class as and with
 - i. a unique name following the AutomationML rules,
 - ii. a RefBaseClassPath to
 1. the role class defined for the direct super node of the considered node or
 2. the role eClassClassSpecification defined in Section 6.1 if there is no supernode).
- b. Complete the attributes “eClassVersion”, “eClassClassificationClass” and “eCI@ssIRDI” as defined in Section 6.1
- c. For each attribute of the considered node create an AutomationML attribute to the role class and with
 - i. a unique name following the AutomationML rules,
 - ii. a meaningful description following the AutomationML rules,
 - iii. a default value (if necessary) following the AutomationML rules,
 - iv. a unit following the AutomationML rules,
 - v. a data type following the AutomationML rules, and
 - vi. a “RefSemantic” following section 6.1.

Repeat the steps 2 and 3 for all items of your catalogue that should be part of your AutomationML RoleClassLibrary.

Note 1: It is recommended to rebuild the node hierarchy of the catalogue as a role class hierarchy.

Note 2: It is recommended to use the names of catalogue classes and its attributes as names of the role classes and their attributes.

Figure 32 depicts an example for a simple AutomationML role class library that is implemented according to the described algorithm.

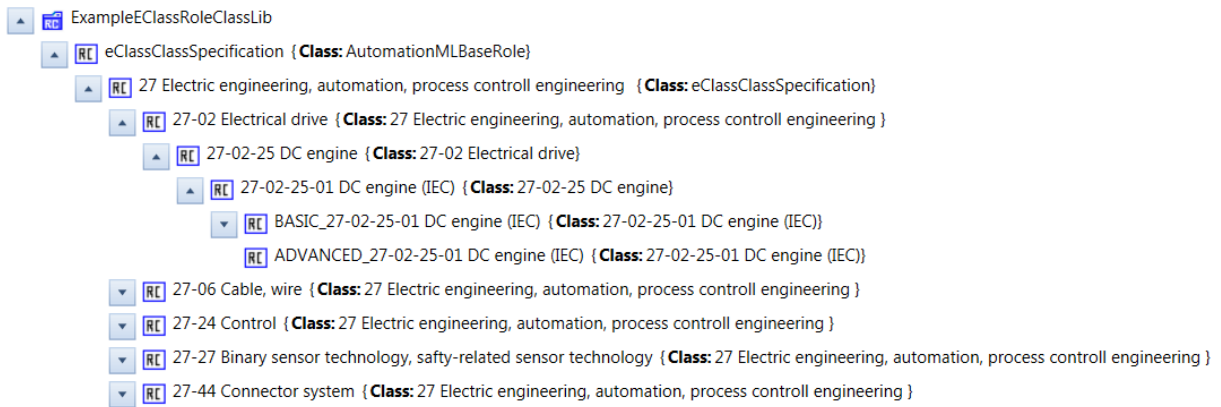


Figure 32 – Example Role Class

Within this example each role class that represents an eClass application class has eClass properties that are modelled as AutomationML attributes. Figure 33 shows possible attributes for an role class.

▼	eClassVersion
▼	eClassClassificationClass
▼	eClassIRDI
▲	Construction form of DC motor
Name	Construction form of DC motor
Description	Arrangement of machine parts in reference to anchorage, arrangement of bearings and shafts
Value	
Unit	
DataType	xs:string
▼	RefSemantic: ECLASS:0173-1#02-BAE069#007
▲	Construction size DC-Motor
Name	Construction size DC-Motor
Description	Housing size of a DC motor, from which the attachment and anchorage dimensions are derived
Value	
Unit	
DataType	xs:string
▼	RefSemantic: ECLASS:0173-1#02-BAE072#005
▼	Cooling type
▼	Field rotation speed
▼	GTIN
▼	Impact load
▼	Manufacturer name
▼	Manufacturer product number
▼	Min. rotation speed
▼	Nominal speed
▼	Power loss, static, current-independent [PIs]
▼	Product name
▼	Product type description
▼	Protection type (reduced)
▼	Rated performance (reduced)
▼	Rated voltage (reduced)
▼	Supplier name
▼	Supplier product number

