Document Title	Standardization Template
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	535

Document Status	published
Part of AUTOSAR Standard	Foundation
Part of Standard Release	R20-11

Document Change History

Date	Release	Changed by	Description
2020-11-30	R20-11	AUTOSAR Release Management	• introduce advisory markup • editorial changes • Migration of document to standard FO
2019-11-28	R19-11	AUTOSAR Release Management	• harmonize the use of BlueprintCondition, FormalBlueprintGenerator • editorial changes • changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• uptraces wrt. life cycles • include ARMQL relevant parts • harmonize Blueprint parts
2017-12-08	4.3.1	AUTOSAR Release Management	• editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• extend Blueprintables • update specification levels • convert constraints in specification items • introduction of platform based document structure • introduction of Profiles for Data Exchange Points

2015-07-31	4.2.2	AUTOSAR Release Management	• introduction of LifeCycleState for constraint and specification items • editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• introduction of Blueprint Policy • include safety extension relevant items • extension of acceptance test items
2014-03-31	4.1.3	AUTOSAR Release Management	• editorial changes including tagged specification items • update content of specification levels
2013-10-31	4.1.2	AUTOSAR Release Management	• editorial changes including tagged specification items • extension of blueprinting to further AUTOSAR classes
2013-03-15	4.1.1	AUTOSAR Administration	• editorial changes including tagged specification items • extension of blueprinting to further AUTOSAR classes (e.g. build action manifest) • introduction of life cycle support • improvement of document traceability • refinement of traceability support
2011-12-22	4.0.3	AUTOSAR Administration	Initial Release

Table of Contents

References

[1] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[2] Requirements on Standardization Template
AUTOSAR_RS_StandardizationTemplate

[3] Predefined Names in AUTOSAR
AUTOSAR_TR_PredefinedNames

[4] Specification of Safety Extensions
AUTOSAR_TPS_SafetyExtensions

[5] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList

[6] Key words for use in RFCs to Indicate Requirement Levels
http://www.ietf.org/rfc/rfc2119.txt

[7] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

[8] XML Path language (XPath)
http://www.w3.org/TR/xpath/

[9] ANTLR parser generator V3

[10] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration

[11] Unique Names for Documentation, Measurement and Calibration: Modeling and Naming Aspects including Automatic Generation
AUTOSAR_TR_AIMeasurementCalibrationDiagnostics

[12] XML Specification of Application Interfaces
AUTOSAR_MOD_AISpecification

[13] Specification of Timing Extensions
AUTOSAR_TPS_TimingExtensions

[14] Explanation of Application Interfaces of the Powertrain Engine Domain
AUTOSAR_EXP_AIPowertrain

[15] SW-C and System Modeling Guide
AUTOSAR_TR_SWCModelingGuide

[16] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes

[17] Specification of CRC Routines
AUTOSAR_SWS_CRCLibrary

[18] Methodology
AUTOSAR_TR_Methodology

[19] Meta Model
AUTOSAR_MMOD_MetaModel

[20] Collection of constraints on AUTOSAR M1 models
AUTOSAR_TR_AutosarModelConstraints

[21] XML Schema Production Rules
AUTOSAR_TPS_XMLSchemaProductionRules

[22] Meta Model-generated XML Schema
AUTOSAR_MMOD_XMLSchema

[23] Software Process Engineering Meta-Model Specification
http://www.omg.org/spec/SPEM/2.0/

1 Introduction

AUTOSAR models are in many cases not created from scratch but existing content is taken as the basis. The existing content could be contributed by the AUTOSAR initiative itself in form of standardized model elements.

This document specifies the Standardization Template. This template is intended to support the delivery of standardized model elements by AUTOSAR and others.

AUTOSAR 4.0 already specifies the blueprint approach for standardization. This approach is continued and refined by the Standardization Template. It thereby replaces Appendix A in Software Component Template ([1]).

As an particular example, let us consider the standardization of application interfaces. That is, in terms of the AUTOSAR meta-model the standardization mainly applies to the definition of `PortPrototype`s for specific purposes.

Due to the structure of the AUTOSAR meta-model it is not possible to merely express a standardized `PortPrototype` because for good reasons the latter does not exist on its own but is always owned by a `SwComponentType`.

The Standardization Template specifies the approach to overcome this situation.

For more details such as use cases please refer to [2].

1.1 Document Conventions

Technical terms are typeset in mono spaced font, e.g. `PortPrototype`. As a general rule, plural forms of technical terms are created by adding "s" to the singular form, e.g. `PortPrototype`s. By this means the document resembles terminology used in the AUTOSAR XML Schema.

This document contains constraints in textual form that are distinguished from the rest of the text by a unique numerical constraint ID, a headline, and the actual constraint text starting after the ⌈ character and terminated by the ⌋ character.

The purpose of these constraints is to literally constrain the interpretation of the AUTOSAR meta-model such that it is possible to detect violations of the standardized behavior implemented in an instance of the meta-model (i.e. on M1 level).

Makers of AUTOSAR tools are encouraged to add the numerical ID of a constraint that corresponds to an M1 modeling issue as part of the diagnostic message issued by the tool.

The attributes of the classes introduced in this document are listed in form of class tables. They have the form shown in the example of the top-level element AUTOSAR:

Please note that constraints are not supposed to be enforceable at any given time in an AUTOSAR workflow. During the development of a model, constraints may legitimately be violated because an incomplete model will obviously show inconsistencies.

However, at specific points in the workflow, constraints shall be enforced as a safeguard against misconfiguration.

The points in the workflow where constraints shall be enforced, sometimes also known as the "binding time" of the constraint, are different for each model category, e.g. on the classic platform, the constraints defined for software-components are typically enforced prior to the generation of the RTE while the constraints against the definition of an Ecu extract shall be applied when the Ecu configuration for the Com stack is created.

For each document, possible binding times of constraints are defined and the binding times are typically mentioned in the constraint themselves to give a proper orientation for implementers of AUTOSAR authoring tools.

Class	**AUTOSAR**			
Package	M2::AUTOSARTemplates::AutosarTopLevelStructure			
Note	Root element of an AUTOSAR description, also the root element in corresponding XML documents. **Tags:**xml.globalElement=true			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data of an Autosar file. **Tags:**xml.sequenceOffset=10

arPackage	ARPackage	*	aggr	This is the top level package in an AUTOSAR model.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=arPackage.shortName, arPackage.variationPoint.shortLabel
				vh.latestBindingTime=blueprintDerivationTime
				xml.sequenceOffset=30

fileInfoComment	FileInfoComment	0..1	aggr	This represents a possibility to provide a structured comment in an AUTOSAR file.
				Stereotypes: atpStructuredComment
				Tags:
				xml.roleElement=true
				xml.sequenceOffset=-10
				xml.typeElement=false

introduction	DocumentationBlock	0..1	aggr	This represents an introduction on the Autosar file. It is intended for example to rpresent disclaimers and legal notes. **Tags:**xml.sequenceOffset=20

Table 1.1: AUTOSAR

The first rows in the table have the following meaning:

Class: The name of the class as defined in the UML model.

Package: The UML package the class is defined in. This is only listed to help locating the class in the overall meta model.

Note: The comment the modeler gave for the class (class note). Stereotypes and UML tags of the class are also denoted here.

Base Classes: If applicable, the list of direct base classes.

The headers in the table have the following meaning:

Attribute: The name of an attribute of the class. Note that AUTOSAR does not distinguish between class attributes and owned association ends.

Type: The type of an attribute of the class.

Mul.: The assigned multiplicity of the attribute, i.e. how many instances of the given data type are associated with the attribute.

Kind: Specifies, whether the attribute is aggregated in the class (`aggr` aggregation), an UML attribute in the class (`attr` primitive attribute), or just referenced by it (`ref` reference). Instance references are also indicated (`iref` instance reference) in this field.

Note: The comment the modeler gave for the class attribute (role note). Stereotypes and UML tags of the class are also denoted here.

Please note that the chapters that start with a letter instead of a numerical value represent the appendix of the document. The purpose of the appendix is to support the explanation of certain aspects of the document and does not represent binding conventions of the standard.

1.2 Requirements Tracing

The following table references the requirements specified in [2] and links to the fulfillments of these.

Requirement	Description	Satisfied by
[RS_STDT_00001]	Shall support and explain Blueprints in general	[TPS_STDT_00002] [TPS_STDT_00027] [TPS_STDT_00042] [TPS_STDT_00065] [TPS_STDT_00067]
[RS_STDT_00002]	Formalized description of BSW SWS	[TPS_STDT_00014] [TPS_STDT_00040] [TPS_STDT_00041] [TPS_STDT_00049] [TPS_STDT_00067] [TPS_STDT_00090] [TPS_STDT_00091]
[RS_STDT_00003]	Shall allow to represent port blueprints	[TPS_STDT_00007] [TPS_STDT_00047] [TPS_STDT_00061] [TPS_STDT_00082]
[RS_STDT_00004]	Shall allow to represent `shortName` patterns	[TPS_STDT_00003] [TPS_STDT_00047] [TPS_STDT_00055]
[RS_STDT_00005]	Shall support keywords and keyword abbreviations	[TPS_STDT_00004] [TPS_STDT_00012] [TPS_STDT_00068] [TPS_STDT_00069] [TPS_STDT_00070]
[RS_STDT_00006]	Shall be implemented without compatibility problems to existing template	[TPS_STDT_00033] [TPS_STDT_00041] [TPS_STDT_00047]
[RS_STDT_00007]	Shall be based on the AUTOSAR XML schema	[TPS_STDT_00033] [TPS_STDT_00041] [TPS_STDT_00047]
[RS_STDT_00008]	Shall provide means to support analyzing the conformity of implementations with the AUTOSAR standards	[TPS_STDT_00001] [TPS_STDT_00003] [TPS_STDT_00012] [TPS_STDT_00042] [TPS_STDT_00048] [TPS_STDT_00052] [TPS_STDT_00054] [TPS_STDT_00059] [TPS_STDT_00060]
[RS_STDT_00009]	Shall be able to represent requirements stated in SWS	[TPS_STDT_00001] [TPS_STDT_00042] [TPS_STDT_00050] [TPS_STDT_00052] [TPS_STDT_00060]
[RS_STDT_00010]	Shall refer to ECUC parameter definition	[TPS_STDT_00025] [TPS_STDT_00040]
[RS_STDT_00011]	Shall be able to standardize components	[TPS_STDT_00024]
[RS_STDT_00012]	Shall be able to standardize architecture	[TPS_STDT_00024]
[RS_STDT_00013]	Shall be able to express parts of reference paths resp. package hierarchies	[TPS_STDT_00013] [TPS_STDT_00051]
[RS_STDT_00014]	Shall be able to express levels of obligation	[TPS_STDT_00028] [TPS_STDT_00053] [TPS_STDT_00067]
[RS_STDT_00015]	Shall support different Approaches to derive from Blueprints	[TPS_STDT_00028]
[RS_STDT_00016]	Shall be able to express information about the state of model elements	[TPS_STDT_00038]
[RS_STDT_00017]	Shall cover the compatibility of blueprints and derived objects	[TPS_STDT_00005] [TPS_STDT_00008] [TPS_STDT_00051] [TPS_STDT_00072] [TPS_STDT_00085] [TPS_STDT_00086] [TPS_STDT_00087]
[RS_STDT_00018]	Shall allow to describe the dependencies of APIs (e.g. invocation and callback/polling interfaces)	[TPS_STDT_00014] [TPS_STDT_00048] [TPS_STDT_00090] [TPS_STDT_00091]

