GUIDELINE

HOW TO CREATE A SUBMODEL TEMPLATE SPECIFICATION

December 2022

Imprint

Publisher
Industrial Digital Twin Association
Lyoner Strasse 18
60528 Frankfurt am Main
Germany
https://www.industrialdigitaltwin.org/

Version history

Date	Version	Changes made
2022-12-05	1st Version	Release of the Guideline

Contents

Figures

Tables

1 General

1.1 Overview

This document is an enabler for the IDTA specification series of Submodel template specifications. Each part of the mentioned series specifies the contents of a Submodel template (SMT) for the Asset Administration Shell (AAS). The the Asset Administration Shell is described in [1], [2], [3] and [6]. First exemplary Submodel contents were described in [4] and [8], while the actual format of this document was derived by the "Administration Shell in Practice" [5].

The IEC working group IEC TC65 WG24 aims at publishing an international standard [7]. Figure 1 illustrates the AAS and Submodel template guiding the creation of Submodels. SubmodelElements and Submodel template elements may reference entries in concept repositories. The creators of Submodel template specification are designated as the Asset Administration Shell responsibles.

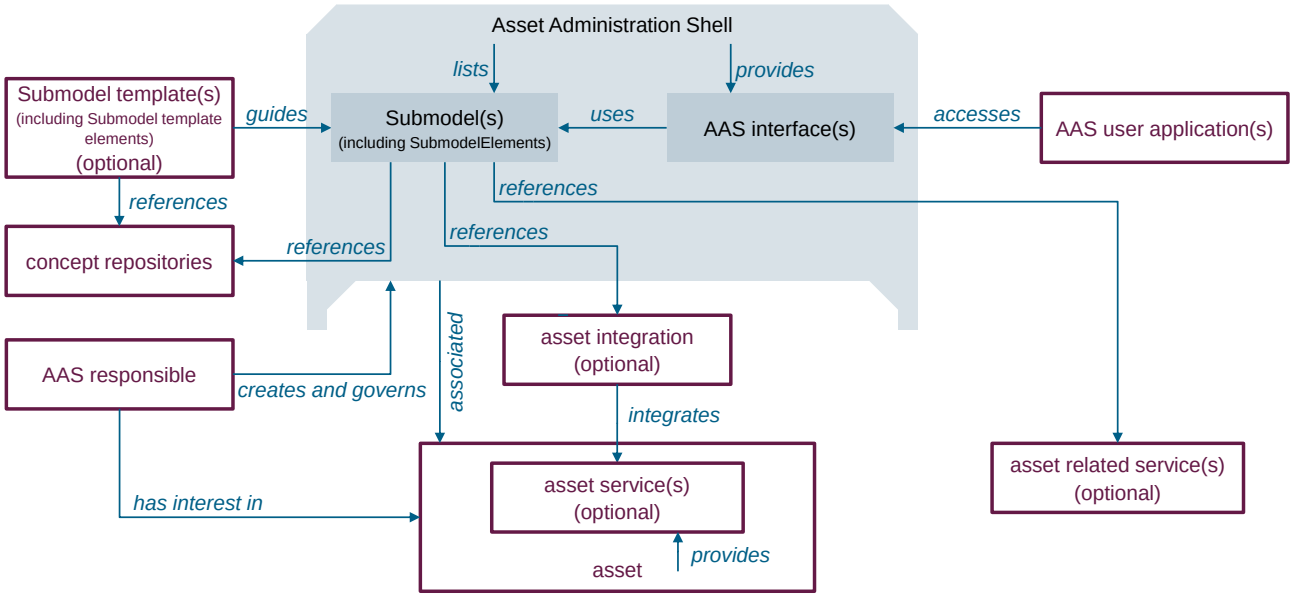

Figure 1 – Detailed overview of Asset Administration Shell and related roles [7]

Figure 2 illustrates two the Asset Administration Shell user applications, to use the interoperable Submodel information.

From the viewpoint of the meta-model [6], a Submodel template is a Submodel with kind = Template.

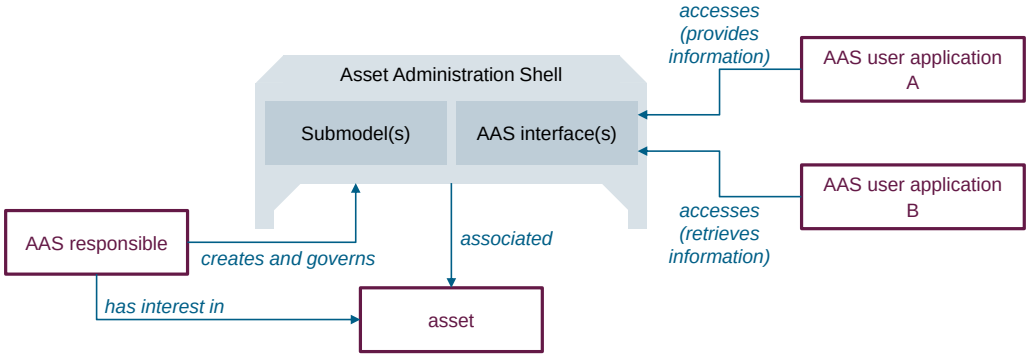

Figure 2 – Information exchange between AAS user applications [7]

This document facilitates the creation of Submodel template for the the Asset Administration Shell. These Submodel templates will allow the Asset Administration Shell responsibles to create and provide standardized models, which represent certain aspects of an asset. Specifically, this document describes how Submodel template specifications for created Submodel template shall be created in order to maintain a set of common features and structures. For this purpose, different workflows are described.

The Submodel template comprises computer-readable information, such as AASX package files, but also human-readable information. The latter can be described in form of a Submodel template specification.

Note: In future, the provision of human-readable information by means of markdown files in GitHub or such is foreseen.

1.2 Scope of this document

Scope of this document is to provide a uniform human readable and machine-readable way of a Submodel template. It describes the possible workflows and necessary working steps to create a Submodel template. It allows multiple Submodel template teams working in parallel on different Submodel template, while maintaining a common structure and required features. Different human stakeholds shall be able to understand information and services associated with the particular Submodel template and realized Submodel instances.

Not scope of this document is the IDTA process for registering Submodel template; this is described by [R1].

1.3 Stakeholders

Stakeholders for reading and applying this document are, among others:

- members of a Submodel template working team creating a Submodel template specification
- developers creating technical provisions dealing with specific Submodels and their subject matter
- subject matter experts and users required to 'fill out' Submodels
- architects of the Asset Administration Shell and working groups

1.4 Abbreviations

The following abbreviations are used in this document:

Table 1 – Used abbreviations

Abbreviation	Description
AAS	Asset Administration Shell
AASX	AASX file package
DOC	Word processor document
SM	Submodel
SMT	Submodel template
WD	Working draft
XMI	XML Metadata Interchange

1.5 Conventions

For some terms, a special notation is used, in order to distinguish them from English text. Defined in [6], these include:

- Asset Administration Shell (AAS)
- Qualifier
- Submodel
- SubmodelElement

Bibliography items are designated as reference such as [1], ressources are designated as reference such as [R1].

1.6 Meta model version

This document currently targets meta model version V3.0RC01. Submodels and SubmodelElementCollections are used frequently as means of hierarchical structuring SubmodelElements.

1.7 Further general information

No further general information is given, yet.

2 Format of a SMT document

2.1 General

The creation of a SMT specifcation has multiple deliveries (see 3.1), including the human-readable part of SMT, currently a word processor document. This human-readable part is called SMT document.

The deliverables shall include pure machine-readable AASX files and may also include samples and more (see Table 3).

2.2 Structure of the human-readable part of SMT

For the structure of the content for the human-readable part of SMT, the SMT document, a template is available at IDTA office.

- Cover page which allows a high degree of recognition of the SMT series, providing:
  - uniform cover page design
  - IDTA document number (see 2.3)
  - title of the SMT, often just called "the Submodel for..."
  - semantic version information (natural numbers for version, revision) (see 2.4)
- Imprint
- Version history, giving information to the public, allowing to understand the evolution of the SMT over multiple versions.
- Tables of contents, figures, tables.
- Tables and figures shall provide titles.
- Uniform scheme for clauses and subclauses, typically:

Table 2 – Common (sub-)sections of the human-readable part of SMT

(Sub-) Section	Aim
1 General	Provide general overview.
1.1 About this document	Provide very brief introduction to AAS and associated documents. Only referring to other documents. Typically taken unchanged from the template.
1.2 Scope of the Submodel	Give precise but concise definition of the scope, clarifying industry segements, stakeholders, life cycle steps, associated assets and subject matter to be represented.
1.3 Relevant standards	The use of existing standards in encouraged.
2 Approach of the Submodel	Different subclauses explaining background and approaches, which lead to the current design of the SMT. Often, a UML diagram of the SMT entities is given. Often, a preview, how the subject matter could be presented to the user, is given, e.g. a screenshot of a plugin for the AASX Package Explorer, or inside an authoring system.
3 Submodel and SubmodelElements	Detailed description of the different entities by the means of the defined table format.
4 {Further normative}	Further clauses with normative content.
Appendix A - Explanations on used table formats	To allow the understanding of the definitions without extensive referring to other documents this most important information shall be given in every SMT.
Appendix B - Bibliography	If possible, refer to more extensive material instead of re-describing.