Figure 33 – Attributes within an Example Role Class

Figure 34 depicts the XML view of the example. For better understanding it does not include all attribute definitions and the descriptions for the attributes.

```

<RoleClassLib Name="ExampleEClassRoleClassLib">
  <Description>Component roles developed by Gecko Project (proprietary)</Description>
  <Version>0.9</Version>
  <RoleClass Name="eClassClassSpecification" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole">
    <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
      <Description>Specifies the IRDI for the application or classification class</Description>
    </Attribute>
    <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
      <Description>Specifies the eClass Classification Class for the AutomationML object coded as 8-digit integer with two digits for each hierarchical
        level of the class structure.</Description>
    </Attribute>
    <Attribute Name="eClassVersion" AttributeDataType="xs:string">
      <Value>9.0</Value>
    </Attribute>
    <RoleClass Name="27 Electric engineering, automation, process controll engineering ">
      RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification">
        <RoleClass Name="27-02 Electrical drive">
          RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
            27 Electric engineering, automation, process controll engineering ">
            <RoleClass Name="27-02-25 DC engine">
              RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/27 Electric
                engineering, automation, process controll engineering /27-02 Electrical drive">
              <RoleClass Name="27-02-25-01 DC engine (IEC)">
                RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
                  27 Electric engineering, automation, process controll engineering /27-02 Electrical drive/
                    27-02-25 DC engine">
                <Attribute Name="eClassVersion" AttributeDataType="xs:string">
                  <Value>9.0</Value>
                </Attribute>
                <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
                  <Value>27022501</Value>
                </Attribute>
                <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
                  <Value>0173-1#01-AKE162#011</Value>
                </Attribute>
                <RoleClass Name="BASIC_27-02-25-01 DC engine (IEC)">
                  RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/27 Electric engineering,
                    automation, process controll engineering /27-02 Electrical drive/27-02-25 DC engine/
                      27-02-25-01 DC engine (IEC)">
                  <Attribute Name="eClassVersion" AttributeDataType="xs:string">
                    <Value>9.0</Value>
                  </Attribute>
                  <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
                    <Value>27022501</Value>
                  </Attribute>
                  <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
                    <Value>0173-1---BASIC_1_1#01-ABW077#009</Value>
                  </Attribute>
                  <Attribute Name="Construction form of DC motor" AttributeDataType="xs:string">
                    <RefSemantic CorrespondingAttributePath="ECLASS:0173-1#02-BAE069#007" />
                  </Attribute>
                  <Attribute Name="Construction size DC-Motor" AttributeDataType="xs:string">
                    <RefSemantic CorrespondingAttributePath="ECLASS:0173-1#02-BAE072#005" />
                  </Attribute>
                </RoleClass>
              </RoleClass>
            </RoleClass>
          </RoleClass>
        </RoleClass>
      </RoleClass>
    <RoleClass Name="27-06 Cable, wire">
      RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
        27 Electric engineering, automation, process controll engineering ">
    </RoleClass>
    <RoleClass Name="27-24 Control">
      RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
        27 Electric engineering, automation, process controll engineering ">
    </RoleClass>
    <RoleClass Name="27-27 Binary sensor technology, safty-related sensor technology">
      RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
        27 Electric engineering, automation, process controll engineering ">
    </RoleClass>
    <RoleClass Name="27-44 Connector system">
      RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
        27 Electric engineering, automation, process controll engineering ">
    </RoleClass>
  </RoleClassLib>

```

Figure 34 – Example Role Class XML View

Note: To apply the generation process of an AutomationML RoleClassLibrary that represents the eCl@ss Dictionary or parts of it, it is necessary to have the legal rights to use the eCl@ss Dictionary. For further information please contact the eCl@ss e.V. office.

Role class library application process

At second the role classes have to be used within the semantic referencing process.

6.2.2 Attribute transformation best practice

In eCl@ss an attribute consists of the elements characteristic, definition, value, unit and data type. The same elements are used in AML with the following notations: "Name", "Description", "Value", "Unit" and "DataType".

In eCl@ss the RefSemantic is extracted from the list of characteristics. They use a combination of letters, numbers and special characters, which follow in AML after "ECLASS:".

In the following Table 5 the terms are accordingly contrasted and the XML formats were added.

Table 5 – Translation of the attributes of eCl@ss to AML

eCl@ss description	AutomationML description	XML Example AutomationML
eCl@ss property	AutomationML Attribute	<pre><Attribute> ... </Attribute></pre>
Preferred name	Name	<pre><Attribute Name="min. input voltage (at DC)"> ... </Attribute></pre>
Definition	Description	<pre><Attribute > <Description>greatest possible value of DC voltage, that can be applied at the input of a working fund </Description> </Attribute></pre>
Value	Value	<pre><Attribute> <Value>18</Value> </Attribute></pre>
Unit	Unit	<pre><Attribute Unit="V"> ... </Attribute></pre>
DataType	DataType	<pre><Attribute AttributeDataType="xs:float"> ... </Attribute></pre>
(Merkmalliste)	RefSemantic	<pre><Attribute> <RefSemantic CorrespondingAttributePath="ECLASS:0173-1#02- AAB977#007"> </RefSemantic> </Attribute></pre>

Aside from the notations used by eCl@ss the data formats (which are given in http://wiki.eclass.eu/wiki/Property#Data_type_.28Property.29 for example) have to be translated, too. In AutomationML there is always a "xs:" in front of the notation. For example the data format "BOOLEAN" in eCl@ss becomes "xs:boolean" in AML.