▽

△

Requirement	Description	Satisfied by
[RS_STDT_00019]	Shall define the mandatory semantics for a Blueprint	[TPS_STDT_00003] [TPS_STDT_00006] [TPS_STDT_00010] [TPS_STDT_00021] [TPS_STDT_00028] [TPS_STDT_00048]
[RS_STDT_00020]	Shall support variants of a Variable Dataprototype	[TPS_STDT_00028] [TPS_STDT_00030] [TPS_STDT_00044] [TPS_STDT_00045] [TPS_STDT_00046]
[RS_STDT_00021]	Shall support multiple instantiation for an example SWC with Port Blueprint	[TPS_STDT_00003] [TPS_STDT_00036] [TPS_STDT_00037]
[RS_STDT_00022]	Means of exchange format between stakeholders for blueprints	[TPS_STDT_00025]
[RS_STDT_00023]	Shall be able to standardize Alias Names	[TPS_STDT_00011]
[RS_STDT_00024]	Shall be able to standardize Unique Names and Display Names	[TPS_STDT_00031]
[RS_STDT_00025]	Shall be able to standardize life cycle states	[TPS_STDT_00043] [TPS_STDT_00064]
[RS_STDT_00026]	Shall allow to represent port interface blueprints	[TPS_STDT_00009] [TPS_STDT_00066]
[RS_STDT_00027]	Shall allow to evaluate the integrity of Blueprints	[TPS_STDT_00034]
[RS_STDT_00028]	Shall allow to generate BSW "Standard AUTOSAR Interface" description from model	[TPS_STDT_00023] [TPS_STDT_00067]
[RS_STDT_00029]	Shall be able to represent further Blueprints	[TPS_STDT_00014] [TPS_STDT_00015] [TPS_STDT_00016] [TPS_STDT_00017] [TPS_STDT_00018] [TPS_STDT_00019] [TPS_STDT_00020] [TPS_STDT_00022] [TPS_STDT_00023] [TPS_STDT_00026] [TPS_STDT_00035] [TPS_STDT_00049] [TPS_STDT_00079] [TPS_STDT_00083] [TPS_STDT_00084] [TPS_STDT_00090]
[RS_STDT_00030]	Shall allow to standardize package structures	[TPS_STDT_00013] [TPS_STDT_00067]
[RS_STDT_00031]	Shall support general specification items	[TPS_STDT_00042] [TPS_STDT_00056] [TPS_STDT_00057] [TPS_STDT_00058] [TPS_STDT_00089]
[RS_STDT_00032]	Shall be able to provide Blueprints for Roles and Rights	[TPS_STDT_00062]
[RS_STDT_00033]	Shall be able to provide Blueprints for Build Action Manifest	[TPS_STDT_00063] [TPS_STDT_00065]
[RS_STDT_00034]	Blueprinting of Implicit Communication Behavior	[TPS_STDT_00071] [TPS_STDT_00073] [TPS_STDT_00074] [TPS_STDT_00075] [TPS_STDT_00076]
[RS_STDT_00035]	Shall support blueprinting of keywords	[TPS_STDT_00077]
[RS_STDT_00036]	StandardizationTemplate shall specify the representation of requirements in AUTOSAR documents	[TPS_STDT_00078]
[RS_STDT_00037]	StandardizationTemplate shall specify the representation of specification items in AUTOSAR documents	[TPS_STDT_00080]

▽

△

Requirement	Description	Satisfied by
[RS_STDT_00038]	StandardizationTemplate shall specify the representation of constraint items in AUTOSAR documents	[TPS_STDT_00081] [TPS_STDT_00088]
[RS_STDT_00039]	StandardizationTemplate shall specify the representation of test items in AUTOSAR documents	[TPS_STDT_00029]
[RS_STDT_00040]	Multiplicity of elements in derived objects	[TPS_STDT_00032] [TPS_STDT_00039]
[RS_STDT_00042]	Shall provide the ability to define naming conventions for public symbols	[TPS_STDT_00004] [TPS_STDT_00012] [TPS_STDT_00068] [TPS_STDT_00069] [TPS_STDT_00070]
[RS_STDT_00101]	Description of Data Exchange Point Shall Provide a Human Readable High-Level Overview	[TPS_STDT_00120] [TPS_STDT_00121]
[RS_STDT_00102]	Description of Data Exchange Point Shall Describe Work Product in Methodology	[TPS_STDT_00100] [TPS_STDT_00102] [TPS_STDT_00103] [TPS_STDT_00104] [TPS_STDT_00123] [TPS_STDT_00156] [TPS_STDT_00187] [TPS_STDT_00188] [TPS_STDT_00192] [TPS_STDT_00193]
[RS_STDT_00103]	Description of Data Exchange Point Shall Describe Intended Use	[TPS_STDT_00100] [TPS_STDT_00102] [TPS_STDT_00103] [TPS_STDT_00104] [TPS_STDT_00123] [TPS_STDT_00124] [TPS_STDT_00156] [TPS_STDT_00187] [TPS_STDT_00188] [TPS_STDT_00192] [TPS_STDT_00193]
[RS_STDT_00104]	Description of Data Exchange Point Shall Describe Tool and Organization	[TPS_STDT_00121]
[RS_STDT_00105]	Description of Data Exchange Point Shall Describe AUTOSAR Revision	[TPS_STDT_00122] [TPS_STDT_00191] [TPS_STDT_00211]
[RS_STDT_00106]	Description of Data Exchange Point Shall Describe Relevant or Excluded Subset of the AUTOSAR Meta-Model	[TPS_STDT_00102] [TPS_STDT_00103] [TPS_STDT_00104] [TPS_STDT_00107] [TPS_STDT_00108] [TPS_STDT_00109] [TPS_STDT_00112] [TPS_STDT_00113] [TPS_STDT_00114] [TPS_STDT_00119] [TPS_STDT_00124] [TPS_STDT_00126] [TPS_STDT_00129] [TPS_STDT_00138] [TPS_STDT_00139] [TPS_STDT_00140] [TPS_STDT_00141] [TPS_STDT_00142] [TPS_STDT_00143] [TPS_STDT_00144] [TPS_STDT_00145] [TPS_STDT_00146] [TPS_STDT_00157] [TPS_STDT_00159] [TPS_STDT_00163] [TPS_STDT_00174] [TPS_STDT_00177] [TPS_STDT_00178] [TPS_STDT_00179] [TPS_STDT_00180] [TPS_STDT_00181] [TPS_STDT_00182] [TPS_STDT_00186] [TPS_STDT_00190] [TPS_STDT_00191] [TPS_STDT_00195] [TPS_STDT_00196] [TPS_STDT_00197] [TPS_STDT_00198] [TPS_STDT_00199] [TPS_STDT_00200]
[RS_STDT_00107]	Description of Data Exchange Point Shall Describe Relevant or Excluded Subset of Model	[TPS_STDT_00130] [TPS_STDT_00157]

▽

△

Requirement	Description	Satisfied by
[RS_STDT_00108]	Description of Data Exchange Point Shall Describe Relevant Constraints	[TPS_STDT_00102] [TPS_STDT_00103] [TPS_STDT_00104] [TPS_STDT_00111] [TPS_STDT_00124] [TPS_STDT_00125] [TPS_STDT_00147] [TPS_STDT_00157] [TPS_STDT_00164] [TPS_STDT_00165]
[RS_STDT_00109]	Description of Data Exchange Point Shall Describe Relevant Spec Items	[TPS_STDT_00102] [TPS_STDT_00103] [TPS_STDT_00104] [TPS_STDT_00124] [TPS_STDT_00157]
[RS_STDT_00110]	Description of Data Exchange Point Shall Describe Model Completeness	[TPS_STDT_00157] [TPS_STDT_00174]
[RS_STDT_00111]	Description of Data Exchange Point Shall Describe Applicability of Default Values	[TPS_STDT_00127] [TPS_STDT_00157] [TPS_STDT_00204] [TPS_STDT_00207]
[RS_STDT_00113]	Description of Data Exchange Point Shall Describe Limitation of Values of Primitive Attributes	[TPS_STDT_00157] [TPS_STDT_00173] [TPS_STDT_00203]
[RS_STDT_00114]	Description of Data Exchange Point Shall Support Severity Levels for Compliance with Individual Rules of the Profile	[TPS_STDT_00126] [TPS_STDT_00157] [TPS_STDT_00172] [TPS_STDT_00186]
[RS_STDT_00115]	Description of Data Exchange Point Shall Describe Rationales of Decisions	[TPS_STDT_00168] [TPS_STDT_00170]
[RS_STDT_00116]	Description of Data Exchange Point Shall Describe Usage of AUTOSAR Extension Mechanisms	[TPS_STDT_00132] [TPS_STDT_00157]
[RS_STDT_00117]	AUTOSAR Shall Provide Guidelines for Comparison of Profiles for Data Exchange Points	[TPS_STDT_00115] [TPS_STDT_00116]
[RS_STDT_00118]	AUTOSAR Shall Provide Guidelines for Compatibility of Profiles for Data Exchange Points	[TPS_STDT_00101] [TPS_STDT_00110] [TPS_STDT_00115] [TPS_STDT_00116] [TPS_STDT_00128] [TPS_STDT_00131] [TPS_STDT_00133] [TPS_STDT_00134] [TPS_STDT_00135] [TPS_STDT_00136] [TPS_STDT_00160] [TPS_STDT_00183] [TPS_STDT_00201] [TPS_STDT_00202] [TPS_STDT_00205] [TPS_STDT_00206] [TPS_STDT_00208] [TPS_STDT_00209] [TPS_STDT_00210]
[RS_STDT_00120]	AUTOSAR Shall Provide Support for Handling of Incomplete Profiles for Data Exchange Points	[TPS_STDT_00105] [TPS_STDT_00106]
[RS_STDT_00121]	AUTOSAR Shall Provide Guidance for Checking Compliance of AUTOSAR Model Against Profiles for Data Exchange Points	[TPS_STDT_00117] [TPS_STDT_00118] [TPS_STDT_00125] [TPS_STDT_00129] [TPS_STDT_00159] [TPS_STDT_00163] [TPS_STDT_00164] [TPS_STDT_00165] [TPS_STDT_00167] [TPS_STDT_00169]
[RS_STDT_00122]	AUTOSAR Shall Provide Guidance for Identification of Not Yet Described Aspects within Profiles for Data Exchange Points	[TPS_STDT_00111]
[RS_STDT_00125]	Support of AUTOSAR Specific Modeling Patterns	[TPS_STDT_00175] [TPS_STDT_00176]

Table 1.2: RequirementsTracing

2 Support for Traceability

AUTOSAR has defined four levels of requirements for its standardization work:

1. AUTOSAR Project Objectives
2. AUTOSAR Main Requirements
3. AUTOSAR Requirements Specifications (RS, SRS, ATR)
4. AUTOSAR Specifications (SWS, TPS, AI, TR, MOD, ATS, EXP etc.)

The used abbreviations are defined in [3].

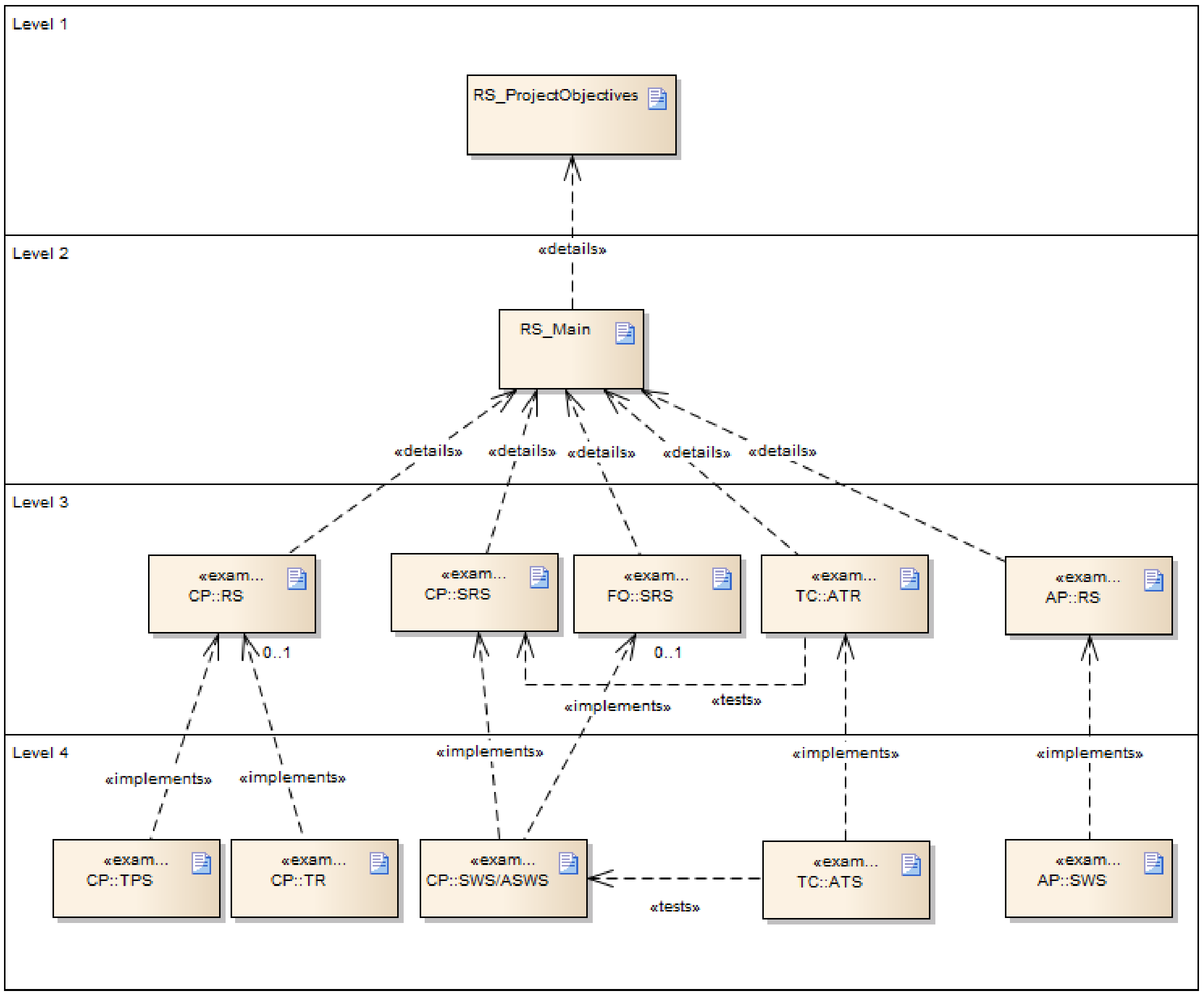


Figure 2.1: Specification levels

The assignment of platform based documents is realized by the "applies to" relation as illustrated in figure 2.2.