Appendix C - List of authors	IDTA depends on agile efforts of many volunteers. Public appreciation could and should be given.
Closing section	Link to IDTA.

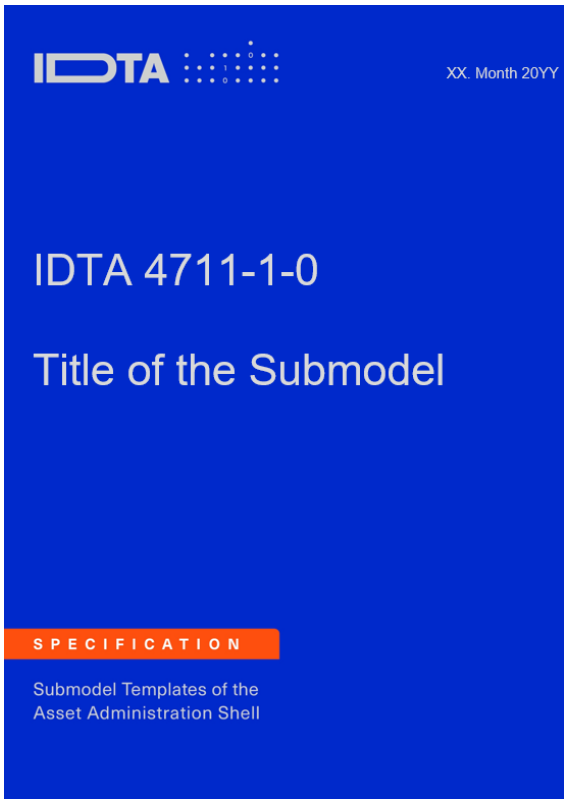

Figure 3 – Sample cover page of a SMT document

2.3 IDTA document number

The IDTA document number is retrieved from the IDTA office. The number for SMT starts at the 2000 number range (the 1000 - 1999 number range is reserved for meta models). It is possible to apply the IDTA office to reserve a range of document numbers, e.g. for future extensions of the described subject matter.

2.4 Semantic version information of the SMT

The administrative information of identifiables in the AAS entities comprise two attributes: version and revision [6]. The version and revision are also indicated by the IDTA document number (e.g. IDTA 4711-1-0 stands für the SMT IDTA 4711 version 1 revision 0). For the SMT, this is seen as semantic versioning:

- The version designates the major version of the SMT. An increment typically indicates a breaking change, requiring an adoption of the information by a human.
- The revision designates a minor version of the SMT. An increment typically designates recognizable changes, fixes and feature improvements, which are not breaking changes. A revision requires a version. This means, if there is no version there is no revision neither.

3 Workflows

3.1 General

In this section, multiple possible workflows are being described. Any workflow might be executed by an architect, however, reviews within the SMT team and with possible domain experts are considered essential. The workflow and the achieved results shall comply to the IDTA Process Description [R1].

3.2 Arctifacts to be delivered

For the "(6) Review" phase of the process [R1] and for publication by IDTA, the follwing artifacts are mandatory/optional.

Table 3 – Artifacts of a SMT Specification

Artifact	Description	Cardinality
SMT specification	Word processor document, representing the SMT specification at a whole	One
Pure SMT AASX file	AASX file for the SMT, containing a Submodel and SubmodelElements of kind = Template. No Qualifiers according 7.2, 7.3 shall be attributed.	One
Qualified SMT AASX file	An AASX as above, but SubmodelElements are attributed with Qualifiers, such as described in 7.2, 7.3.	ZeroToOne
Example AASX file(s)	AASX files with exemplary AAS, Submodels and SubmodelElements of kind = Instance, demonstrating the purpose of the SMT	ZeroToMany
Generic forms preset(s)	Option file(s) for the AasxPluginGenericForms in JSON format to allow easy creation and filling out of SMTs	ZeroToMany
Source format file for given figures	If the SMT specification uses figures for illustrations, the source format files shall be given for later maintenance.	ZeroToMany

3.3 Document driven workflow

The document driven workflow is rather simple and can be executed without very much invocation of advanced tools. It is suitable for rather simple Submodels. Figure 4 demonstrates the workflow.

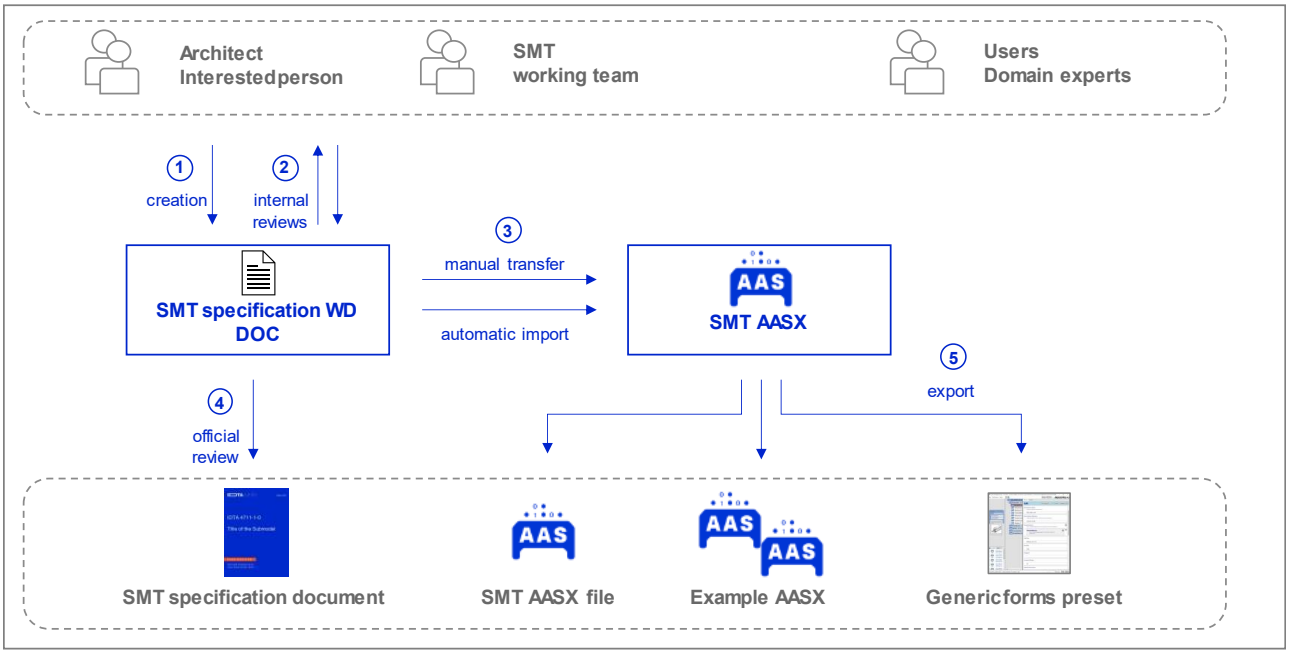

Figure 4 – Document driven workflow

The specification document is in the center of activities. The following working steps can be distinguished:

(1) Some member of the Submodel working team, e.g. the architect or the interested person [R1], create an initial version of the working draft (WD) of SMT specification document. This is a word process document (DOC), following the templateof the IDTA.

The template is filled out with all structural relevant text section as described in 2.2.

The requisites of the process "(5) Designing a Submodel b)" [R1] are considered.

(2) Frequent reviews within the working team are executed, using just the actual status of the working draft.

If more extensive reviews of the working draft are required, e.g. together with another working group, then a line numbered PDF is recommended and commenting via the IDTA comments template [R2] is recommended.

(3) If the discussions within the working team come to its conclusions and the "Designing a Submodel" phase [R1] is concluded, a SMT AASX file shall be created (SMT AASX).

This can be done either by manually editing the AASX, e.g. using the AASX Package Explorer [R3].

Or an (semi-) automatic import can be facilitated, e.g. by AASX Package Explorer, menu option "File / Import / Import Submodel from table".

(4) For approaching the official review, "(6) Review" phase [R1] is initiated, by preparing the final version of the working draft (WD) and handing over to architect and IDTA office.

These persons will follow on with the process, which will finally lead to a publication via IDTA GitHub and IDTA homepage.