In Table 5 the respective translations are listed. As AutomationML has to rely on the data types defined in XML specifications the semantical richness of data types in eCl@ss is not available in AutomationML. The original eCl@ss data types have to be determined out of AutomationML files by use of the IRDI given in the SemanticRef information of the attribute.

In the following table additionally examples for the data format represented in eCl@ss and AutomationML are given.

Table 6 – Translation of the data types from eCl@ss to AutomationML permitted data types

eCl@ss	Example eCl@ss	AutomationML	Example AutomationML
BOOLEAN	Yes	xs:boolean	true
STRING	0173-1#01-ADG629#001 ; DN 700 ; 10 Mbps	xs:string	0173-1#01-ADG629#001 ; DN 700 ; 10 Mbps
STRING_TRANSLATABLE	Red ; Green ; Aluminum	xs:string	Red ; Green ; Aluminum
INTEGER_COUNT	1 ; 10 ; 111	xs:integer	1 ; 10 ; 111
INTEGER_MEASURE	1 ; 10 ; 111	xs:integer	1 ; 10 ; 111
INTEGER_CURRENCY	1 ; 10 ; 111	xs:integer	1 ; 10 ; 111
REAL_COUNT	1,5 ; 102,35	xs:float	1,5 ; 102,35
REAL_MEASURE	1,5 ; 102,35	xs:float	1,5 ; 102,35
REAL_CURRENCY	1,5 ; 102,35	xs:float	1,5 ; 102,35
RATIONAL	1/3, 1 2/3	xs:float	0.333333333, 1.666666666
RATIONAL_MEASURE	1/3, 1 2/3	xs:float	0.333333333, 1.666666666
TIME	12:45	xs:time	12:45:00
TIMESTAMP	1979-01-15 12:45	xs:datetime	1979-01-15T12:45:00.0
DATE	1979-01-15	xs:date	1979-01-15
URL	http://www.automationml.org	xs:anyURI	http://www.automationml.org
Level Type (only Advanced) Includes the minimal, maximal, typical and nominal value	min, max, typ, nom	xs:string	min, max, typ, nom
Axis Type (only Advanced) Is a placement type and defines points in a geometry schema (axis1_placement_type, axis2_placement_2d_type, axis2_placement_3d_type), according to ISO 13584-42:2010, referring to ISO 10303-42	axis1_placement_type, axis2_placement_2d_type, axis2_placement_3d_type	xs:string	axis1_placement_type, axis2_placement_2d_type, axis2_placement_3d_type

Example

As an example for the usage of these definitions, the engineering of a production system containing a conveyor system shall be applied. Within this system drives, inductive proximity switches, and different types of cables are applied.

Figure 35 and Figure 36 represent the role class library generated by using the eCl@ss classification catalogue basic Version 9.0 for the example plant. It contains relevant parts of the classification catalogue including the classification classes for drives, inductive proximity switches and different types of cables.