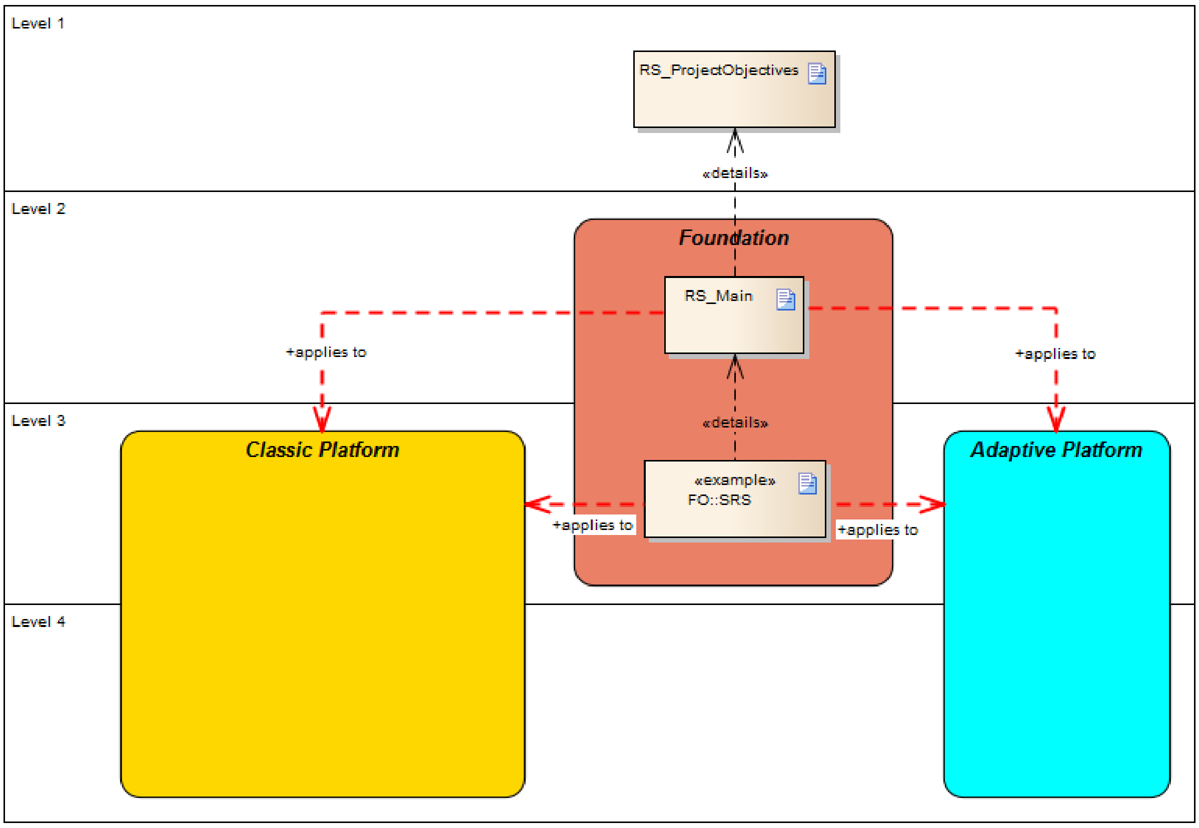


Figure 2.2: Platform based document structure

[TPS_STDT_00001] Support bottom up tracing ⌈Standardization Template supports bottom up tracing between these levels by the meta-class `Traceable`. This allows to represent traceable entities and to establish traces between those. These entities reside within a `DocumentationBlock`. One prominent place is `DocumentationBlock.trace` in particular within `Identifiable.introduction`.⌋*(RS_STDT_00008, RS_STDT_00009)*

[constr_2625] Allowed uptraces wrt. life cycles ⌈Table 2.1 defines the allowed combinations of uptraces with respect to life cycle states [TPS_STDT_00064].⌋*()*

	Trace to					
Trace from	draft	valid	obsolete	preliminary	removed	shallBecome Mandatory
draft	1	1	0	1	0	1
valid	0	1	0	0	0	0
obsolete	1	1	1	1	0	1
preliminary	1	1	0	1	0	1
removed	1	1	1	1	1	1
shallBecomeMandatory	0	1	0	0	0	1

Table 2.1: Matrix of allowed uptraces wrt. life cycles

There "'1"' means allowed and "'0"' means not allowed combination. The above [constr_2625] shall ensure that the tracing element does not have a life cycle state that is in contradiction to the life cycle state of the traced element. E.g. a specification items is tracing a requirement.

[TPS_STDT_00080] Representation of specification items in AUTOSAR documents ⌈AUTOSAR specification items are represented using the structure with the following attributes:

- The headline consists of an Id (short name) which shall be written inside squared brackets and shall follow [TPS_STDT_00042].
- After the Id the `LifeCycleState` follows in curly brackets. The allowed values are VALID, DRAFT and OBSOLETE and shall follow [TPS_GST_00051]. If there is no `LifeCycleState` information stated then the state is VALID.
- After the `LifeCycleState` an optional specification item title (long name) should be stated to improve human readability.
- The next line starts with an opening half bracket and the content of the specification item follows. The end of it shall be marked by the closing half bracket.
- After the closing half bracket an opening round bracket indicates the comma separated list of requirements which are fulfilled by this specification item. The end of it shall be marked by the closing round bracket. If no up traces are available the round brackets shall be written with empty content.
- The specification items shall describe the semantics and syntax of models.

⌋*(RS_STDT_00037)*

[TPS_STDT_00081] Representation of constraint items in AUTOSAR template documents ⌈AUTOSAR constraint items in template documents are represented using the structure with the following attributes:

- The Id (short name) of the constraint is composed by "constr_" and a four digit number as identifier. Both shall be written in squared brackets. The four digit number (identifier) shall be harmonized globally and committed.
- After the Id the `LifeCycleState` follows in curly brackets. The allowed values are VALID, DRAFT and OBSOLETE and shall follow [TPS_GST_00051]. If there is no `LifeCycleState` information stated then the state is VALID.
- After the `LifeCycleState` the constraint title (long name) follows.
- The constraint content shall be written inside the opening and closing half bracket.
- The constraint items shall further restrict the validity of models.

⌋*(RS_STDT_00038)*

[TPS_STDT_00088] Representation of constraint items in AUTOSAR non template documents ⌈AUTOSAR constraint items in AUTOSAR non template documents are represented using the structure with the following attributes:

- The headline consists of an Id (short name) which shall be written inside squared brackets and shall follow [TPS_STDT_00042].
- After the Id the `LifeCycleState` follows in curly brackets. The allowed values are VALID, DRAFT and OBSOLETE and shall follow [TPS_GST_00051]. If there is no `LifeCycleState` information stated then the state is VALID.
- After the `LifeCycleState` the constraint title (long name) follows.
- The constraint content shall be written inside the opening and closing half bracket.

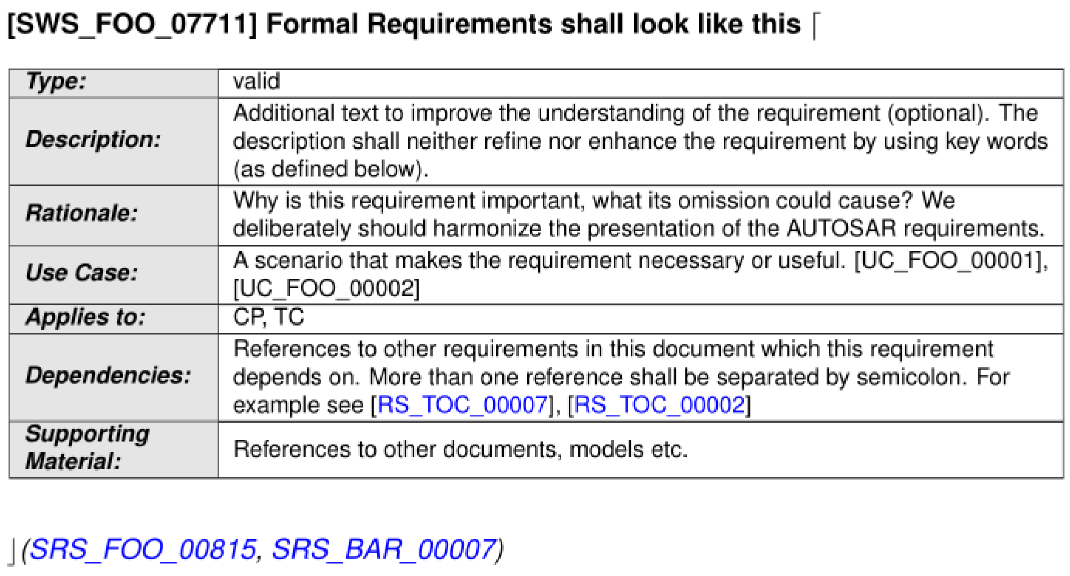
⌋*(RS_STDT_00038)*

[TPS_STDT_00078] Representation of requirements in AUTOSAR documents ⌈AUTOSAR requirements are represented using the structure of [TPS_STDT_00060] where the following attributes are presented as a table:

- Id (short name) and requirement (long name) are shown in the headline.
- The requirement (long name) shall be a complete English sentence using one of the keywords from [TPS_STDT_00053]. That means a mandatory requirement follows the written form: "<who> shall do <what>".
- "implements" represents the uptrace at the end of the table
- "applies to" shall contain a comma separated tag list with one or more of the following values "CP", "AP", "FO", "TC", "TA"
- Type, Description, Rationale, Applies To, Use Case, Dependencies and Supporting Material are shown as table rows.
- The value of Type shall be one of "valid", "draft" or "obsolete", see [TPS_STDT_00064].

⌋*(RS_STDT_00036)*

The rendition is illustrated in figure 2.3.



[SWS_FOO_07711] Formal Requirements shall look like this ⌈

Type:	valid
Description:	Additional text to improve the understanding of the requirement (optional). The description shall neither refine nor enhance the requirement by using key words (as defined below).
Rationale:	Why is this requirement important, what its omission could cause? We deliberately should harmonize the presentation of the AUTOSAR requirements.
Use Case:	A scenario that makes the requirement necessary or useful. [UC_FOO_00001], [UC_FOO_00002]
Applies to:	CP, TC
Dependencies:	References to other requirements in this document which this requirement depends on. More than one reference shall be separated by semicolon. For example see [RS_TOC_00007], [RS_TOC_00002]
Supporting Material:	References to other documents, models etc.

⌋*(SRS_FOO_00815, SRS_BAR_00007)*

Figure 2.3: Requirements Table

[constr_2603] Use of "applies to" in context of the specification level ⌈On specification level 1 and 2 only the requirements table including the appliesTo attribute shall be used. On the specification levels 3 and 4 only the requirements table without the appliesTo attribute shall be used. Exception: Documents of the foundation which are handled on specification level 3.⌋*()*

Rational: This avoids unintentional cross references which disturb the structure of tracing.

[constr_2604] Allowed uptraces in context of "applies to" values ⌈Traces to documents of upper specification levels shall be conform to the values assigned to appliesTo.⌋*()*

Note: Patterns of not allowed uptraces are marked with "NOT ALLOWED" in figure 2.4.