(5) Using the SMT AASX model, the final SMT AASX file can be defined. Examples can be generated, using the action "Submodel / Turn to kind instance" [R3]. If adequate, a Generic forms preset can be

done by defining the Qualifiers as in 7.3 and exporting via "File / Export / Export Submodel as options for generic forms" [R3]. This preset might be used to easily generate more example AASX files. All these files shall be handed over to the architect and the IDTA office, as well.

3.4 Model based workflow

The model-based workflow is suitable for complex subject matters and working teams looking deeply into particular aspects and capabilities of the resulting AASX model. Figure 5 demonstrates the workflow.

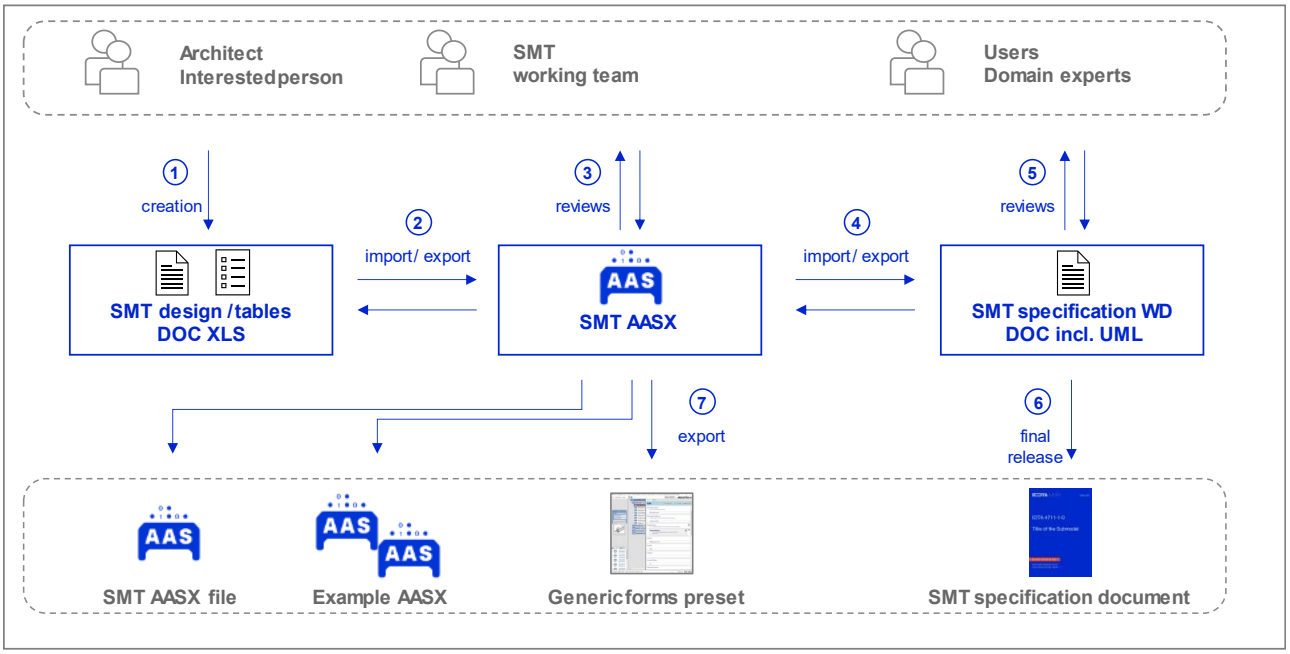

Figure 5 – Model based workflow

The AASX model of the SMT is in the center of activities. The following working steps can be distinguished:

(1) Some members of the Submodel working team, e.g. the architect or the interested person [R1], create an initial design approach and partitioning into multiple tables of SubmodelElements.

It is also recommended to formulate the scope of the Submodel, e.g. during the kick off the working team.

Table formats can be DOC or XLS, multiple tables can be parsed in a single document. These tables can also be easily reviewed with external persons, as well.

(2) Early in time, these tables are imported into an SMT AASX. The AASX Package Explorer [R3] provides such functions under "File / Import / Import from table", see [R5].

By the same mechanism, tables can be exported back via "File / Export / Export to tables" to form a SMT design roundtrip.

(3) Early reviews with the group can be executed direct by working on the SMT AASX model.

The AASX model can be easily turned into Example AASX by using the action "Submodel / Turn to kind instance" [R3] to testdrive the filling in with exemplary data.

Or, by defining Qualifiers as in 7.3 and using "File / Export / Export Submodel as options for generic forms" [R3], a Generic forms preset can be generated, which could be handled to multiple people to test-drive the SMT.

Or, by using "File / Export / Export Submodel as snippet for PredefinedConcpets" [R3], source code for predefined concepts could be generated to be used for programmatic test exports of exemplary Submodel contents.

(4) Using the SMT AASX as a turntable, imports/ exports can also be done with respect to the working draft (WD) of SMT specification document.

Also, the scope and further definitons might be integrated already into the document.

The required tables can be directly exported [R3].

Additionally, UML can be generated by "File / Export / Export Submodel as UML" [R3] (see clause 4).

(5) Using the SMT specification working draft (WD), together with the SMT AASX model, reviews with the SMT working team, users and domain experts can be facilitated.

If more extensive reviews of the working draft are required, e.g. together with another working group, then a line numbered PDF is recommended and commenting via the IDTA comments template [R2] is recommended.

(6) For approaching the official review, "(6) Review" phase [R1] is initiated, by preparing the final version of the working draft (WD) and handing over to architect and IDTA office.

Architect and IDTA office will follow on with the process, which will finally lead to a publication via IDTA GitHub and IDTA homepage.

(7) Using the SMT AASX model, the final SMT AASX file can be defined.

At least one Example AAS shall be generated, e.g. by using the action "Submodel / Turn to kind instance" [R3].

If adequate, a Generic forms preset can be realized by defining the Qualifiers as documented in clause 7 and exported (see clause 5.3). This preset might be used in turn to easily generate more Example AASX files.

All generated files shall be handed over to the architect and the IDTA office, as well.

3.5 Semantic driven workflow

The semantic based workflow is especially suitable for complex subject matters with no existing semantic definitions available for the subject under consideration. In the case of the semantic definition of a submodel this also includes structural information. The workflow can also be applied in cases there is already a semantic definition available for the submodel but in a different machine-readable format. Figure 6 demonstrates the workflow.

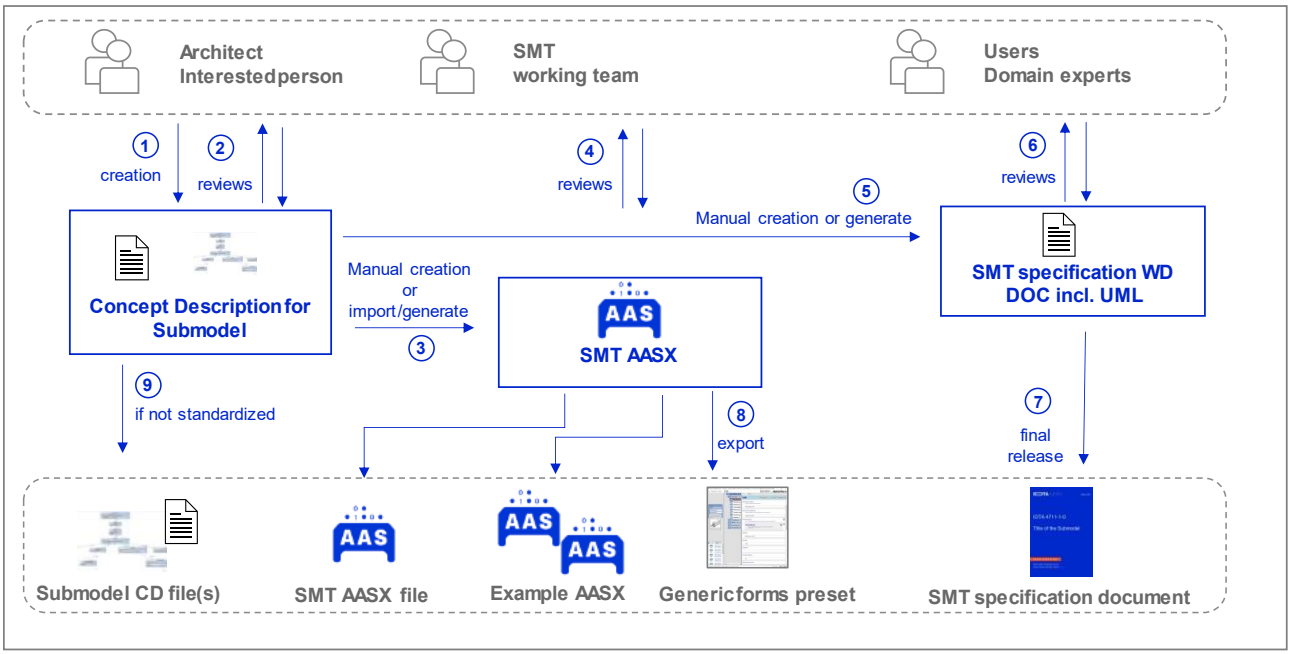

Figure 6 – Semantic driven workflow

The semantic definition of the submodel itself is in the center of activities. A semantic definition of a submodel is typically a model itself, in short such a model is sometimes called a semantic model. The following working steps can be distinguished:

(1) There are two ways to execute step (1):

(a) there is already an **existing open semantic definition** available for the submodel under consideration in some standardized way, for example an application class in a dictionary like ECLASS or IEC CDD, or an existing W3C ontology or an existing semantic model of CATENA-X etc. In this context a semantic definition can only be directly used if this semantic definition has a globally unique identifier. Otherwise, if for example an IEC or ISO standard or an OPC UA companion specification exists but not with unique identifiers for the model then Step (b) needs to be followed.