Figure 35 – Example eClass classification role class library

```

<RoleClassLib Name="ExampleEClassRoleClassLib">
  <Version>0.9</Version>
  <RoleClass Name="eClassClassSpecification" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole">
    <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
      <Description>Specifies the IREDI for the application or classification class</Description>
    </Attribute>
    <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
      <Description>Specifies the eClass Classification Class for the AutomationML object coded as 8-digit integer with two digits for each hierarchical
        level of the class structure.</Description>
    </Attribute>
    <Attribute Name="eClassVersion" AttributeDataType="xs:string">
      <Description>Specifies the version of the eClass catalogue, which the eClass category is related to</Description>
    </Attribute>
    <RoleClass Name="27 Electric engineering, automation, process controll engineering"
      RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification">
      <RoleClass Name="27-02 Electrical drive"
        RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
        27 Electric engineering, automation, process controll engineering ">
        <RoleClass Name="27-02-25 DC engine"
          RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
          27 Electric engineering, automation, process controll engineering /27-02 Electrical drive">
          <RoleClass Name="27-02-25-01 DC engine (IEC)"
            RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
            27 Electric engineering, automation, process controll engineering /27-02 Electrical drive/27-02-25 DC engine">
            <Attribute Name="eClassVersion" AttributeDataType="xs:string">
              <Value>9.0</Value>
            </Attribute>
            <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
              <Value>27022501</Value>
            </Attribute>
            <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
              <Value>0173-1#01-AKE162#011</Value>
            </Attribute>
            <RoleClass Name="BASIC_27-02-25-01 DC engine (IEC)"
              RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
              27 Electric engineering, automation, process controll engineering /
                27-02 Electrical drive/27-02-25 DC engine/27-02-25-01 DC engine (IEC)">
              <Attribute Name="eClassVersion" AttributeDataType="xs:string">
                <Value>9.0</Value>
              </Attribute>
              <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
                <Value>27022501</Value>
              </Attribute>
              <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
                <Value>0173-1---BASIC_1_1#01-ABW077#009</Value>
              </Attribute>
            </RoleClass>
            <RoleClass Name="ADVANCED_27-02-25-01 DC engine (IEC)"
              RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
              27 Electric engineering, automation, process controll engineering /
                27-02 Electrical drive/27-02-25 DC engine/27-02-25-01 DC engine (IEC)">
              <Attribute Name="eClassVersion" AttributeDataType="xs:string">
                <Value>9.0</Value>
              </Attribute>
              <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
                <Value>27022501</Value>
              </Attribute>
              <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
                <Value>0173-1---ADVANCED_1_1#01-ADN779#005</Value>
              </Attribute>
            </RoleClass>
          </RoleClass>
        </RoleClass>
      </RoleClass>
      <RoleClass Name="27-06 Cable, wire"
        RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
        27 Electric engineering, automation, process controll engineering ">
      <RoleClass Name="27-06-18 Communication cabel"
        RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
        27 Electric engineering, automation, process controll engineering /27-06 Cable, wire">
      <RoleClass Name="27-06-18-01 Data cabel"
        RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
        27 Electric engineering, automation, process controll engineering /
          27-06 Cable, wire/27-06-18 Communication cabel"/>
      </RoleClass>
      <RoleClass Name="27-06-03 Ready-made data cable"
        RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
        27 Electric engineering, automation, process controll engineering /27-06 Cable, wire">
      <RoleClass Name="27-06-03-07 Bus cabel"
        RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
        27 Electric engineering, automation, process controll engineering /
          27-06 Cable, wire/27-06-03 Ready-made data cable" />
      <RoleClass Name="27-06-03-11 Assembled sensor actuator-line"
        RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
        27 Electric engineering, automation, process controll engineering /
          27-06 Cable, wire/27-06-03 Ready-made data cable" />
      </RoleClass>
      </RoleClass>
      <RoleClass Name="27-24 Control"
        RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
        27 Electric engineering, automation, process controll engineering ">
      <RoleClass Name="27-24-06 PC-based controls"
        RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
        27 Electric engineering, automation, process controll engineering /27-24 Control">
      <RoleClass Name="27-24-06-90 PC-basierte controls (unspecified)"
        RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"
        27 Electric engineering, automation, process controll engineering /
          27-24 Control/27-24-06 PC-based controls" />
      </RoleClass>
      <RoleClass Name="27-24-26 Field bus, decentralized peripheral"
        RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/"

```

```

27 Electric engineering, automation, process controll engineering /27-24 Control">
<RoleClass Name="27-24-26-04 Field bus, decentralized peripheral - digital I/O module"
RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
27 Electric engineering, automation, process controll engineering /
27-24 Control/27-24-26 Field bus, decentralized peripheral" />
<RoleClass Name="27-24-26-07 Field bus, decentralized peripheral - basic device"
RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
27 Electric engineering, automation, process controll engineering /
27-24 Control/27-24-26 Field bus, decentralized peripheral" />
</RoleClass>
</RoleClass>
<RoleClass Name="27-27 Binary sensor technology, safty-related sensor technology"
RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
27 Electric engineering, automation, process controll engineering ">
<RoleClass Name="27-27-01 Proximity switch"
RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
27 Electric engineering, automation, process controll engineering /
27-27 Binary sensor technology, safty-related sensor technology">
<RoleClass Name="27-27-01-01 Induktive proximity switch"
RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
27 Electric engineering, automation, process controll engineering /
27-27 Binary sensor technology, safty-related sensor technology/27-27-01 Proximity switch">
<Attribute Name="eClassIRDI" AttributeDataType="xs:string">
<Value>0173-1#01-AGZ376#013</Value>
</Attribute>
<RoleClass Name="BASIC_27-27-01-01 Induktive proximity switch"
RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
27 Electric engineering, automation, process controll engineering /
27-27 Binary sensor technology, safty-related sensor technology/
27-27-01 Proximity switch/27-27-01-01 Induktive proximity switch">
<Attribute Name="eClassIRDI" AttributeDataType="xs:string">
<Value>0173-1---BASIC_1_1#01-ABT934#010</Value>
</Attribute>
</RoleClass>
<RoleClass Name="ADVANCED_27-27-01-01 Induktive proximity switch"
RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
27 Electric engineering, automation, process controll engineering /
27-27 Binary sensor technology, safty-related sensor technology/
27-27-01 Proximity switch/27-27-01-01 Induktive proximity switch">
<Attribute Name="eClassIRDI" AttributeDataType="xs:string">
<Value>0173-1---ADVANCED_1_1#01-ADN934#005</Value>
</Attribute>
</RoleClass>
</RoleClass>
</RoleClass>
</RoleClass>
<RoleClass Name="27-44 Connector system"
RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
27 Electric engineering, automation, process controll engineering ">
<RoleClass Name="27-44-01 Industry connector"
RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
27 Electric engineering, automation, process controll engineering /27-44 Connector system">
<RoleClass Name="27-44-01-01 Rectangular connectors (set)"
RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
27 Electric engineering, automation, process controll engineering /27-44 Connector system/
27-44-01 Industry connector" />
</RoleClass>
</RoleClass>
</RoleClass>
</RoleClassLib>