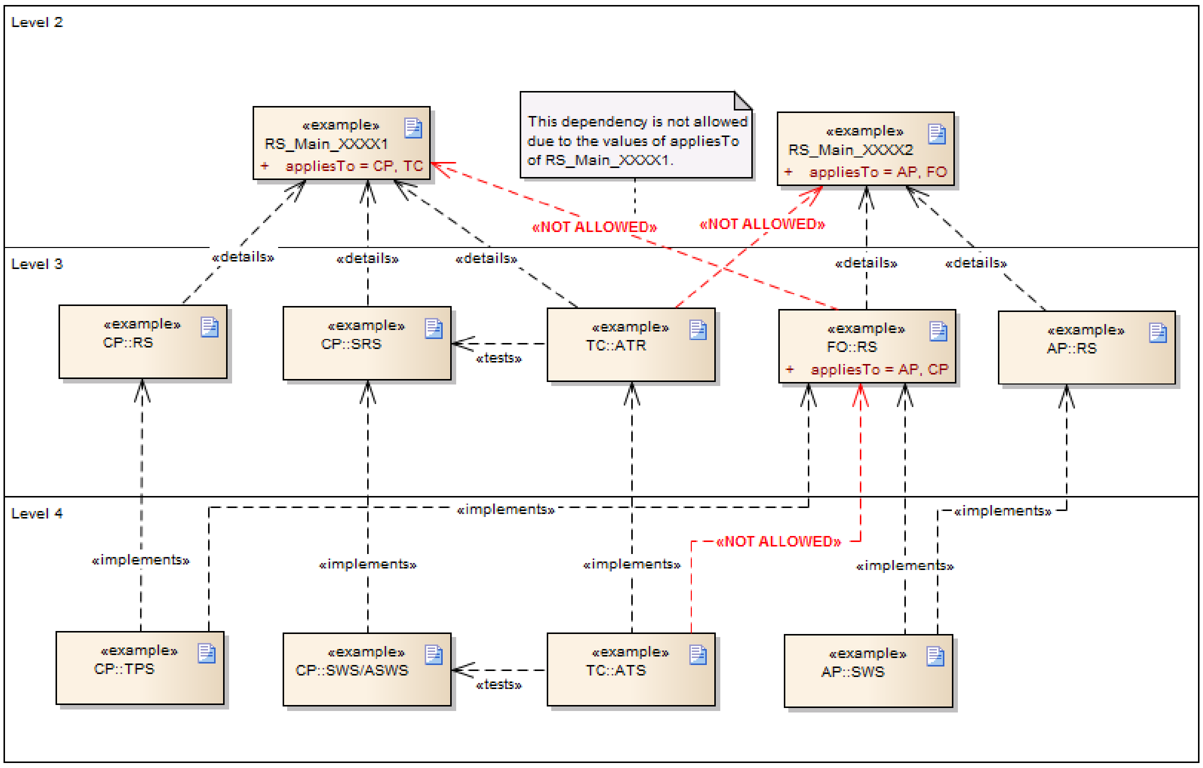


Figure 2.4: Use of appliesTo

Note: Optional requirements on level 1 to 4 of the AUTOSAR requirements hierarchy are not allowed. An optional part of an implementation is only optional for the end-user of AUTOSAR. In order to provide this option, the corresponding choice shall be mandatory in the according specification. That means, a feature described as "AUTOSAR should support foobar" can never be correct, because the underlying requirements layer is always static and would have no chance to decide whether "foobar" should be part of it or not. A correct writing would be e. g. "AUTOSAR shall support optional foobar".

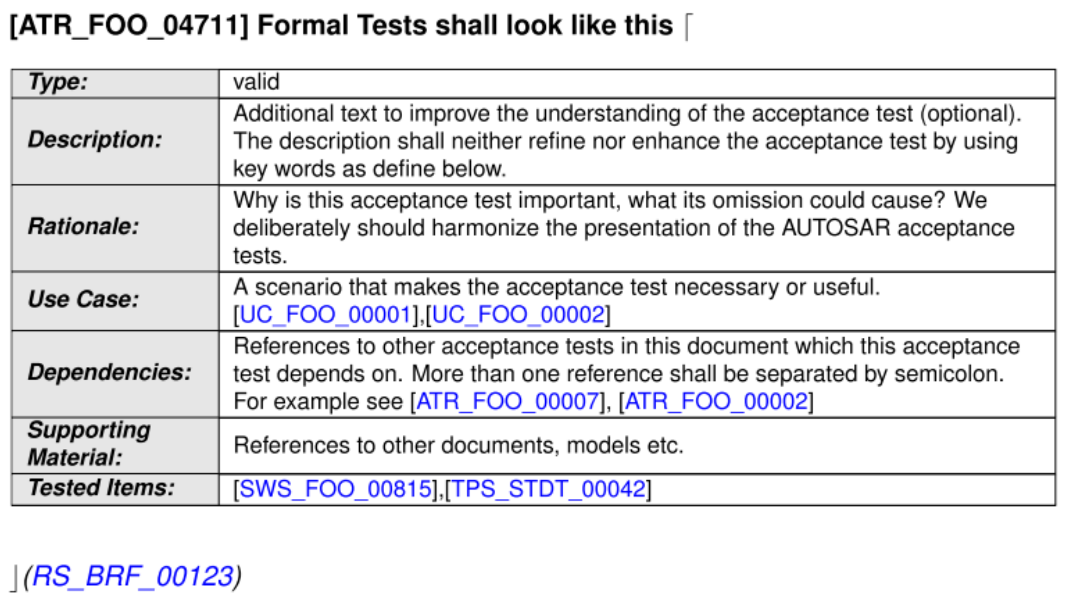
[TPS_STDT_00029] Representation of test items in AUTOSAR documents ⌈AUTOSAR test items are represented using the structure of [TPS_STDT_00060] where the following attributes are presented as a table:

- Id (short name) and test item (long name) are shown in the headline.
- The test item (long name) shall be a complete English sentence.
- "implements" represents the uptrace at the end of the table
- Type, Description, Rationale, Use Case, Dependencies, Supporting Material and Tested Items are shown as table rows.
- The value of Type shall be one of "valid", "draft" or "obsolete", see [TPS_STDT_00064].

⌋*(RS_STDT_00039)*

The representation of test items [TPS_STDT_00029] can also be used in level 4 as in level 3, see figure 2.1.

The rendition is illustrated in figure 2.5.



[ATR_FOO_04711] Formal Tests shall look like this ⌈

Type:	valid
Description:	Additional text to improve the understanding of the acceptance test (optional). The description shall neither refine nor enhance the acceptance test by using key words as define below.
Rationale:	Why is this acceptance test important, what its omission could cause? We deliberately should harmonize the presentation of the AUTOSAR acceptance tests.
Use Case:	A scenario that makes the acceptance test necessary or useful. [UC_FOO_00001],[UC_FOO_00002]
Dependencies:	References to other acceptance tests in this document which this acceptance test depends on. More than one reference shall be separated by semicolon. For example see [ATR_FOO_00007], [ATR_FOO_00002]
Supporting Material:	References to other documents, models etc.
Tested Items:	[SWS_FOO_00815],[TPS_STDT_00042]

⌋*(RS_BRF_00123)*

Figure 2.5: Test Item Table

Note: The unicodes of the half brackets are for opening half bracket: 0x2308 and for closing half bracket: 0x230B.

`Traceable` is specialized in

- **[TPS_STDT_00059]** **`TraceableText`** ⌈This represents a paragraph level text which can be referenced in order to establish requirements tracing. It is an abstract class from which particular specializations support specific kinds of tracing such as requirements / constraints.⌋*(RS_STDT_00008)*

  [constr_2540] Tagged text category ⌈The `category` of `TraceableText` shall be one of

  `ADVISORY_ITEM` The text represents a particular advisory. Such an item is applicable primarily in template specifications. It is similar to a constraint item but represents the characteristic of a WARNING rather than an ERROR.

  `CONSTRAINT_ITEM` The text represents a particular constraint. Such an item is applicable primarily in template specifications. It is similar to a specification item but represents issues that may be validated automatically e.g. by a tool.

  `IMPLEMENTATION_ITEM` The text represents a short description of an implementation. It is applicable primarily within the `introduction` of a model element.

  `REQUIREMENT_ITEM` The text represents a particular requirement. Such an item is applicable primarily in requirement specifications.

SAFETY_* The text represents the type of safety requirements. The allowed values (*) are defined in [TPS_SAFEX_00102] in [4].

SPECIFICATION_ITEM The text represents a particular item in the specification. Such an item is a requirement for the implementation of the software specification.

SRC The text represents the source code content.

TEST_ITEM The text represents a short description of a test. Such an item is applicable primarily in test specifications.

⌋*()*

- **[TPS_STDT_00060]** **StructuredReq** ⌈This represents a structured requirement as it is used within AUTOSAR RS documents.⌋*(RS_STDT_00008, RS_-STDT_00009)*

Note that as TraceableText is aggregated in DocumentationBlock it also requires a proper rendition in printed documents. For an example of a proper rendition see [TPS_STDT_00001] above.

[constr_2565] Trace shall not be nested ⌈Due to the intended atomicity of requirements respectively specification items, Traceable shall not be nested.⌋*()*

[TPS_STDT_00042] namePattern for shortNames of TraceableText in Standardization Documents ⌈The intended name pattern applicable to short names TraceableText (in fact representing e.g. requirement tags) in AUTOSAR standardization documents is defined as

```
{keyword(TraceCategory)}_{module}_({special}[_{index}])|{index}
```

In this pattern, the placeholders are defined as:

- `keyword(TraceCategory)` is defined in [3] in keyword set `InformationCategories`, entries with classification `TraceCategory`.
- `module` is either module abbreviation in [5] or an entry of the keyword set `DocumentAbbreviations` with classification `DocumentAbbreviation` in [3]. Inside one document only the same module abbreviation or keyword shall be used.
- `index` is a numerical index
- `special` is one of (`SPEC`, `NA`, `GEN`, `CONSTR`). Note that `special` may also have an optional index. This allows to provide different special items with more detailed information.

⌋*(RS_STDT_00009, RS_STDT_00008, RS_STDT_00001, RS_STDT_00031)*

Note that some existing specifications historically contain multiple abbreviations inside the document and do therefore not follow this pattern. These are exceptions and shall not be applied to new documents.

[TPS_STDT_00056] Identifying not applicable requirements ⌈For those requirements which are not applicable to a particular specification, [TPS_STDT_00042] allows the `special` to be `NA`.

In order to apply this, specification item with the `shortName` e.g ([RS_STDT_NA] or even [RS_STDT_NA_00099]) may be created which traces back to the not applicable requirement items.

By this, not applicable requirements are easily identified in requirements tracing tables. Requirements tracing is complete since it also explicitly expresses the not applicable requirements.⌋*(RS_STDT_00031)*

[TPS_STDT_00057] Identifying generally fulfilled requirements ⌈For those requirements which are fulfilled by a generic concept, [TPS_STDT_00042] allows the `special` to be `GEN`.

In order to apply this, specification item with an appropriate `shortName` (e.g. [RS_STDT_GEN] or even [RS_STDT_GEN_00098]) may be created which traces back to the generally fulfilled requirement items.

By this, requirements considered to be fulfilled in general are easily identified in requirements tracing tables. Requirements tracing is complete since it also explicitly expresses the generally (or implicitly fulfilled) requirements.⌋*(RS_STDT_00031)*

[TPS_STDT_00058] Identifying requirements which need more specialization ⌈For those requirements which are fulfilled by items in a general specification together with items in individual specifications, [TPS_STDT_00042] allows the `special` to be `SPEC`.

In order to apply this, an item with an appropriate `shortName` (e.g. [RS_STDT_SPEC] or even [RS_STDT_SPEC_00092]) may be crated which traces back to the requirement items which need additional items in the individual specification.

By this,it is possible to identify the requirement items in the general specification, which need complementary items in an individual specification. This finally allows to perform a complete requirements tracing.⌋*(RS_STDT_00031)*

Figure 2.6 illustrates a requirements tracing table which utilizes the features provided by [TPS_STDT_00056] and [TPS_STDT_00058]:

SWS CanIf

Requirements traceability to SRS BSW General

Requirement	Description	Satisfied by
[RS_BSW_001]	Requirement title …	[SWS_BSW_0100]
[RS_BSW_002]	Requirement title …	[SWS_CANIF_0815]
		[SWS_CANIF_2000]
		[SWS_BSW_SPEC]
[RS_BSW_003]	Requirement title …	[SWS_BSW_0100]
		[SWS_BSW_0105]
[RS_BSW_004]	Requirement title …	[SWS_CANIF_0158]
		[SWS_BSW_0101]
[RS_BSW_005]	Requirement title …	[SWS_CANIF_NA]
		[SWS_BSW_0102]
		[SWS_BSW_SPEC]
[RS_BSW_006]	Requirement title …	[SWS_CANIF_NA]
[RS_BSW_007]	Requirement title …	[SWS_CANIF_0784]
		[SWS_BSW_0104]
		[SWS_BSW_SPEC]
[RS_BSW_008]	Requirement title …	[SWS_CANIF_NA]

…

Requirements traceability to SRS CAN

[RS_CANIF_001]	Requirement title …	[SWS_CANIF_0434]
[RS_CANIF_002]	Requirement title …	[SWS_CANIF_0435]
[RS_CANIF_003]	Requirement title …	[SWS_CANIF_0436]

…

Figure 2.6: Example for trace table using NA and SPEC

[TPS_STDT_00089] Identifying specification items which are constraints in AUTOSAR non template documents ⌈For those specification items which are constraints, [TPS_STDT_00042] allows the special to be `CONSTR`. In order to apply this, an item with an appropriate shortName (e.g. [SWS_Dem_CONSTR_06101]) may be created. For this case, the numerical `index` is mandatory.⌋*(RS_STDT_00031)*

[TPS_STDT_00052] Characteristics of `TraceableText` ⌈`TraceableText` should[1] be:

[1]This usage of the word "should" indicates that this is not always easy to decide. For example [TPS_STDT_00052] could also have been divided in one `TraceableText` per item.