(b) **no suitable open semantic definition is available** for the submodel under consideration. In this case some member of the Submodel working team, e.g. the architect or the interested person [R1], creates an initial semantic definition or model in an appropriate format and with appropriate globally unique concept identifiers (semanticIds), e.g. BAMM [R9] or ECLASS fast track[1]. In a later stage external standardization of this submodel template specific semantic definition in standards development organizations (SDOs) like ECLASS, IEC etc. can and should be planned.

[1] see https://www.eclass.eu/fileadmin/downloads/application-documents/ECLASS_terms-of-use_4-2_en.pdf

Note: This approach also might start with Excel or UML diagrams or any other supporting material before doing the first semantic definition of the submodel, similar to what is described in chapter 3.4.

(2) Frequent reviews are executed with the domain experts and stakeholders.

(3) As soon as the semantic definition has a mature state, an AASX file is created. In the basic approach this is done manually. In an advanced approach a corresponding importer or generator is available that creates the AASX. Example: for ECLASS there is an importer available in the AASX Package Explorer that creates a corresponding submodel. Semi-automatic approaches might also be supported.

(4) Internal reviews ensure that the created AASX file is correct.

(5) Then a corresponding textual specification of the submodel template needs to be created. Again, this is either done manually or this is generated (semi-)automatically.

(6) Again, internal reviews ensure that the created SMT Specification WD document is meeting the requirements.

(7) After the SMT Specification WD document is available an official review can be started.

(8) If the SMT Specification can be released after the findings of the review are incorporated the release is prepared containing the identified deliveries as specified (see 3.2).

(9) If no existing open semantic definition for the submodel under consideration is available (see step (1) b) then also the artifacts used to describe the semantic model in a machine-readable way – if available - shall be added to the release for future maintenance and reuse.

4 UML generation

4.1 General

Having an SMT AASX file available, Unified Modeling Language (UML) can be generated easily via the AASX Package Explorer. This is achieved using "File / Export / Export Submodel as UML" [R3].

Precondition is, that cardinalities of the different SubmodelElement are designated via the Qualifier "Multiplicity", according to clause 7.2. UML can be generated for Submodels of kind = Template or kind = Instance, even with (example) values attached.

4.2 UML design style

Unified Modeling Language (UML) is highly specified, general-purpose, modeling language in the field of software engineering and is used for many purposes. One purpose is illustrating the structure of AAS elements given by a SMT (SMT) specification. For this purpose, within such specification, the following provisions are set:

(1) The structure of AAS element shall be represented by a class diagram, which is titled according to the name of the SMT specification.
(2) AAS elements providing childs (such as SubmodelElementCollection, Operation and more) shall be represented by UML class elements, with the AAS element type or its abbreviation according [6] set as stereotype.
(3) Such childs (of complex elements such as SubmodelElementCollection, Operation and more) shall be represented as attributes within the respective UML class.
(4) Such attributes shall be marked as public ('+') and shall feature the AAS element type or data type or its abbreviation according [6].
(5) If the cardinality (multiplicity) of such an attribute is other than '[1]', one of the following cardinality notions shall be used: [0..1], [0..*], [1..*].
(6) If such attribute represents an AAS element providing childs (such as SubmodelElementCollection, Operation and more), and this AAS element is represented by the respective class diagram as well, then an aggregation association (filled diamond arrow) shall be expressed, with the name of the AAS element given by the association and its cardinality at the 'part' end of the association.

4.3 UML via XMI export

Figure 7 shows the result of export UML to XMI 2.1 format via the AASX Package Explorer. The XMI does not contain the design of a diagram, so this must be done using an UML authoring tool, such as Enterprise Architect. An intended layout can be achieved.

Note 1: As of Jan 2022, the directionality is not exported correctly. So, direction of each association needs to be set to unspecific, in order to render aggregation associations in a correct way.
Note 2: The UML authoring tool should be able to export vector graphics in order to allow good scaling for publication. In Enterprise Architect this is achieved by "Start / Preferences / General / Clipboard Format: Metafile" and "Publish / Save Image / Save to Clipboard" (V13).
Note 3: The shown example required about 10 min design time.

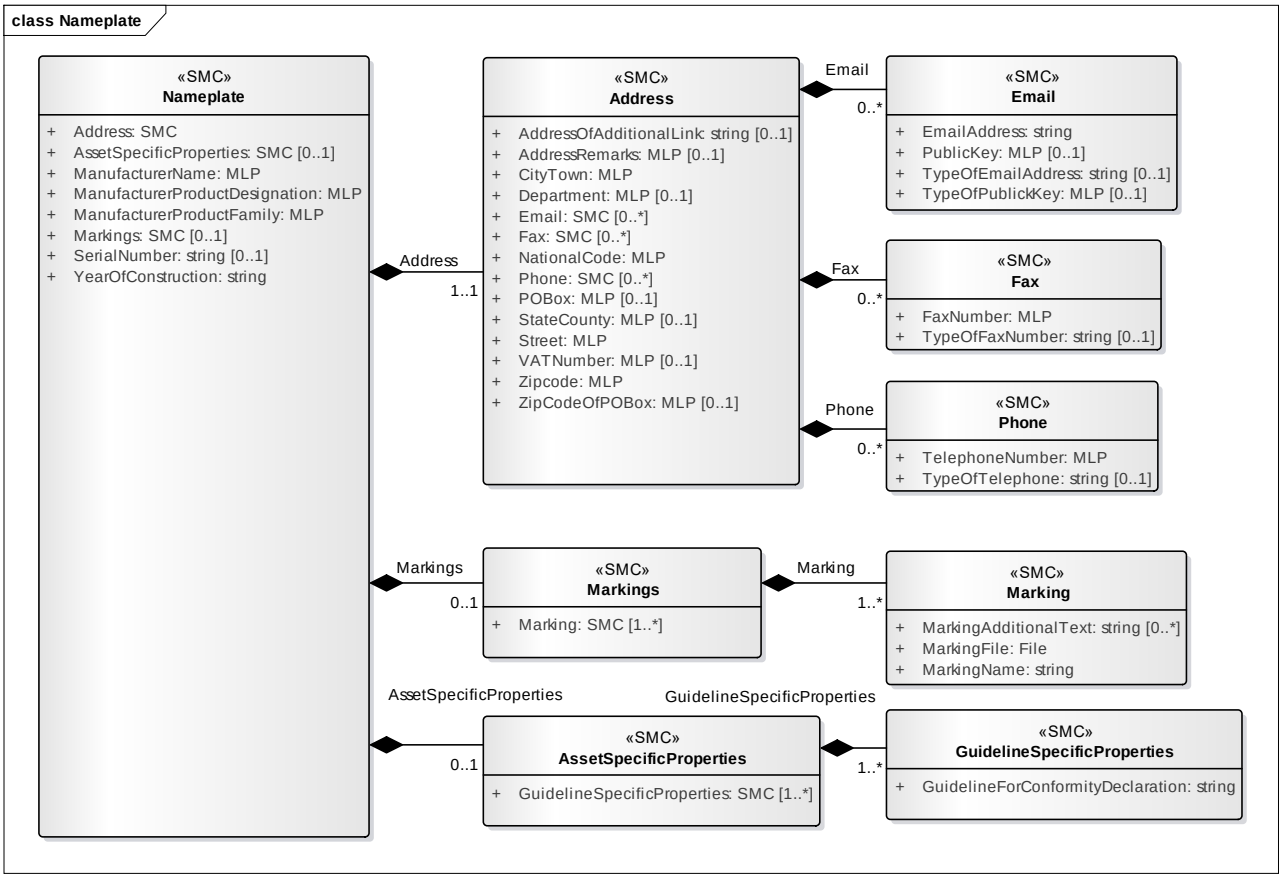

Figure 7 – Example UML generation by exporting XMI and manually layouting in UML authoring tool

4.4 UML via PlantUML export

Figure 8 shows the result of export UML to PlantUML format via the AASX Package Explorer. The file contents can be copied to the clipboard, as well, and directly pasted to [R7].