```

Figure 36 – XML representation of example eClass classification role class library

In addition relevant attributes for the different classes are defined. Figure 37 and Figure 38 depicts the attributes defined for the drive related application class DC-Motor (IEC). For a better understanding of the example only a subset of the possible attributes are included into the figures. In Figure 37 the attributes are depicted as list. Thereby the first three attributes define the semantic of the role class according to the definition in chapter 6.1.1.2. In Figure 38 the same attributes are illustrated as XML representation.

eClassVersion	
eClassClassificationClass	
eClassIRDI	
Construction form of DC motor	
Name	Construction form of DC motor
Description	Arrangement of machine parts in reference to anchorage, arrangement of bearings and shafts
Value	
Unit	
DataType	xs:string
RefSemantic: ECLASS:0173-1#02-BAE069#007	
Construction size DC-Motor	
Name	Construction size DC-Motor
Description	Housing size of a DC motor, from which the attachment and anchorage dimensions are derived
Value	
Unit	
DataType	xs:string
RefSemantic: ECLASS:0173-1#02-BAE072#005	
Cooling type	
Field rotation speed	
Name	Field rotation speed
Description	Rotation speed of the electric field in stator coils
Value	
Unit	1/m
DataType	xs:integer
RefSemantic: ECLASS:0173-1#02-AAC875#005	
GTIN	
Impact load	
Manufacturer name	

Figure 37 – Example of AutomationML role class attributes for eCl@ss basic AC of 27-02-25-01 DC engine (IEC)

```

<RoleClass Name="BASIC_27-02-25-01 DC engine (IEC)"
  RefBaseClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/27 Electric engineering, automation, process controll engineering /
  27-02 Electrical drive/27-02-25 DC engine/27-02-25-01 DC engine (IEC)">
  <Attribute Name="eClassVersion" AttributeDataType="xs:string">
    <Value>9.0</Value>
  </Attribute>
  <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
    <Value>27022501</Value>
  </Attribute>
  <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
    <Value>0173-1---BASIC_1_1#01-ABW077#009</Value>
  </Attribute>
  <Attribute Name="Construction form of DC motor" AttributeDataType="xs:string">
    <Description>Arrangement of machine parts in reference to anchorage, arrangement of bearings and shafts</Description>
    <RefSemantic CorrespondingAttributePath="ECLASS:0173-1#02-BAE069#007" />
  </Attribute>
  <Attribute Name="Construction size DC-Motor" AttributeDataType="xs:string">
    <Description>Housing size of a DC motor, from which the attachment and anchorage dimensions are derived</Description>
    <RefSemantic CorrespondingAttributePath="ECLASS:0173-1#02-BAE072#005" />
  </Attribute>
  <Attribute Name="Cooling type" AttributeDataType="xs:string">
    <Description>Summary of various types of cooling, for use as search criteria that limit a selection</Description>
    <RefSemantic CorrespondingAttributePath="ECLASS:0173-1#02-BAE122#006" />
  </Attribute>
  <Attribute Name="Field rotation speed" AttributeDataType="xs:integer" Unit="1/m">
    <Description>Rotation speed of the electric field in stator coils</Description>
    <RefSemantic CorrespondingAttributePath="ECLASS:0173-1#02-AAC875#005" />
  </Attribute>
  <Attribute Name="GTIN" AttributeDataType="xs:string">
    <Description>internationally unique and unambiguous article number for products and services (Global Trade Item Number, formerly
EAN)</Description>
    <RefSemantic CorrespondingAttributePath="0173-1#02-AAO663#002" />
  </Attribute>
  <Attribute Name="Impact load" AttributeDataType="xs:string">
    <Description>Current increase as a multiple of the nominal current based on short-term load moment leaps</Description>
    <RefSemantic CorrespondingAttributePath="0173-1#02-BAE101#005" />
  </Attribute>
  <Attribute Name="Manufacturer name" AttributeDataType="xs:string">
    <Description>legally valid designation of the natural or judicial person which is directly responsible for the design, production, packaging and labeling of
a product in respect to its being brought into circulation</Description>
    <RefSemantic CorrespondingAttributePath="0173-1#02-AAO677#001" />
  </Attribute>
</RoleClass>