- **identifiable**: `TraceableText` shall be identified by a unique short name (see [TPS_STDT_00042]). This is automatically fulfilled by applying the AUTOSAR meta model and schema.
- **specific**: `TraceableText` should be written such that the content is unambiguous and comprehensive - even if this would not result in an elegant writing style.
- **atomic**: One `TraceableText` should cover one particular issue.
- **verifiable**: The content of `TraceableText` should be written concrete such that it can be verified - not necessarily automatically but at least by human experts.

  In particular the requirement levels specified in [TPS_STDT_00053] shall be applied.

⌋*(RS_STDT_00008, RS_STDT_00009)*

[TPS_STDT_00053] Expression of obligation ⌈The following verbal forms for the expression of obligation shall be used to indicate requirements.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as follows, based on [6].

Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the adjective "LEGALLY REQUIRED", means that the definition is an absolute requirement of the specification due to legal issues.
- MUST NOT: This phrase, or the phrase "MUST NOT", means that the definition is an absolute prohibition of the specification due to legal issues.
- SHALL: This phrase, or the adjective "REQUIRED", means that the definition is an absolute requirement of the specification.
- SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular market-

place requires it or because the vendor feels that it enhances the product while another vendor may omit the same item.

An implementation, which does not include a particular option, SHALL be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, SHALL be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

⌋*(RS_STDT_00014)*

[TPS_STDT_00054] Organisation of `TraceableText` ⌈A set of `TraceableText` within a specification shall have the following properties:

- **hierarchical structure:** Multiple `TraceableText`s shall be structured in several successive levels - this is mostly ensured by the templates for the different kind of AUTOSAR specifications.
- **completeness:** `TraceableText` at one level shall fully implement all `TraceableText` of the previous level.
- **external consistency:** Multiple `TraceableText`s shall not contradict each other.
- **no duplication of information within any level of the hierarchical structure**: The content of one `TraceableText` shall not be repeated in any other `TraceableText` within the same level of the hierarchical structure.
- **maintainability:** A set of `TraceableText` can be modified or extended, e.g. by introduction of new versions of TraceableText or by adding/removing `TraceableText`. The `shortName` of `TraceableText` shall not be reused or changed.

⌋*(RS_STDT_00008)*

The levels mentioned in [TPS_STDT_00054] are illustrated in figure 2.1.

[TPS_STDT_00050] namePattern for AUTOSAR delivered Files ⌈The intended name pattern applied for filenames of AUTOSAR delivered files is defined as

```
AUTOSAR_{keyword(DocumentCategory)}_{DocumentName}
```

In this pattern, the placeholders are defined as:

- `keyword(DocumentCategory)` is defined in [3] in keyword set `InformationCategories`, entries with classification `DocumentCategory`.
- `DocumentName` is the `shortName` of the `Keyword` according to [3], keyword set `DocumentAbbreviation` entries with classification `DocumentAbbreviation` or the shortName of the module in [5]

⌋*(RS_STDT_00009)*

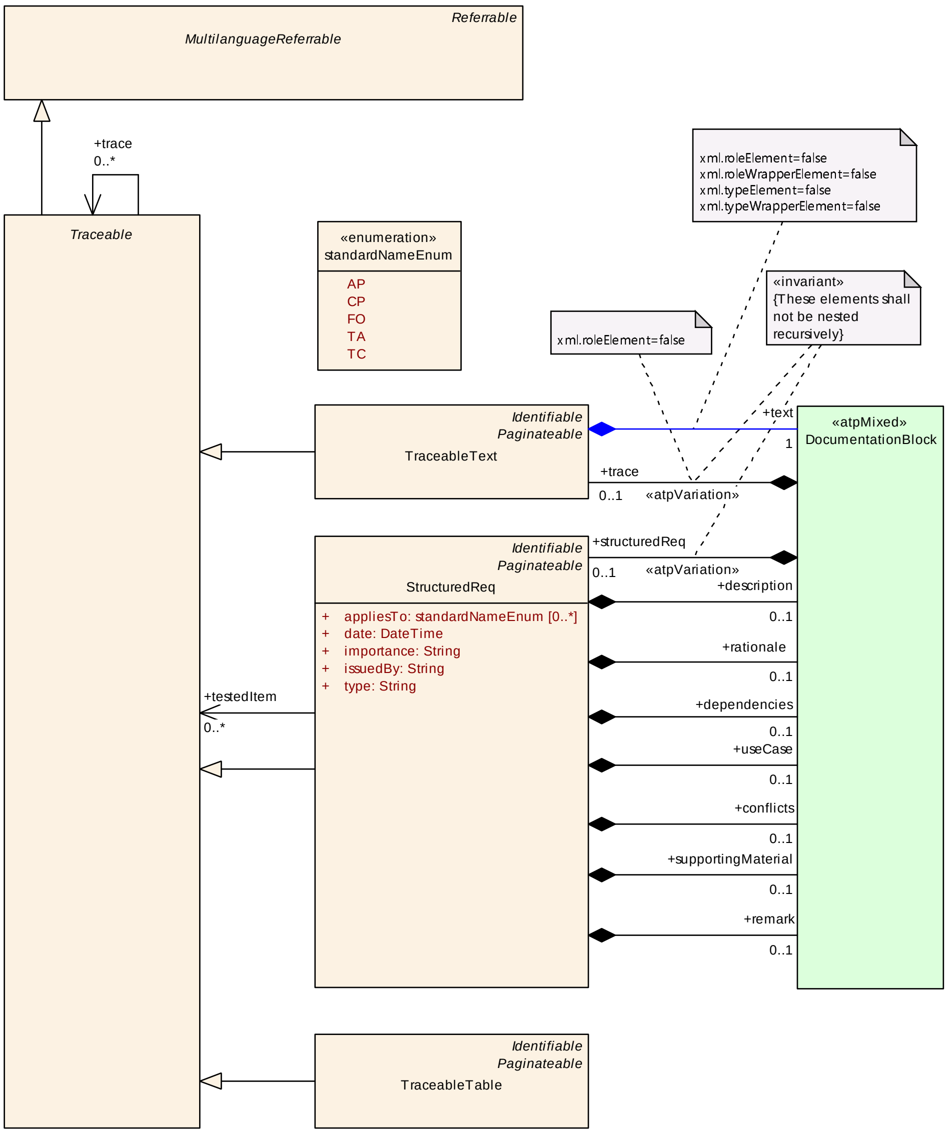


Figure 2.7: Requirements and Tracing

Class	***Traceable*** (abstract)			
Package	M2::MSR::Documentation::BlockElements::RequirementsTracing			
Note	This meta class represents the ability to be subject to tracing within an AUTOSAR model. Note that it is expected that its subclasses inherit either from MultilanguageReferrable or from Identifiable. Nevertheless it also inherits from MultilanguageReferrable in order to provide a common reference target for all Traceables.			
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*			
Subclasses	StructuredReq, *TimingConstraint*, TraceableTable, TraceableText			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	***Traceable*** (abstract)			
trace	Traceable	*	ref	This assocation represents the ability to trace to upstream requirements / constraints. This supports for example the bottom up tracing ProjectObjectives <- MainRequirements <- Features <- RequirementSpecs <- BSW/AI **Tags:**xml.sequenceOffset=20

Table 2.2: Traceable

Class	**TraceableText**			
Package	M2::MSR::Documentation::BlockElements::RequirementsTracing			
Note	This meta-class represents the ability to denote a traceable text item such as requirements etc. The following approach applies: • **shortName** represents the tag for tracing • **longName** represents the head line • **category** represents the kind of the tagged text			
Base	*ARObject*, *DocumentViewSelectable*, *Identifiable*, *MultilanguageReferrable*, *Paginateable*, *Referrable*, *Traceable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

text	DocumentationBlock	1	aggr	This represents the text to which the tag applies.
				Tags:
				xml.roleElement=false
				xml.roleWrapperElement=false
				xml.sequenceOffset=30
				xml.typeElement=false
				xml.typeWrapperElement=false

Table 2.3: TraceableText

Class	**StructuredReq**			
Package	M2::MSR::Documentation::BlockElements::RequirementsTracing			
Note	This represents a structured requirement. This is intended for a case where specific requirements for features are collected. Note that this can be rendered as a labeled list.			
Base	*ARObject*, *DocumentViewSelectable*, *Identifiable*, *MultilanguageReferrable*, *Paginateable*, *Referrable*, *Traceable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

appliesTo	standardNameEnum	*	attr	This attribute represents the platform the requirement is assigned to.
				Tags:
				xml.namePlural=APPLIES-TO-DEPENDENCIES
				xml.sequenceOffset=25

conflicts	DocumentationBlock	0..1	aggr	This represents an informal specification of conflicts. **Tags:**xml.sequenceOffset=40
date	DateTime	1	attr	This represents the date when the requirement was initiated. **Tags:**xml.sequenceOffset=5

▽

△

Class	**StructuredReq**			
dependencies	DocumentationBlock	0..1	aggr	This represents an informal specifiaction of dependencies. Note that upstream tracing should be formalized in the property trace provided by the superclass Traceable. **Tags:**xml.sequenceOffset=30
description	DocumentationBlock	0..1	aggr	Ths represents the general description of the requirement. **Tags:**xml.sequenceOffset=10
importance	String	1	attr	This allows to represent the importance of the requirement. **Tags:**xml.sequenceOffset=8
issuedBy	String	1	attr	This represents the person, organization or authority which issued the requirement. **Tags:**xml.sequenceOffset=6
rationale	DocumentationBlock	0..1	aggr	This represents the rationale of the requirement. **Tags:**xml.sequenceOffset=20
remark	DocumentationBlock	0..1	aggr	This represents an informal remark. Note that this is not modeled as annotation, since these remark is still essential part of the requirement. **Tags:**xml.sequenceOffset=60
supporting Material	DocumentationBlock	0..1	aggr	This represents an informal specifiaction of the supporting material. **Tags:**xml.sequenceOffset=50
testedItem	Traceable	*	ref	This assocation represents the ability to trace on the same specification level. This supports for example the of acceptance tests. **Tags:**xml.sequenceOffset=70
type	String	1	attr	This attribute allows to denote the type of requirement to denote for example is it an "enhancement", "new feature" etc. **Tags:**xml.sequenceOffset=7
useCase	DocumentationBlock	0..1	aggr	This describes the relevant use cases. Note that formal references to use cases should be done in the trace relation. **Tags:**xml.sequenceOffset=35

Table 2.4: StructuredReq

3 Life Cycle of AUTOSAR Definitions

In order to support evolution and backward compatibility of the standardized model elements like port prototype blueprints, port interfaces, keyword abbreviations, SW-Cs (in ASW) or of the API of a BSW module etc. AUTOSAR supports life cycles. The meta model and the details of the application of this meta model is specified in chapter "Life Cycle Support" of Generic Structure Template [7].

[TPS_STDT_00038] Life Cycle Support ⌈STDT is able to express information about the state of the blueprints by references from within a `LifeCycleInfoSet`.⌋*(RS_-STDT_00016)*

[TPS_STDT_00064] Applied Life Cycle Information Sets on AUTOSAR provided Models (M1) ⌈

The following life cycle states are applied for AUTOSAR provided model elements. They correspond to [TPS_GST_00051]:

valid This indicates that the related entity is a valid part of the document. This is the default.

draft This indicates that the related entity is introduced newly in the model but still experimental. This information is published but is subject to be changed without backward compatibility management.

obsolete This indicates that the related entity is obsolete and kept in the model for compatibility reasons. If this tag is set, the note shall express the recommended alternative solution.

preliminary This indicates that the related entity is preliminary in the model. It is subject to be changed without backwards compatibility management. An AUTOSAR release does not contain such elements. It is intended for AUTOSAR internal development.

removed This indicates that the related entity is removed from the model. It shall not be used and should not even appear in documents. An AUTOSAR release does not contain such elements. It is intended for AUTOSAR internal development.

Even if such removed elements are not included in an `.arxml` they can still be referenced in a `LifeCycleInfoSet` by using the ≪`atpUriDef`≫ attribute of type `Referrable`: `lcObject`, respectively `useInstead`.

shallBecomeMandatory This indicates that the related entity should be mandatory from the semantical perspective and will become mandatory in future. It is yet left optional to avoid backwards compatibility issues. Such elements should be provided whenever possible.