Note 1: The generated UML can be slightly adjusted, but not manually layouted, by some options. In the example, the line "mainframe SMT Nameplate" introduced a frame around the diagram.
Note 2: The example file was saved as SVG, which allows a vector scaling for publication.
Note 3: The shown example required about 15 sec design time.

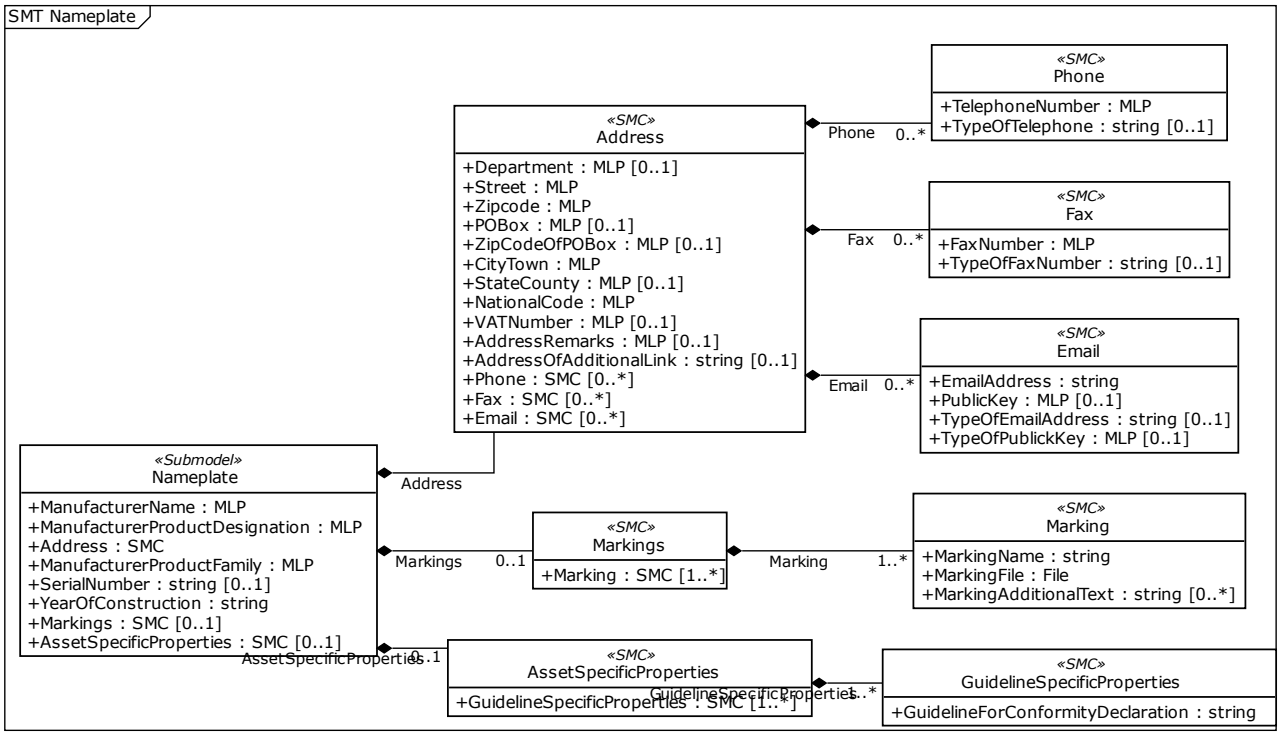

Figure 8 – Example UML generation by exporting to PlantUML and automatic layouting

5 Generic forms preset

5.1 General

The AASX Package Explorer [R3] allows using the plugin AasxPluginGenericForms to assist users in creating Submodels based on SMTs. The plugin displays a visual assistant to easily create a Submodel instance according to a SMT and fill out this instance with data. This allows also non-expert users to provide subject matter information to the AAS.

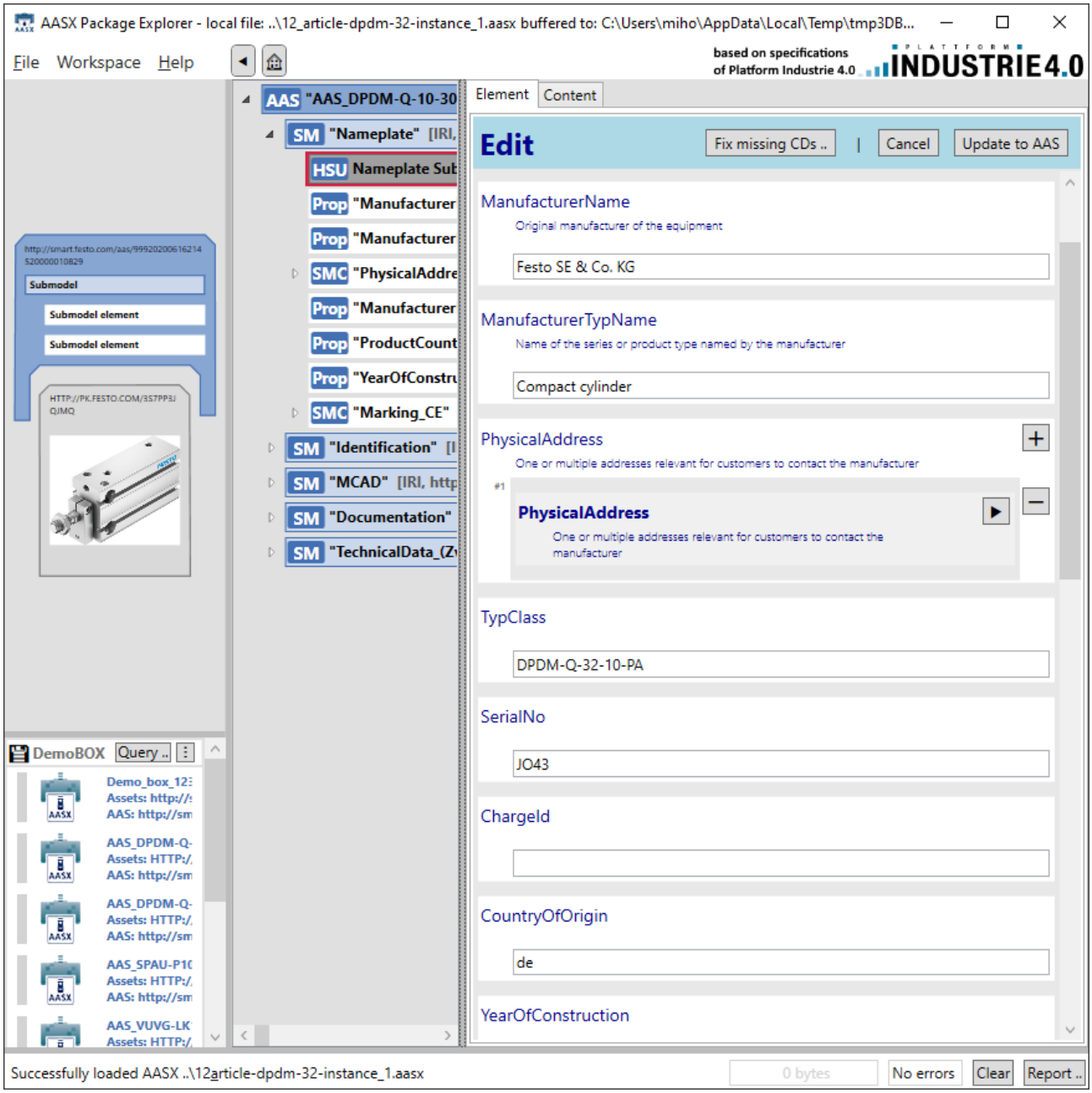

Figure 9 – AASX Package Explorer offering the easy filling out of Submodel "Nameplate"

5.2 Attribution of a SMT

The different SubmodelElements of a SMT need to be attributed by Qualifers in order to provide enough information for an adequate rendering of generic forms. These Qualifiers are described in clause 7.3.

5.3 Exporting options file

The SMT migt be exported via "File / Export / Export Submodel as options for generic forms" [R3]. This leads to the creation of a file "*.add-options.json", which can be used by other users to automatically generate and fill out Submodels.