```

Figure 38 – XML representation of example of attributes for the eClass classification role motor

Based on the defined role class library a device catalogue can be created covering the necessary devices for the example.

Figure 39 depicts a system unit class library containing a system unit “Motor” class for a drive derived from the role class “BASIC_27-02-25-01 DC-Motor (IEC)” and a system unit class

"InductiveProximitySwitches" derived from the role class "BASIC_27-27-01-01 Inductive proximity switch".

Figure 39 represent the eCI@ss related attributes for the system unit class Motor.

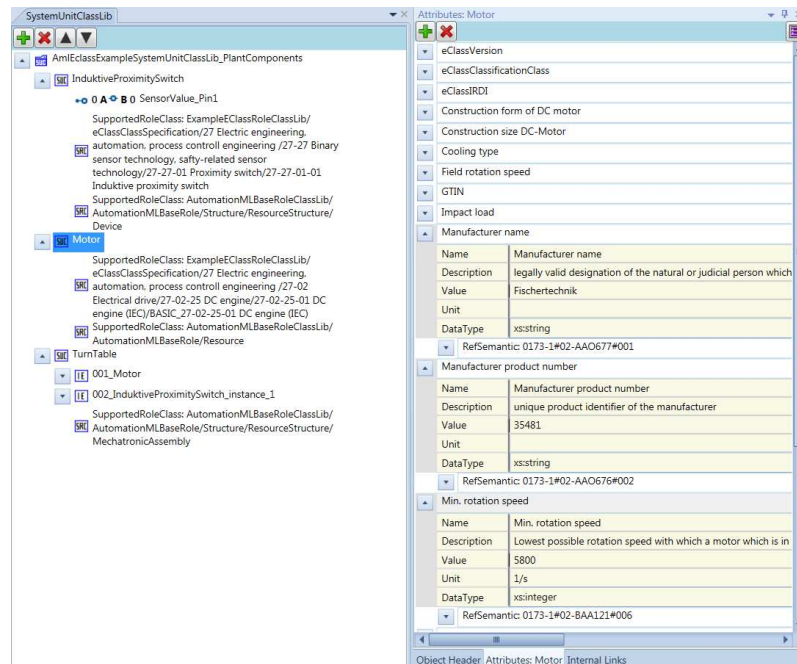


Figure 39 – Example of system unit class library including the attributes for the SUCs