If an object is not referenced in a `LifeCycleInfoSet`, the related entity is a valid part of the current model.⌋*(RS_STDT_00025)*

Note that according to [TPS_STDT_00064] if there is no life cycle information for an element then it is defined that the element is valid. In other words, in general there is no need to define a `LifeCycleInfoSet` with `defaultLcState` "valid". Nevertheless there might be use cases when it could be useful to explicitly define such a `LifeCycleInfoSet`. For example if element "x" gets life cycle state "obsolete" and subsequently this is identified as an error and the life cycle returns back to "valid". This could be documented in such a `LifeCycleInfoSet`.

Listing 3.1 provides the ARXML representation of the life cycle according to [TPS_-GST_00051] respectively [TPS_STDT_00064].

Listing 3.1: AUTOSAR Standard `LifeCycleStateDefinitionGroup`

```
<!-- LifeCycleStateDefinitionGroup: AutosarLifeCycleStates -->
      <LIFE-CYCLE-STATE-DEFINITION-GROUP>
        <SHORT-NAME>AutosarLifeCycleStates</SHORT-NAME>
        <LONG-NAME>
          <L-4 L="EN">Life Cycle Definitions used in AUTOSAR
              Standards</L-4>
        </LONG-NAME>
        <DESC>
          <L-2 L="EN">This set represents the life cycle
              definitions used by AUTOSAR on M1 and M2 level. See
              also [TPS_GST_00051] respectively [TPS_GST_00064].</
              L-2>
        </DESC>
        <LC-STATES>
  <!-- LifeCycleState: valid -->
          <LIFE-CYCLE-STATE>
            <SHORT-NAME>valid</SHORT-NAME>
            <LONG-NAME>
              <L-4 L="EN">VALID</L-4>
            </LONG-NAME>
            <DESC>
              <L-2 L="EN">This indicates that the related entity
                  is a valid part of the document. This is the
                  default.</L-2>
            </DESC>
          </LIFE-CYCLE-STATE>
<!-- LifeCycleState: draft -->
          <LIFE-CYCLE-STATE>
            <SHORT-NAME>draft</SHORT-NAME>
            <LONG-NAME>
              <L-4 L="EN">DRAFT</L-4>
            </LONG-NAME>
            <DESC>
              <L-2 L="EN">This indicates that the related entity
                  is introduced newly in the (meta) model but
                  still experimental. This information is
                  published but is subject to be changed without
                  backward compatibility management.</L-2>
            </DESC>
          </LIFE-CYCLE-STATE>
<!-- LifeCycleState: obsolete -->
          <LIFE-CYCLE-STATE>
            <SHORT-NAME>obsolete</SHORT-NAME>
            <LONG-NAME>
              <L-4 L="EN">OBSOLETE</L-4>
            </LONG-NAME>
            <DESC>
              <L-2 L="EN">This indicates that the related entity
                  is obsolete and kept in the (meta) model for
                  compatibility reasons. </L-2>
            </DESC>
            <INTRODUCTION>
              <P>
                <L-1 L="EN">If this life cycle state is set, the
```

```
<TT TYPE="ARMetaClassRole">LifeCycleInfo.remark</TT> shall express the
    recommended alternative solution.</L-1>
                    </P>
                  </INTRODUCTION>
                </LIFE-CYCLE-STATE>
      <!-- LifeCycleState: preliminary -->
                <LIFE-CYCLE-STATE>
                  <SHORT-NAME>preliminary</SHORT-NAME>
                  <LONG-NAME>
                    <L-4 L="EN">PRELIMINARY</L-4>
                  </LONG-NAME>
                  <DESC>
                    <L-2 L="EN">This indicates that the related entity
                        is preliminary in the (meta) model. It is
                        subject to be changed without backwards
                        compatibility management. An AUTOSAR release
                        does not contain such elements. It is intended
                        for AUTOSAR internal development.</L-2>
                  </DESC>
                </LIFE-CYCLE-STATE>
      <!-- LifeCycleState: removed -->
                <LIFE-CYCLE-STATE>
                  <SHORT-NAME>removed</SHORT-NAME>
                  <LONG-NAME>
                    <L-4 L="EN">REMOVED</L-4>
                  </LONG-NAME>
                  <DESC>
                    <L-2 L="EN">This indicates that the related entity
                        is still in the (meta) model for whatever reason
                        . It shall not be used and should not even
                        appear in documents. </L-2>
                  </DESC>
                  <INTRODUCTION>
                    <P>
                      <L-1 L="EN">An AUTOSAR release does not contain
                          such elements. It is intended for AUTOSAR
                          internal development. <BR /> Removed elements
                          are not included in an .arxml delivery but can
                           be referenced in a LifeCycleInformationSet by
                           using the
<TT TYPE="ARStereotype">atpUriDef</TT> attributes of type
<TT TYPE="ARMetaClass">Referrable</TT>:
<TT TYPE="ARMetaClassRole">LifeCycleInfo.lcObject</TT>, respectively
<TT TYPE="ARMetaClassRole">LifeCycleInfo.useInstead</TT>.</L-1>
                    </P>
                  </INTRODUCTION>
                </LIFE-CYCLE-STATE>
      <!-- LifeCycleState: shallBecomeMandatory -->
                <LIFE-CYCLE-STATE>
                  <SHORT-NAME>shallBecomeMandatory</SHORT-NAME>
                  <LONG-NAME>
                    <L-4 L="EN">SHALL-BECOME-MANDATORY</L-4>
                  </LONG-NAME>
                  <DESC>
                    <L-2 L="EN">This indicates that the related entity
                        should be mandatory from the semantical
                        perspective and will become mandatory in future.
                         It is yet left optional to avoid backwards
                        compatibility issues. Such elements should be
                        provided whenever possible.</L-2>
                  </DESC>
                </LIFE-CYCLE-STATE>
              </LC-STATES>
            </LIFE-CYCLE-STATE-DEFINITION-GROUP>
```

4 The Principles of Blueprints

[TPS_STDT_00002] The Principles of Blueprints ⌈This chapter describes the support of the AUTOSAR meta-model for the pre-definition of model elements taken as the basis for further modeling. These pre-definitions are called blueprints.⌋*(RS_STDT_-00001)*

For example, an authoring tool provides the such predefined PortInterface as a kind of toolbox from which the definitions can be copied to a project.

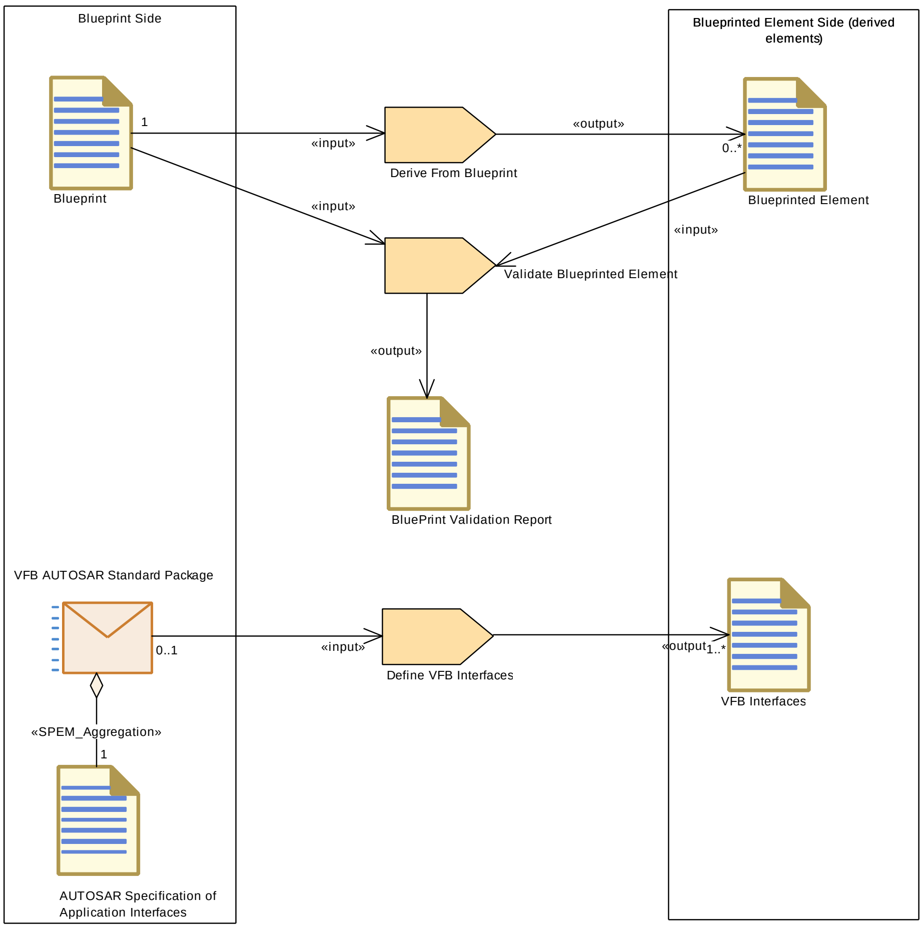


Figure 4.1: Blueprint methodology approach

Figure 4.1 illustrates the usecase. The blueprint is on one hand used as an input to derive objects (DeriveFromBlueprint) and later also used to validate the derived objects. As an example the figure shows that the application interfaces are used to derive VFB interfaces (namely PortInterfaces).

4.1 Abstract pattern for Blueprints

The blueprint approach is represented by the abstract blueprint structure as shown in figure 4.2. It is based on three entities:

- **Blueprint**, represented by AtpBlueprint, acts as the predefinition of the element. Basically it follows the same structure as the derived elements.

  But there might be additional elements to support the fact that it is a blueprint. An example for this is that PortPrototypeBlueprint also specifies initValues which is not the case for PortPrototype which get their initial values from appropriate ComSpecs.

- **Blueprinted Element**, represented by AtpBlueprintable, acts as the element which was derived from the Blueprint. These elements are derived from blueprints mainly by copy and refine. This "refine" may add further attribute values, update shortName etc. The details of possible refinements are specified for each blueprint individually.

  Note that the subsequent processing of blueprinted elements (e.g. RTE generation) does not refer to the blueprints anymore.

- **Blueprint Mapping**, represented by AtpBlueprintMapping, acts as a reference between blueprints and their derived elements. The main purpose of this blueprint mapping is to
  - provide the ability to validate for each derived element that they conform to the blueprint.
  - reflect the fact that the derived elements are part of a common concept.

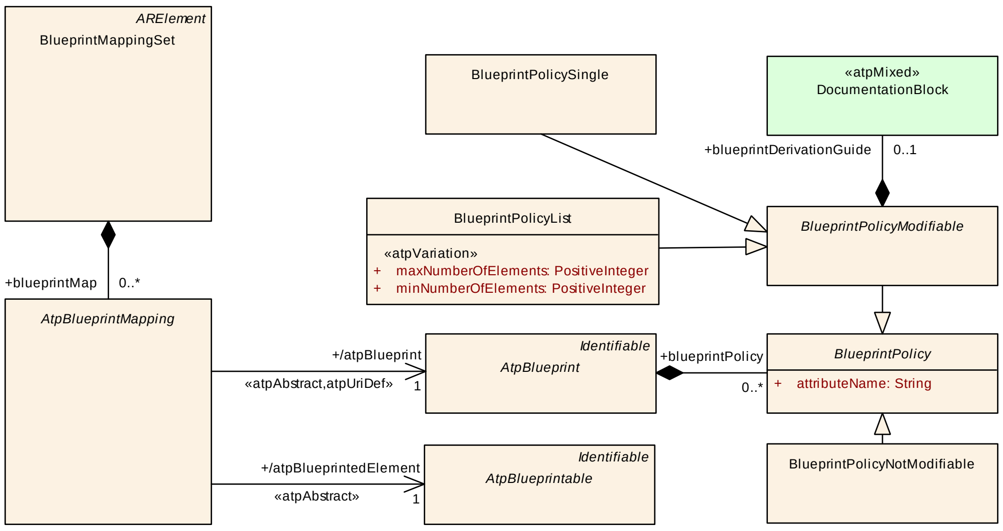


Figure 4.2: Abstract Blueprint Structure

Meta-classes for elements eligible for blueprinting are defined as specializations of AtpBlueprintable while meta-classes for blueprints are defined as specializations of AtpBlueprint. An example is given in figure 4.3.