5.4 Usage of generic forms in AAS user applications

In order to use a generic forms preset in AASX Package Explorer [R3], the user needs to locate the ".\plugins\AasxPluginGenericForms" folder and copy the respective "*.add-options.json" file into this folder (see Figure 10). After re-start of the application, using "Workspace / Plugins / New Submodel", a new Submodel can be created.

The architect and IDTA office will also check, if the Submodel and its generic forms preset can be included into the standard deployment of the AASX Package Explorer.

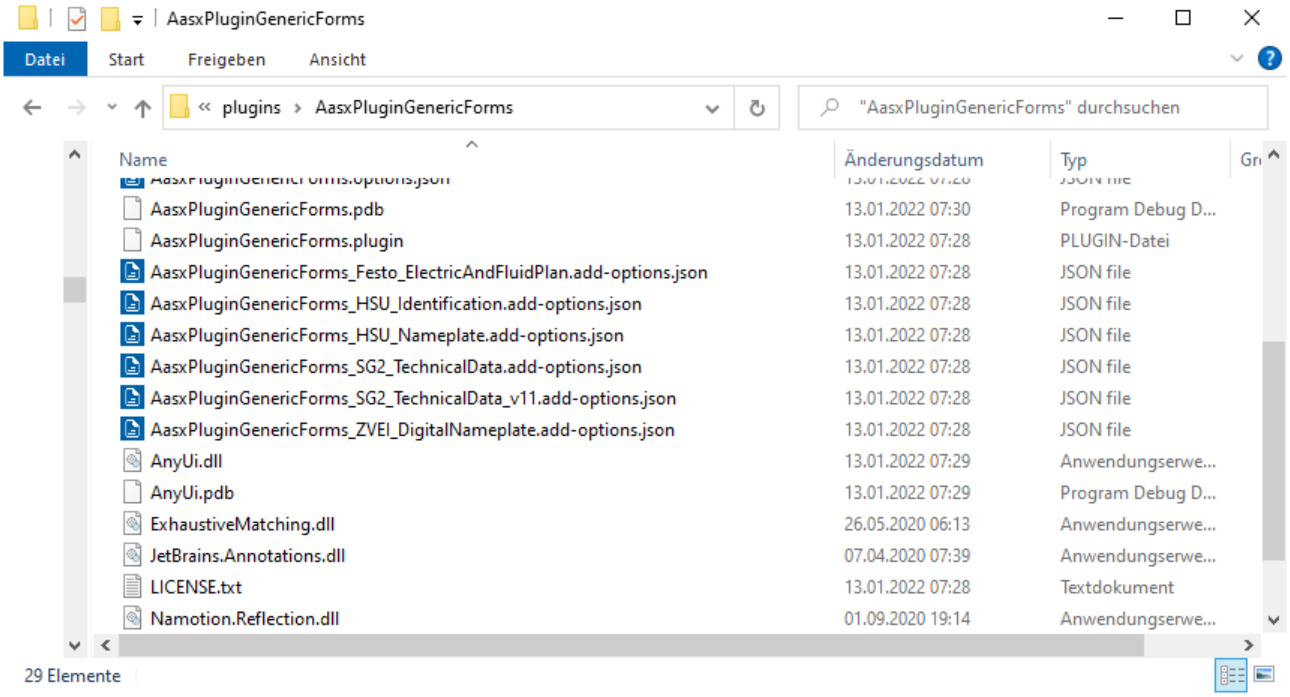

Figure 10 – Plugin folder for Generic forms

6 Table format for Submodels and SubmodelElements

6.1 General

The SMT document shall provide a set of tables for the Submodels and SubmodelElements, which are specified by the SMT. In the SMT, these AAS elements shall be of kind = Template. The tables are designed to provide an overview on these AAS elements, allowing all stakesholders (see 1.3) to understand the design of the particular SMT.

The described table format is not self-explanatory. Each SMT document shall feature an appendix, explaining the table format (see Annex A). This appendix is provided by the template document of the IDTA.

6.2 Table heads

The SMT document shall feature a separate table for each AAS element, which is specified by the SMT and which hierarchically specifies child AAS elements (consequently named "AAS element with childs"). Typically, but not exclusively, these are Submodels and SubmodelElementCollections (see 1.6), but can be also Operations or Entities.

Figure 11 – Format of table heads

Figure 11 shows the format of the heading part of such table. For the different rows, the following provisions are given by Table 4:

Table 4 – Format of table heads

ID	Label	Provision
A	idShort:	This cell shall contain the idShort of the AAS element with children. Could be an idShort with enumeration template such as "{000}" or "{00}" (see 7.4). If the AAS element is used multiple times as child of a parent AAS element, then a list of comma separated idShorts can be given. Must not contain further information, such as "Note: the above idShort shall always be as stated."
B	Class:	This cell shall contain the AAS element type of the given AAS element with children. Can be: Submodel, SubmodelElementCollection, Operation, Entity, AnnotatedRelationshipElement.
C	semanticId:	This cell shall contain the idShort of the AAS element with children. Headed either by [IRI] or [IRDI].
D	Parent:	This cell shall contain the idShort of the parent of the AAS element with children. If multiple parents in the SMT use it, then a list of comma separated idShorts can be given. Must not contain further information, such as notes or annotations.
E	Explanation	This cell shall contain an explanation, which should be identical to the english language of the description of the AAS element with children.

Note: It is strongly recommended to use exactly the rows and columns as described. The import tools of AASX Package Explorer rely on this.

6.3 Table row items

Each table for AAS elements with childs features multiple row items, one per child of he AAS element. Such row item shall be exactly one table row in the word processor document. In order to accomodate sufficient information for the stakeholders, each cell of the row item might contain multiple information, delimited by line breaks.

Note: This row format simplifies the manual editing of tables, such as described in the document driven workflow (see 3.3).
Note: It is strongly recommended to use exactly the rows and columns as described. The import tools of AASX Package Explorer rely on this.

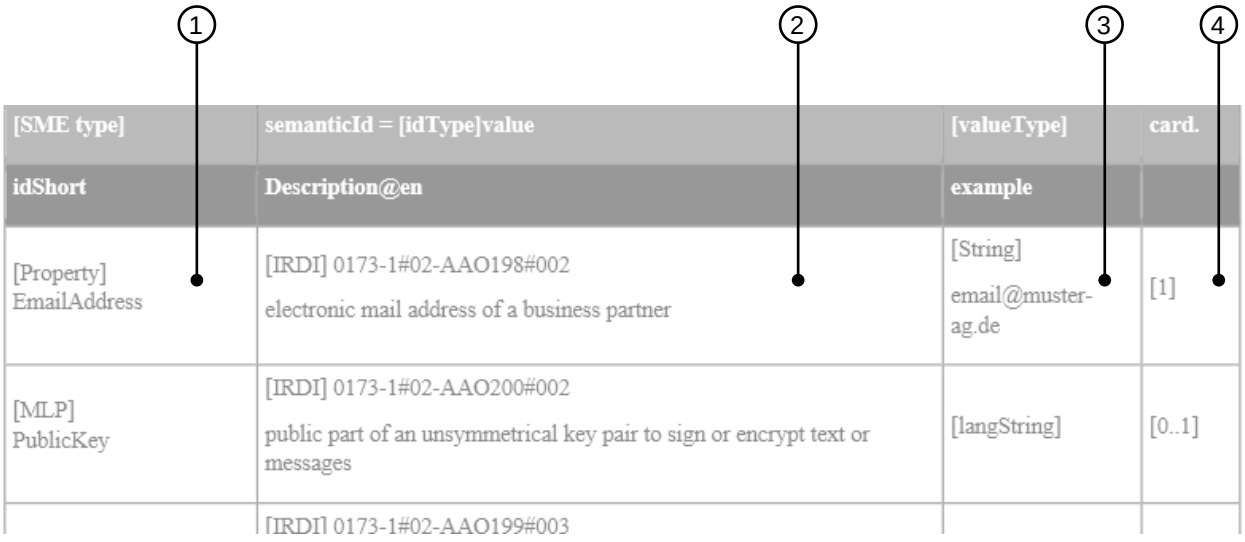

Figure 12 – Format of table row items

Figure 12 shows the format of the individual row items of such table. For the different row items, the following provisions are given by Table 5:

Table 5 – Format of table row items

ID	Label	Provision
1	[SME type] idShort	In the first line, the type of the AAS element shall be given. Abbreviation according [6] is allowed. The information shall be sorrounded by '[ ]'. In the second line, the idShort of the AAS element shall be given. Could be an idShort with enumeration template such as "{000}" or "{00}" (see 7.4).
2	semanticId = [idType]value Description@en	In the first line, the semanticId of the AAS element shall be given. Headed either by [IRI] or [IRDI]. In the second line, the english language of the description of the AAS element shall be given. The AASX files for the SMT may specify further languages. If used to define ConceptDescription, this information shall be took over to the definition of the ConceptDescription, depending on its data specification template. In the following lines, more information can be given: A line ending with an '@' and ISO 639 language code in lowercase can be used to specify a further language for description/ definition. A line starting with "Note: " shall be considered as a note in the SMT document and is not taken over to description/ definition. A line starting with "Constraint: " shall state an textual constraint on the usage of the AAS element. It is not taken over to description/ definition.
3	[valueType] example	If the AAS element is a Property, the valueType shall be given in the first line. The information shall be sorrounded by '[ ]'. If the AAS element is a MultiLanguageProperty, '[langString]' shall be given in the first line. In the next line, one or more example values can be given. String values might end with an '@' and ISO 639 language code in lowercase.
4	card.	In the first line, the cardinality of the AAS element shall be given. This can be [1], [0..1], [0..*], [1..*].