In Figure 40 the XML representation of the example system unit class library can be found. Within this example the attributes of the SUC are not mentioned for better readability.

```
<SystemUnitClassLib Name="AmlEclassExampleSystemUnitClassLib_PlantComponents">
  <Version>0.9</Version>
  <SystemUnitClass Name="InductiveProximitySwitch">
    <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
      <Value>0173-1#01-AGZ376#013</Value>
    </Attribute>
    <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
      <Value>27270101</Value>
    </Attribute>
    <Attribute Name="eClassVersion" AttributeDataType="xs:string">
      <Value>9.0</Value>
    </Attribute>
    <ExternalInterface Name="SensorValue_Pin1"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication/SignalInterface"
      ID="fb307d80-3469-47b8-8b8b-dfa2fd84b86d" />
    <SupportedRoleClass RefRoleClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
      27 Electric engineering, automation, process controll engineering /
      27-27 Binary sensor technology, safty-related sensor technology/27-27-01 Proximity switch/27-27-01-01 Inductive proximity switch" />
    <SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ResourceStructure/Device" />
  </SystemUnitClass>
  <SystemUnitClass Name="Motor">
    <Attribute Name="eClassVersion" AttributeDataType="xs:string">
      <Value>9.0</Value>
    </Attribute>
    <Attribute Name="eClassClassificationClass" AttributeDataType="xs:string">
      <Value>27022501</Value>
    </Attribute>
    <Attribute Name="eClassIRDI" AttributeDataType="xs:string">
      <Value>0173-1---BASIC_1_1#01-ABW077#009</Value>
    </Attribute>
    <SupportedRoleClass RefRoleClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
      27 Electric engineering, automation, process controll engineering /
      27-02 Electrical drive/27-02-25 DC engine/27-02-25-01 DC engine (IEC)/BASIC_27-02-25-01 DC engine (IEC)" />
    <SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
  </SystemUnitClass>
  <SystemUnitClass Name="TurnTable">
    <InternalElement
      Name="001_Motor"
      RefBaseSystemUnitPath="AmlEclassExampleSystemUnitClassLib_PlantComponents/Motor"
      ID="9b4adf04-daea-474f-9040-8bea47172174">
      <SupportedRoleClass RefRoleClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
        27 Electric engineering, automation, process controll engineering /
        27-02 Electrical drive/27-02-25 DC engine/27-02-25-01 DC engine (IEC)/BASIC_27-02-25-01 DC engine (IEC)" />
      <SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
    </InternalElement>
    <InternalElement
      Name="002_InductiveProximitySwitch_instance_1"
      RefBaseSystemUnitPath="AmlEclassExampleSystemUnitClassLib_PlantComponents/InductiveProximitySwitch"
      ID="ed5f71c9-2b38-484d-baff-133b000c6b1b">
    </InternalElement>
    <ExternalInterface Name="SensorValue_Pin1"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication/SignalInterface"
      ID="fb307d80-3469-47b8-8b8b-dfa2fd84b86d" />
  </SystemUnitClass>
</SystemUnitClassLib>
```

```

RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication/SignalInterface"
ID="c941a685-c539-412a-84dc-fb9f5147b277" />
<SupportedRoleClass RefRoleClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
27 Electric engineering, automation, process controll engineering /27-27 Binary sensor technology, safty-related sensor technology/
27-27-01 Proximity switch/27-27-01-01 Induktive proximity switch/BASIC_27-27-01-01 Induktive proximity switch" />
<SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ResourceStructure/Device" />
</InternalElement>
<SupportedRoleClass
RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ResourceStructure/MechatronicAssembly"/> </SystemUnitClass>
</SystemUnitClassLib>

```

Figure 40 – XML representation of example of system unit class library

Using the defined role class library and the defined system unit class library the instance hierarchy covering the engineering data to be exchanged can be modelled. Figure 41 represents the instance hierarchy of the example system containing one turntable which contains a drive and a proximity switch, a control cabinet to host the controller and a set of wires to establish the wiring between devices and control cabinet. It becomes evident that the internal element “001_Motor” has the role requirement “BASIC_27-02-25-01 DC engine (IEC)” and is derived from the system unit class “Motor”.