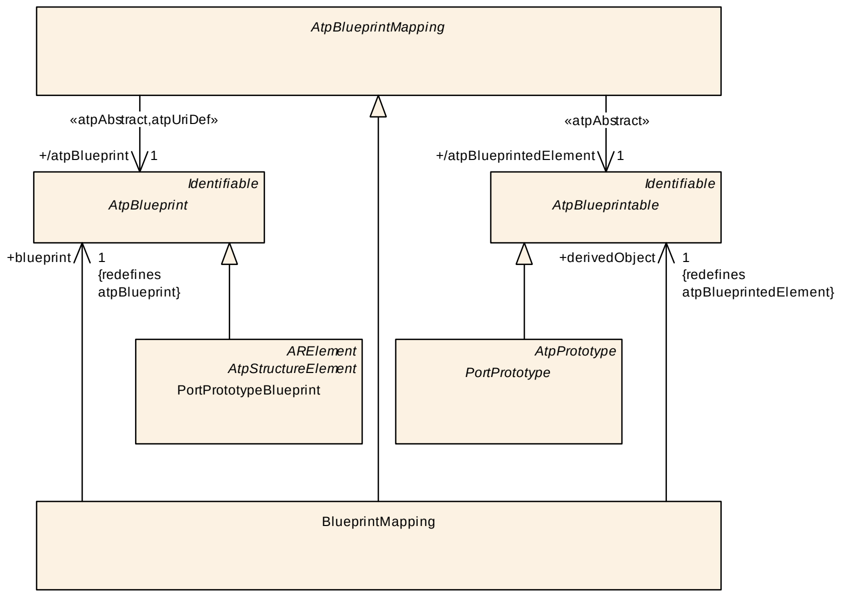


Figure 4.3: Port Blueprints as an example for separate meta-classes for Blueprint and blueprinted Element

[TPS_STDT_00072] Same Meta Class For Blueprints and Derived Objects ⌈For most of the elements eligible for blueprinting, no extra meta-class is required because the same meta-class applies for blueprints and blueprinted elements. The meta-class of such an element inherits from both `AtpBlueprint` and `AtpBlueprintable`.⌋ *(RS_STDT_00017)* An example is given in figure 4.4.

[TPS_STDT_00041] Constraints may be violated in Blueprints ⌈For blueprints using the same meta-class as the derived objects, the constraints defined for these objects may be violated by the blueprints such as:

- Required attributes may be missing.
- Referenced objects may not exist. Strictly speaking, references in blueprints can all be considered as ≪`atpUriDef`≫

⌋*(RS_STDT_00002, RS_STDT_00006, RS_STDT_00007)*

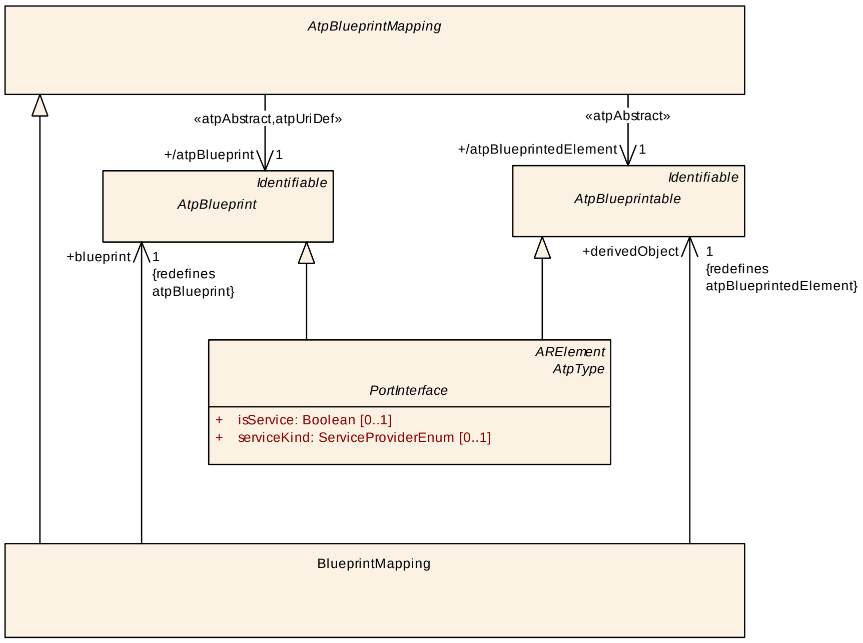


Figure 4.4: PortInterface Blueprints as an example for using the same meta-class for Blueprint and blueprinted Element

[TPS_STDT_00033] Recognize Blueprints ⌈According to [7] the blueprints reside in a package of `category` "BLUEPRINT". Downstream AUTOSAR Tools such as RTE-generator shall ignore Elements living in a package of `category` "BLUEPRINT".⌋ *(RS_STDT_00006, RS_STDT_00007)*

Blueprints are specializations of `AtpBlueprint`. Introduction of standardization therefore does not introduce compatibility problems to existing templates. Note that since AUTOSAR 4.0.3 `AtpBlueprint.shortNamePattern` is replaced by `Identifier.namePattern` resp. `CIdentifier.namePattern`. In addition since AUTOSAR 4.4.0 `blueprintValue` exists and is used e.g. in the context of ARMQL (AUTOSAR Model Query Language).

[TPS_STDT_00032] BlueprintPolicy ⌈Blueprintable elements shall be characterized by BlueprintPolicy to indicate whether they will be modifiable or not modifiable.

- `BlueprintPolicyNotModifiable` means, that the related attribute is not modifiable during the blueprinting (see listing 4.1).
- `BlueprintPolicyList` means, that the related attribute is modifiable during the blueprinting. It applies only to an attribute with upper multiplicity greater than 1 (see listing 4.2).
- `BlueprintPolicySingle` means, that the related attribute is modifiable during the blueprinting. It applies only to an attribute with upper multiplicity equal 1 (see listing 4.3).

⌋*(RS_STDT_00040)*

[constr_2590] One BlueprintPolicy is allowed ⌈For each attribute of a blueprint, at most one BlueprintPolicy is allowed.⌋*()*

[constr_2591] BlueprintPolicyNotModifiable ⌈If BlueprintPolicyNotModifiable is assigned to an attribute, then during blueprinting it is not allowed to modify the value of the attribute and all its contained content.⌋*()*

[constr_2592] No BlueprintPolicy ⌈If no BlueprintPolicy is assigned to an attribute, then arbitrary modifications are allowed while deriving from the blueprint.⌋*()*

[constr_2593] Expression for identifying the attribute a BlueprintPolicy relates to ⌈The expression language for identifying the related attribute of a BlueprintPolicy is a subset version of xpath, see [8]. For navigation over the model we use the names as they are used in XML.⌋*()*

[TPS_STDT_00039] Xpath Expressions for BlueprintPolicy ⌈The BlueprintPolicy uses a subset of xpath expressions described in Table 4.1. Other xpath expressions shall not be use to avoid complexity. The root node is the blueprint owning the BlueprintPolicy.⌋*(RS_STDT_00040)*

Path Expressions	Description
nodename	Selects all nodes with the name "nodename"
/	Selects from the root node
@	Selects attributes
@<attribute>='<value>'	Selects an element node, which has the <attribute> set to <value>
text()='<value>'	Selects an element node, which contains the text <value>
*	Matches any element node
[n]	Selects the n-th element node

Table 4.1: Allowed xpath expressions in BlueprintPolicy

The xpath expression [n] in Table 4.1 starts with [1] due to the XML Path Language Specification [8]. The use of [n] is only allowed for ordered elements. One BlueprintPolicy can refine more than one attribute.

In listing 4.1 the root node is selected by the nodename (COMPU-INTERNAL-TO-PHYS). In listing 4.2 the root node is selected by nodename/nodename/* (COMPU-INTERNAL-TO-PHYS/COMPU-SCALES/*).

Listing 4.1: Example for BlueprintPolicyNotModifiable

```
<COMPU-METHOD>
  <SHORT-NAME>Dem_DebounceResetStatusType</SHORT-NAME>
  <CATEGORY>TEXTTABLE</CATEGORY>
  <BLUEPRINT-POLICYS>
    <BLUEPRINT-POLICY-NOT-MODIFIABLE>
      <ATTRIBUTE-NAME>COMPU-INTERNAL-TO-PHYS</ATTRIBUTE-NAME>
    </BLUEPRINT-POLICY-NOT-MODIFIABLE>
  </BLUEPRINT-POLICYS>
  <COMPU-INTERNAL-TO-PHYS>
    <COMPU-SCALES>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0x00</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0x00</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>DEM_DEBOUNCE_STATUS_FREEZE</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0x01</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0x01</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>DEM_DEBOUNCE_STATUS_RESET</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
    </COMPU-SCALES>
  </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

Listing 4.2: Example for BlueprintPolicyList

```
<COMPU-METHOD>
  <SHORT-NAME>Dcm_SecLevelType</SHORT-NAME>
  <CATEGORY>TEXTTABLE</CATEGORY>
  <BLUEPRINT-POLICYS>
    <BLUEPRINT-POLICY-LIST>
      <ATTRIBUTE-NAME>COMPU-INTERNAL-TO-PHYS/COMPU-SCALES/*</ATTRIBUTE-NAME>
      <BLUEPRINT-DERIVATION-GUIDE>
        <P>
          <L-1 L="EN">The range 0x01...0x3F is used configuration dependent</L
              -1>
        </P>
        <P>
          <L-1 L="EN">The range 0x40...0xFF is reserved by document</L-1>
        </P>
      </BLUEPRINT-DERIVATION-GUIDE>
      <MAX-NUMBER-OF-ELEMENTS BLUEPRINT-VALUE="undefined">undefined</MAX-
          NUMBER-OF-ELEMENTS>
      <MIN-NUMBER-OF-ELEMENTS>1</MIN-NUMBER-OF-ELEMENTS>
    </BLUEPRINT-POLICY-LIST>
  </BLUEPRINT-POLICYS>
  <COMPU-INTERNAL-TO-PHYS>
    <COMPU-SCALES>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0x00</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0x00</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>DCM_SEC_LEV_LOCKED</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
    </COMPU-SCALES>
  </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

The listing 4.3 illustrates the use of `BlueprintPolicySingle`.

Listing 4.3: Example for BlueprintPolicySingle

```
    <PORT-PROTOTYPE-BLUEPRINT>
      <SHORT-NAME NAME-PATTERN="{anyName}">AFbForCmft</SHORT-NAME>
      <LONG-NAME>
        <L-4 L="EN">Acceleration Feedback for Comfort</L-4>
      </LONG-NAME>
      <DESC>
        <L-2 L="EN">Cluster of information regarding acceleration and
            acceleration saturation feedbacks from Vehicle Longitudinal
            Control (VLC) to Adaptive Cruise Control (ACC). This information
            is used for comfort reasons.</L-2>
      </DESC>
      <BLUEPRINT-POLICYS>
        <BLUEPRINT-POLICY-SINGLE>
          <ATTRIBUTE-NAME>INTERFACE-REF</ATTRIBUTE-NAME>
          <BLUEPRINT-DERIVATION-GUIDE>
            <P>
              <L-1 L="EN">Shall only refer to an interface of vendor xyz
                  with the same shortname.</L-1>
            </P>
          </BLUEPRINT-DERIVATION-GUIDE>
        </BLUEPRINT-POLICY-SINGLE>
      </BLUEPRINT-POLICYS>
      <INTERFACE-REF DEST="SENDER-RECEIVER-INTERFACE">/AUTOSAR/
          AISpecification/PortInterfaces_Blueprint/AFbForCmft1</INTERFACE-REF
          >
    </PORT-PROTOTYPE-BLUEPRINT>
```

In listing 4.4 the BlueprintPolicySingle selects an element node with attribute which equals a defined string (PORTS/P-PORT-PROTOTYPE/SHORT-NAME[@NAME-PATTERN='{Name}_AsymDecrypt']).

Listing 4.4: Example for BlueprintPolicySingle with attribute name pattern

```
<BLUEPRINT-POLICY-SINGLE>
   <ATTRIBUTE-NAME>PORTS/P-PORT-PROTOTYPE/SHORT-NAME[@NAME-
      PATTERN='{Name}_AsymDecrypt']</ATTRIBUTE-NAME>
   <BLUEPRINT-DERIVATION-GUIDE>
      <P>
         <L-1 L="EN">Name = {ecuc(Csm/CsmAsymDecrypt/
            CsmAsymDecryptConfig.SHORT-NAME)}</L-1>
      </P>
   </BLUEPRINT-DERIVATION-GUIDE>
</BLUEPRINT-POLICY-SINGLE>
```

This results in the selection of the element node illustrated in listing 4.5.

Listing 4.5: Selected element node <SHORT-NAME>

```
<P-PORT-PROTOTYPE>
   <SHORT-NAME NAME-PATTERN="{Name}_AsymDecrypt">AsymDecrypt</
      SHORT-NAME>
   <PROVIDED-INTERFACE-TREF DEST="CLIENT-SERVER-INTERFACE">/
      AUTOSAR/Csm/ClientServerInterfaces_Blueprint/
      CsmAsymDecrypt</PROVIDED-INTERFACE-TREF>
</P-PORT-PROTOTYPE>
```

In listing 4.6 the BlueprintPolicySingle selects an element node which contains a defined text pattern (OPERATIONS/CLIENT-SERVER-OPERATION[SHORT-

NAME/text()="ReadData"]/ARGUMENTS/ARGUMENT-DATA-PROTOTYPE[SHORT-NAME/text()="Data"]).