After the table header, before the first row items, two table rows shall be given featuring the respective labels defined in
Table 5, as illustrated by Figure 12.

7 Qualifers and attributes of SMT elements

7.1 General

According IEC 63278-1, a SMT element specifies the structure for a SubmodelElement. Especially, it specifies, to which concept repository entry for an information, relation or service or hierarchical structure (so called purpose) the SubmodelElement is related to. For the AASX of SMT, each SMT element is a SubmodelElement with kind = Template and the relation is done via the semanticId of the SubmodelElement.

In order to express the particular purpose of the structure, attributes shall specify:

- Optional & Cardinality
- Either-Or
- Example & Default & Initial Values (Here: Value used for Example Value)
- allowed ranges of properties (e.g. temperature in range -60 to +200 Celsius)
- regular expressions for naming (e.g. Document{00}) and for allowed values of properties
- required languages for multi language properties
- user access mode (read-write or read-only)

These attributes might be specified by dedicated means of ConceptDescriptions (e.g. see the BAMM model) or by means of Qualifiers [6] (see below).

Furthermore, the AASX Package Explorer allows using the plugin AasxPluginGenericForms to assist users in creating Submodels based on SMTs. For this purpose, further attributes are provided here (see below).

7.2 Qualifiers controlling the structure

The following Qualifers may be used, to allow a SMT element controlling the generation of instantiated SubmodelElements (see Table 6). These Qualifiers are available as presets for the AASX Package Explorer [R3].

Table 6 – Qualifiers controlling the structure

Qualifier/type*[2] (grey = legacy)[3]	Qualifier/semanticId* Description	Qualifier/value[4]*
SMT/Cardinality Multiplicity	https://admin-shell.io/SubmodelTemplates/Cardinality/1/0 This Qualifier allows to specify, how many SubmodelElement instances of this SMT element are allowed in the actual collection (hierarchy level of the Submodel). Note: All SubmodelElement instances need to have an unique idShort. For this, a template string can given in the idShort of the SMT element (see 7.4).	Allowed: One ZeroToOne ZeroToMany OneToMany

[2] For *, conver to the meta model description in [6]

[3] In grey, legacy Qualifier names are indicated

[4] Allowed values and/ or example values are given

SMT/EitherOr	https://admin-shell.io/SubmodelTemplates/EitherOr/1/0 The Qualifier.value defines an id of an equivalence class. Only ids in the range [A-Za-z0-9] are allowed. If multiple SMT elements feature the same equivalence class, only one of these are allowed in the actual collection (hierarchy level of the Submodel).	Examples: 1 LOLLIPOP
SMT/InitialValue	https://admin-shell.io/SubmodelTemplates/InitialValue/1/0 Specifies the initial value of the SubmodelElement instance, when it is created for the first time.	Example, e.g. for property "Working days per week": 5
SMT/DefaultValue	https://admin-shell.io/SubmodelTemplates/DefaultValue/1/0 Specifies the default value of the SubmodelElement instance. Often, this might designate a neutral, zero or empty value depending on the valueType of a SMT element.	Examples: 0 ""
SMT/ExampleValue	https://admin-shell.io/SubmodelTemplates/ExampleValue/1/0 Specifies an example value of the SubmodelElement instance, in order to allow the user to better understand the intention and possible values of a SubmodelElement instance. Note: Multiple examples can be given by delimiting them by '|' In case of a translateable string (langString) the example value shall be an English example string. Alternative (to be decided): add suffix like @en to string to denote language.	Examples: 42 "Hello" 3.1415
SMT/AllowedRange	https://admin-shell.io/SubmodelTemplates/AllowedRange/1/0 Specifies a set of allowed continous numerical ranges. Note: Multiple ranges can be given by delimiting them by '|'. Note: A single range is defined by interval start and end, either including or excluding the given number. Note: Interval start and end are delimited by ','; '.' is the decimal point Note: '*' allows to enter the default value	Examples: [0,10] (0.0,9.9] *|[1,6] [2,3]|[6,7]
SMT/AllowedIdShort	https://admin-shell.io/SubmodelTemplates/AllowedIdShort/1/0 Specifies a regular expression validating the idShort of the created SubmodelElement instance. Note: The format shall conform to POSIX extended regular expressions.	Example: Title[\d{2,3}]

SMT/AllowedValue	https://admin-shell.io/SubmodelTemplates/AllowedValue/1/0 Specifies a regular expression validating the value of the created SubmodelElement instance in its string representation. Note: the format shall conform to POSIX extended regular expressions.	Example: (red|green|blue)
SMT/RequiredLang	https://admin-shell.io/SubmodelTemplates/RequiredLang/1/0 If the SMT element is a multi language property (MLP), specifies the required languages, which shall be given. Note: Multiple languages can be given by multiple Qualifiers. Note: Multiple languages can be given by delimiting them by '|' Note: languages are specified either by ISO 639-1 or ISO 639-2 codes.	Example: en|fr
SMT/AccessMode	https://admin-shell.io/SubmodelTemplates/AccessMode/1/0 Specifies the user access mode for SubmodelElement instance. When a Submodel is received from another party, if set to Read/Only, then the user shall not change the value	Allowed: Read/Write Read/Only

7.3 Qualifiers supporting the generic forms functionality

7.3.1 List of Qualifiers

The following Qualifers are defined to allow a SMT element controlling the presentation of forms. These Qualifiers are available as presets for the AASX Package Explorer.

Table 7 – Qualifiers supporting the generic forms functionality

Qualifier.name*	Qualifier.semanticId* Description	Qualifier.value*
FormTitle	https://admin-shell.io/SubmodelTemplates/FormTitle/1/0 Allows adding a speaking title to the edit field, which could give particular hints for filling out the value.	Please use these fields to describe your product
FormInfo	https://admin-shell.io/SubmodelTemplates/FormInfo/1/0 Allows adding a longer explanation to the edit field, which could give particular hints for filling out the value.	Name of the series or product type named by the manufactuer
FormUrl	https://admin-shell.io/SubmodelTemplates/FormUrl/1/0 Specifies a hypertext link to an external URL for further explanations of the SMT element. The link is offered to be activated by the user.	https://en.wikipedia.org/wiki/RGBA_color_model
Multiplicity	https://admin-shell.io/SubmodelTemplates/Multiplicity/1/0 This Qualifier allows to specify, how many SubmodelElement instances of this SMT element are allowed in the actual collection (hierarchy level of the Submodel). Note: All SubmodelElement instances need to have an unique idShort. For this, a template string can be given in the idShort of the SMT element (see 7.4) Note: Multiplicity is identical to SMT/Cardinality (see above).	One ZeroToOne ZeroToMany OneToMany
EditIdShort	https://admin-shell.io/SubmodelTemplates/EditIdShort/1/0 Specifies, if the user is able to edit the idShort of the SubmodelElement instance within the forms presentation. Note: For simple applications, a value of "False" is recommended. Note: If the designed form allows to create multiple collections, "True" could be an option fo the collection.	True False
EditDescription	https://admin-shell.io/SubmodelTemplates/EditDescription/1/0 Specifies, if the user is able to edit the description of the SubmodelElement instance within the forms presentation. Note: For simple applications, a value of "False" is recommended. Note: If the designed form allows to create multiple collections, "True" could be an option fo the collection.	True False
FormChoices	https://admin-shell.io/SubmodelTemplates/FormChoices/1/0 Allows multiple choices of value; list of values separated by semicolon ';'.	Apples; Pies; Peanuts

7.4 Enumeration template for speaking idShort designations

The idShort of each SubmodelElement shall be unique in its namespace, e.g. the containing collection. At the same time, many users want the idShort to be a speaking name. In order to facilitate the automatic generation of speaking names, a term such as "{000}" or "{00}" might be included in the idShort of an SMT element, which cardinality is other than "One". The number of zeros will correspond to the number of digits of a continous numerical index for idShort.