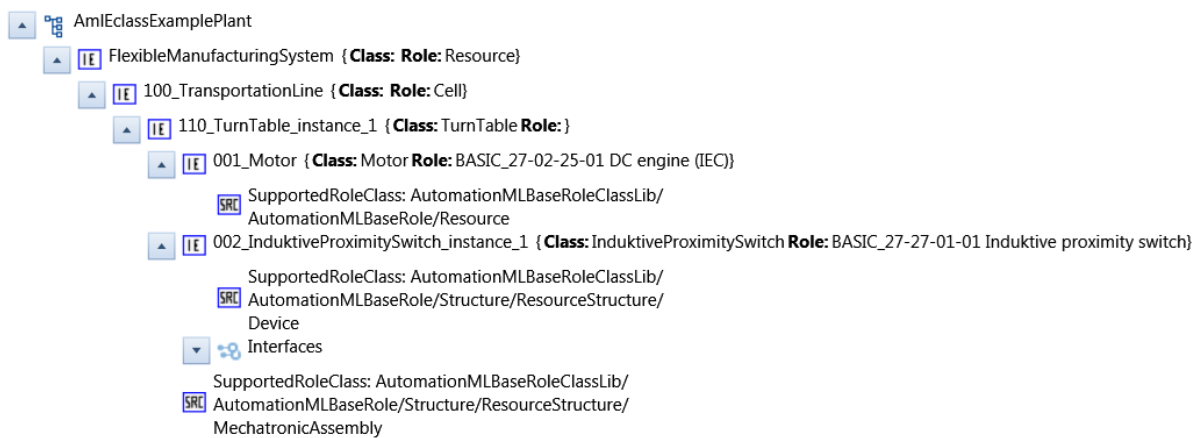


Figure 41 – Example of instance hierarchy

Figure 42 depicts the same IH in XML representation.


```

<InstanceHierarchy Name="AmlEclassExamplePlant">
  <InternalElement Name="FlexibleManufacturingSystem" ID="b52e1030-bc63-4fbd-a628-0b6c6640009a">
    <InternalElement Name="100_TransportationLine" ID="f5f57ce0-e7c0-49b0-83c1-6ea6f4e65d42">
      <InternalElement Name="110_TurnTable_instance_1"
        RefBaseSystemUnitPath="AmlEclassExampleSystemUnitClassLib_PlantComponents/TurnTable"
        ID="eacff900-918e-4c75-8311-584c9dc168f2">
        <InternalElement Name="001_Motor"
          RefBaseSystemUnitPath="AmlEclassExampleSystemUnitClassLib_PlantComponents/Motor"
          ID="de53ae62-fb0c-4f93-b5de-f32ce3e9b4eb">
          <SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
          <RoleRequirements RefBaseRoleClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
            27_Electric engineering, automation, process controll engineering /
            27-02_Electrical drive/27-02-25_DC engine/
            27-02-25-01_DC engine (IEC)/BASIC_27-02-25-01_DC engine (IEC)" />
          </InternalElement>
          <InternalElement Name="002_InduktiveProximitySwitch_instance_1"
            RefBaseSystemUnitPath="AmlEclassExampleSystemUnitClassLib_PlantComponents/InduktiveProximitySwitch"
            ID="ed77a2d8-b1a1-4532-9393-f07f801f4bce">
            <ExternalInterface Name="SensorValue_Pin1"
              RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication/SignalInterface"
              ID="5e84e31b-c3b6-4390-b4b2-5bb2442a50b9" />
            <SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/
              Structure/ResourceStructure/Device" />
            <RoleRequirements RefBaseRoleClassPath="ExampleEClassRoleClassLib/eClassClassSpecification/
              27_Electric engineering, automation, process controll engineering /
              27-27_Binary sensor technology, safty-related sensor technology/27-27-01_Proximity switch/
              27-27-01-01_Induktive proximity switch/BASIC_27-27-01-01_Induktive proximity switch" />
            </InternalElement>
            <SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/
              ResourceStructure/MechatronicAssembly" />
            <InternalElement>
            <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ResourceStructure/Cell" />
            </InternalElement>
            <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
            </InternalElement>
          </InternalElement>
        </InternalElement>
      </InternalElement>
    </InternalElement>
  </InstanceHierarchy>

```

Figure 42 – XML representation of the example instance hierarchy

6.3 Generation process for eCl@ss advanced AutomationML model

This generation process will be defined in a later version of this whitepaper.

Literature

- [1] AutomationML e.V.: AutomationML Whitepaper Part 1 - Architecture and general requirements, [https://www.automationml.org/o.red/uploads/dateien/1375858464-AutomationML Whitepaper Part 1 - AutomationML Architecture V2.2.pdf](https://www.automationml.org/o.red/uploads/dateien/1375858464-AutomationML%20Whitepaper%20Part%201%20-%20AutomationML%20Architecture%20V2.2.pdf), (last access August 2014).
- [2] AutomationML e.V.: AutomationML Whitepaper Part 2 - Role class libraries, [https://www.automationml.org/o.red/uploads/dateien/1375858483-AutomationML Whitepaper Part 2 - AutomationML Libraries_v2.2.pdf](https://www.automationml.org/o.red/uploads/dateien/1375858483-AutomationML%20Whitepaper%20Part%20-%20AutomationML%20Libraries_v2.2.pdf), (last access August 2014).
- [3] AutomationML e.V.: AutomationML Whitepaper Part 3 - Geometry and Kinematics, [https://www.automationml.org/o.red/uploads/dateien/1378299113-AutomationML Whitepaper Part 3 AutomationML GeometryV2.2.pdf](https://www.automationml.org/o.red/uploads/dateien/1378299113-AutomationML%20Whitepaper%20Part%203%20AutomationML%20GeometryV2.2.pdf), (last access August 2014).
- [4] AutomationML e.V.: AutomationML Whitepaper Part 4 - Logic Description, [https://www.automationml.org/o.red/uploads/dateien/1375858589-AutomationML Whitepaper Part 4 - AutomationML Logic Description v2.pdf](https://www.automationml.org/o.red/uploads/dateien/1375858589-AutomationML%20Whitepaper%20Part%204%20-%20AutomationML%20Logic%20Description%20v2.pdf), (last access August 2014).
- [5] J. Prinz: Formalization of object constraints in AutomationML, 3d AutomationML user conference, October 7th. 2014, https://www.automationml.org/o.red/uploads/dateien/1417687897-AutomationML_ConferenceProceedings_2014.zip.