Example: If a Submodel might describe multiple books, and a single book is described by multiple properties collected in a SubmodelElementCollection (SMC) with parent as Submodel, the application will be as described in Figure 13 – Examplary template idShort attribute for "Book".

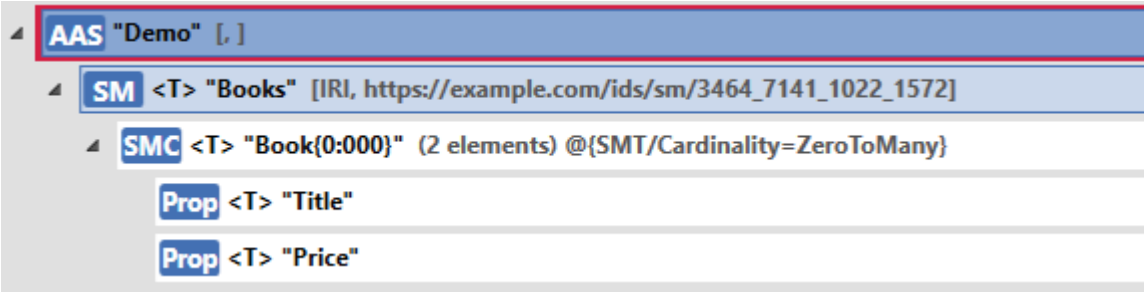

Figure 13 – Examplary template idShort attribute for "Book"

Note: In final version 3.0 of the meta model [6], SubmodelElementCollection is amended by SubmodelElementList. Consequently, above numbering scheme will not be required anymore.

8 ConceptDescriptions for SMT

8.1 General

The meta-model of the AAS and the AASX PackageExplorer [R3] allows expressing ConceptDescriptions conformant to IEC 61360, e.g., for Properties. Describing such ConceptDescriptions e.g., enables defining units of measure and is therefore important for practical application.

ConceptDescription	
Referable:	
idShort:	MinLoad
Identifiable:	
idType:	IRDI
id:	0173-1#02-BAB238#007
isCaseOf:	
reference[0]:	(GlobalReference) (no-local) [IRDI] 0173-1#02-BAB238#007
HasDataSpecification:	
HasDataSpecification (Reference):	
reference[0]:	(ConceptDescription) (local) [IRDI] 0173-1#02-BAB238#007
Data Specification Content IEC61360:	
preferredName:	[en] min. load [de] min. Bürde
unit:	Ω
unitId:	(GlobalReference) (no-local) [IRDI] 0173-1#05-AAA049#003
dataType:	REAL_MEASURE
definition:	[en] smallest load connected in series with an input circuit which ensures that the device operates within the specified accuracy limits [de] kleinste mit dem Eingangskreis hintereinander geschaltete Bürde, die den Betrieb innerhalb der angegebenen Genauigkeitsgrenzen sicherstellt

Figure 14 – Example of a ConceptDescription in AASX Package Explorer [R3]

The delivery of the SMT shall include definitions of ConceptDescriptions for all semanticIds referring to not publicly accessible dictionaries, such as ECLASS, IEC CDD or [R4].

Note: Please be aware that dictionaries above do not provide automatic retrieval (e.g. by REST interface); the automatic provision of units of measure is not possible unless ConceptDescriptions are provided within the AASX package.

Annex A. Explanations on used table formats

Note: This annex shall be included in each SMT document (see 6.1) and is included here for illustration.

1. General

The used tables in this document try to outline information as concise as possible. They do not convey all information on Submodels and SubmodelElements. For this purpose, the definitive definitions are given by a separate file in form of an AASX file of the SMT and its elements.

2. Tables on Submodels and SubmodelElements

For clarity and brevity, a set of rules is used for the tables for describing Submodels and SubmodelElements.

- The tables follow in principle the same conventions as in [5].
- The table heads abbreviate 'cardinality' with 'card'.
- The tables often place two information elements in different rows of the same table cell. In this case, the first information is marked out by sharp brackets [] form the second information. A special case are the semanticIds, which are marked out by the format: (type)(local[5])[idType]value.
- The types of SubmodelElements are abbreviated (among others):

Table 8 – Abbreviations of SubmodelElements

SME type	SubmodelElement type
Prop	Property
MLP	MultiLanguageProperty
Range	Range
File	File
Blob	Blob
Ref	ReferenceElement
Rel	RelationshipElement
SMC	SubmodelElementCollection

- If an idShort ends with '{00}', this indicates a suffix of the respective length (here: 2) of decimal digits, in order to make the idShort unique. A different idShort might be choosen, as long as it is unique in the parent's context.
- The Keys of semanticId in the main section feature only idType and value, such as: [IRI]https://admin-shell.io/vdi/2770/1/0/DocumentId/Id, as descibed in the serialization of the AAS meta model. The attributes "type" and "local" (typically "ConceptDescription" and "(local)" or "GlobalReference" and (no-local)") need to be set accordingly; see [6].
- If a table does not contain a column with "parent" heading, all represented attributes share the same parent. This parent is denoted in the head of the table.
- Multi-language strings are represented by the text value, followed by '@'-character and the ISO 639 language code: example@EN.
- The [valueType] is only given for Properties.

[5] "Attribute "local" was removed in V3.0RC01 [6]

Annex B. Resources

The following ressources are available to help developing SMT specifications. Ressources with heading backslash refer to the IDTA Teams drive.

[Resources used in the document]

Ressource	Description
[R1]	Process Description. Registration of Asset Administration Shell Submodel templates for Digital Twins. Industrial Digital Twin Association (IDTA). V1.0 https://industrialdigitaltwin.org/en/wp-content/uploads/sites/2/2022/01/2021-12-01_IDTA_Process-Submodels_V1.0.pdf
[R2]	IDTA comments template \IDTA - WG2 - Submodels\Submodel-Development\00_Templates\03 IDTA Submodel Spec Comments Template.xlsx
[R3]	AASX Package Explorer https://github.com/admin-shell-io/aasx-package-explorer/releases
[R4]	IDTA GitHub on Identifiers https://github.com/admin-shell-io/id
[R5]	Help file for [R3] to import/ export tables https://github.com/admin-shell-io/aasx-package-explorer/tree/master/src/AasxPluginExportTable/help
[R6]	Predefined concepts for AASX Package Explorer https://github.com/admin-shell-io/aasx-package-explorer/tree/master/src/AasxPredefinedConcepts
[R7]	PlantUML https://plantuml.io
[R8]	ISO/IEC 19505-2:2012, Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure
[R9]	BAMM (BAMM Aspect Meta Model) specification, Open Manufacturing Platform https://openmanufacturingplatform.github.io/sds-bamm-aspect-meta-model/bamm-specification/2.0.0-M1/index.html

Annex C. Bibliography

[1] "Recommendations for implementing the strategic initiative INDUSTRIE 4.0", acatech, April 2013. [Online]. Available: Recommendations for implementing the strategic initiative INDUSTRIE 4.0. Final report of the Industrie 4.0 Working Group - acatech - National Academy of Science and Engineering

[2] "Implementation Strategy Industrie 4.0: Report on the results of the Industrie 4.0 Platform"; BITKOM e.V. / VDMA e.V., /ZVEI e.V., April 2015. [Online]. Available: Implementation Strategy Industrie 4.0 (bitkom.org)

[3] "The Structure of the Administration Shell: TRILATERAL PERSPECTIVES from France, Italy and Germany", March 2018, [Online]. Available: https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/hm-2018-trilaterale-coop.html

[4] "Beispiele zur Verwaltungsschale der Industrie 4.0-Komponente – Basisteil (German)"; ZVEI e.V., Whitepaper, November 2016. [Online]. Available: Beispiele-Verwaltungsschale-Industrie-40-Komponente-White-Paper-Final.pdf (zvei.org)

[5] "Verwaltungsschale in der Praxis. Wie definiere ich Teilmodelle, beispielhafte Teilmodelle und Interaktion zwischen Verwaltungsschalen (in German)", Version 1.0, April 2019, Plattform Industrie 4.0 in Kooperation mit VDE GMA Fachausschuss 7.20, Federal Ministry for Economic Affairs and Energy (BMWi), Available: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/2019-verwaltungsschale-in-der-praxis.html

[6] "Details of the Asset Administration Shell; Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC01)", November 2020, [Online]. Available: Plattform Industrie 4.0 - Details of the Asset Administration Shell - Part 1 (plattform-i40.de)

[7] "IEC 63278-1 ED1: Asset Administration Shell for industrial applications – Part 1: Asset Administration Shell structure"

[8] "IDTA 02006-2-0 Digital Nameplate for industrial equipment", Version 2.0, 20. October 2022, Available: https://github.com/admin-shell-io/submodel-templates/tree/main/published/Digital%20nameplate/2/0

www.industrialdigitaltwin.org