Listing 4.6: Example for BlueprintPolicySingle with text pattern

```
<BLUEPRINT-POLICY-SINGLE>
   <ATTRIBUTE-NAME>OPERATIONS/CLIENT-SERVER-OPERATION[SHORT-NAME
      /text()="ReadData"]/ARGUMENTS/ARGUMENT-DATA-PROTOTYPE[
      SHORT-NAME/text()="Data"]</ATTRIBUTE-NAME>
   <BLUEPRINT-DERIVATION-GUIDE>
      <P>
         <L-1 L="EN">Data = {ecuc(Dem/DemGeneral/
            DemDataElementClass.SHORT-NAME)}</L-1>
      </P>
   </BLUEPRINT-DERIVATION-GUIDE>
</BLUEPRINT-POLICY-SINGLE>
```

This results in the selection of the element node (ARGUMENTS/ARGUMENT-DATA-PROTOTYPE/SHORT-NAME) with SHORT-NAME equal to 'Data' in case (CLIENT-SERVER-OPERATION/SHORT-NAME) is equal to 'ReadData', see listing 4.7.

Listing 4.7: Example for BlueprintPolicySingle with text pattern

```
<OPERATIONS>
   <CLIENT-SERVER-OPERATION>
      <SHORT-NAME>ReadData</SHORT-NAME>
      <INTRODUCTION>
         <P>
            <L-1 L="EN">The server is not allowed to return
               E_NOT_OK, but shall always provide a valid data
               value (e.g. a default/replacement value in an error-
               case) to Dcm/Dem nevertheless the signature of the
               operation includes E_NOT_OK to ensure compatibility
               between server runnable and RTE Call API, since the
               RTE may return negative Std_Return values in certain
                cases (e.g. partition of server stopped)</L-1>
         </P>
      </INTRODUCTION>
      <ARGUMENTS>
         <ARGUMENT-DATA-PROTOTYPE>
            <SHORT-NAME>Data</SHORT-NAME>
            <TYPE-TREF DEST="IMPLEMENTATION-DATA-TYPE">/AUTOSAR/Dem
               /ImplementationDataTypes_Blueprint/DataArrayType</
               TYPE-TREF>
            <DIRECTION>OUT</DIRECTION>
         </ARGUMENT-DATA-PROTOTYPE>
      </ARGUMENTS>
      <POSSIBLE-ERROR-REFS>
         <POSSIBLE-ERROR-REF DEST="APPLICATION-ERROR">/AUTOSAR/Dem/
            ClientServerInterfaces_Blueprint/DataServices/E_OK</
            POSSIBLE-ERROR-REF>
         <POSSIBLE-ERROR-REF DEST="APPLICATION-ERROR">/AUTOSAR/Dem/
            ClientServerInterfaces_Blueprint/DataServices/E_NOT_OK<
            /POSSIBLE-ERROR-REF>
      </POSSIBLE-ERROR-REFS>
   </CLIENT-SERVER-OPERATION>
</OPERATIONS>
```

Class	***AtpBlueprint*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::AbstractBlueprintStructure			
Note	This meta-class represents the ability to act as a Blueprint. As this class is an abstract one, particular blueprint meta-classes inherit from this one.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ARPackage, *AbstractImplementationDataType*, AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, *ApplicationDataType*, BswEntryRelationshipSet, BswModuleDescription, BswModuleEntry, *BuildActionEntity*, BuildActionEnvironment, BuildActionManifest, ClientServerInterfaceToBswModuleEntryBlueprintMapping, CompuMethod, ConsistencyNeeds, DataConstr, DataTypeMappingSet, EcucDefinitionCollection, EcucDestinationUriDefSet, EcucModuleDef, FlatMap, KeywordSet, LifeCycleState, LifeCycleStateDefinitionGroup, ModeDeclarationGroup, *PortInterface*, *PortInterfaceMapping*, PortInterfaceMappingSet, PortPrototypeBlueprint, SwAddrMethod, SwBaseType, *SwComponentType*, VfbTiming			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blueprintPolicy	BlueprintPolicy	*	aggr	This role indicates whether the blueprintable element will be modifiable or not motifiable.

Table 4.2: AtpBlueprint

Class	***AtpBlueprintable*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::AbstractBlueprintStructure			
Note	This meta-class represents the ability to be derived from a Blueprint. As this class is an abstract one, particular blueprintable meta-classes inherit from this one.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ARPackage, *AbstractImplementationDataType*, AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, *ApplicationDataType*, BswEntryRelationshipSet, BswModuleDescription, BswModuleEntry, *BuildActionEntity*, BuildActionEnvironment, BuildActionManifest, CompuMethod, ConsistencyNeeds, DataConstr, DataTypeMappingSet, EcucDefinitionCollection, EcucDestinationUriDefSet, EcucModuleDef, FlatMap, KeywordSet, LifeCycleState, LifeCycleStateDefinitionGroup, ModeDeclarationGroup, *PortInterface*, *PortInterfaceMapping*, PortInterfaceMappingSet, *PortPrototype*, SwAddrMethod, SwBaseType, *SwComponentType*, VfbTiming			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.3: AtpBlueprintable

Class	***AtpBlueprintMapping*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::AbstractBlueprintStructure			
Note	This meta-class represents the ability to express a particular mapping between a blueprint and an element derived from this blueprint. Particular mappings are defined by specializations of this meta-class.			
Base	*ARObject*			
Subclasses	BlueprintMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
atpBlueprint	AtpBlueprint	1	ref	This represents the blueprint. **Stereotypes:** atpAbstract; atpUriDef **Tags:**xml.sequenceOffset=50

▽

△

Class	***AtpBlueprintMapping*** (abstract)			
atpBlueprinted Element	AtpBlueprintable	1	ref	This represents the bluprinted elements which shall be mapped to the blueprint. **Stereotypes:** atpAbstract **Tags:**xml.sequenceOffset=60

Table 4.4: AtpBlueprintMapping

Class	***BlueprintPolicy*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::AbstractBlueprintStructure			
Note	This meta-class represents the ability to indicate whether blueprintable elements will be modifiable or not modifiable.			
Base	*ARObject*			
Subclasses	*BlueprintPolicyModifiable*, BlueprintPolicyNotModifiable			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
attributeName	String	1	attr	This identifies the related attribute of a BlueprintPolicy. For navigation over the model a subset of xpath expressions is used.

Table 4.5: BlueprintPolicy

Class	**BlueprintPolicyList**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::AbstractBlueprintStructure			
Note	The class represents that the related attribute is modifiable during the blueprinting. It applies only to attribute with upper multiplicity greater than 1.			
Base	*ARObject*, *BlueprintPolicy*, *BlueprintPolicyModifiable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maxNumberOf Elements	PositiveInteger	1	attr	Maximum number of elements in list. If the maximum number is not constraint it shall be set to "undefined". **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime
minNumberOf Elements	PositiveInteger	1	attr	Minimum number of elements in the list. If the minimum number is not constraint it shall be set to "undefined". **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime

Table 4.6: BlueprintPolicyList

Class	***BlueprintPolicyModifiable*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::AbstractBlueprintStructure			
Note	The class represents that the related attribute is modifiable during the blueprinting.			
Base	*ARObject*, *BlueprintPolicy*			
Subclasses	BlueprintPolicyList, BlueprintPolicySingle			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blueprint DerivationGuide	DocumentationBlock	0..1	aggr	This role offers the possibility to give addtional information to the policy.

Table 4.7: BlueprintPolicyModifiable

Class	**BlueprintPolicyNotModifiable**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::AbstractBlueprintStructure			
Note	The class represents that the related attribute is not modifiable during the blueprinting.			
Base	*ARObject*, *BlueprintPolicy*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 4.8: BlueprintPolicyNotModifiable

Class	**BlueprintPolicySingle**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::AbstractBlueprintStructure			
Note	The class represents that the related attribute is modifiable during the blueprinting. It applies only to attribute with upper multiplicity equal 1.			
Base	*ARObject*, *BlueprintPolicy*, *BlueprintPolicyModifiable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 4.9: BlueprintPolicySingle

4.2 Mapping of Blueprints to blueprinted Elements

In many cases it will be necessary to identify the relationship of a blueprinted element (e.g. `PortPrototype`) to the corresponding blueprint (e.g. `PortPrototypeBlueprint`) after the blueprinted element has been created according to the blueprint.

For this purpose it would theoretically be possible to establish a reference from `AtpBlueprintable` to `AtpBlueprint` that identifies the pair of related model artifacts. However, this kind of information is relevant only in a narrow scope and does - as mentioned before - not impact the downstream model handling.

Therefore, a `AtpBlueprintMapping` is introduced which refers to both `AtpBlueprintable` and `AtpBlueprint` (see figure 4.2). The `AtpBlueprintMapping` is in turn aggregated at a container for the creation of blueprint mappings, the `BlueprintMappingSet`.

In previous AUTOSAR Releases a specialization of `AtpBlueprintMapping` was created for each particular meta class eligible for blueprinting. This has been replaced by one particular specialization (`BlueprintMapping`)[1].

[1]For compatibility reasons, the abstract patten was not changed. The previous specializations `PortInterfaceBlueprintMapping` and `PortPrototypeBlueprintMapping` are removed.

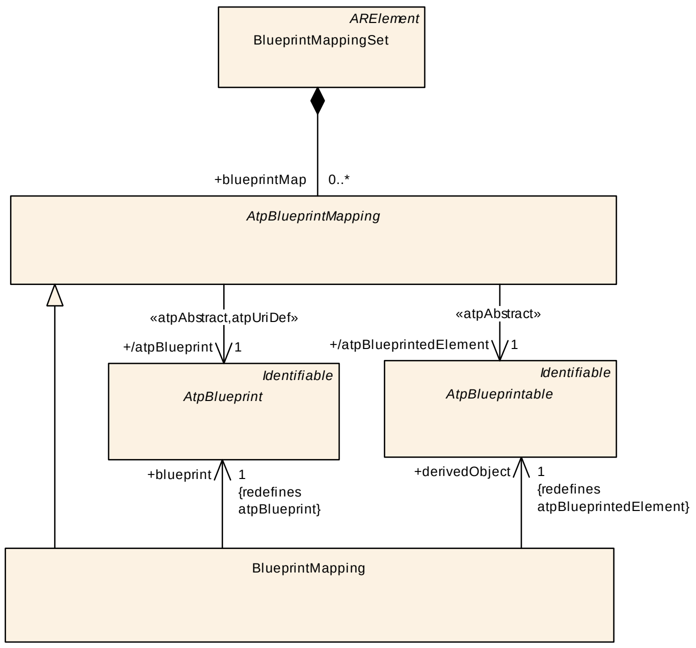


Figure 4.5: Mapping of Derived Objects and their Blueprints

[constr_2566] Blueprintmapping shall map appropriate elements ⌈`BlueprintMapping` shall map elements which represent a valid pair of blueprint / derived object. In most of the cases this means that `blueprint` and `derivedObject` shall refer to objects of the same meta-class.⌋*()*

Class	**BlueprintMappingSet**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::BlueprintMapping			
Note	This represents a container of mappings between "actual" model elements and the "blueprint" that has been taken for their creation. **Tags:**atp.recommendedPackage=BlueprintMappingSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blueprintMap	AtpBlueprintMapping	*	aggr	This represents a particular blueprint map in the set.

Table 4.10: BlueprintMappingSet

4.3 General Rules for Compliance of blueprint and blueprinted element

[TPS_STDT_00005] Compliance with Blueprints ⌈Constraints [constr_2554] and [TPS_STDT_00087] apply in general for the compliance of blueprints with the derived objects.⌋*(RS_STDT_00017)*

[constr_2554] Derived objects shall match the blueprints ⌈Unless specified explicitly otherwise, the attributes of the blueprint shall appear in the derived objects.

As an exception `namePattern` and `blueprintValue` may **not** be copied.⌋*()*

[TPS_STDT_00087] Derived objects may have more attributes than the blueprints ⌈Unless specified explicitly otherwise, derived objects may have more attributes than the blueprints. Such attributes can be

- additional values if the upper multiplicity of the attribute in the meta-model is greater than 1
- those specified by the related templates but not specified in the blueprint

⌋*(RS_STDT_00017)*

[TPS_STDT_00085] Compatibility of `longName`, `desc` and `introduction` of blueprint and blueprinted element ⌈Elements derived from blueprints are allowed to

- change `longName`
- change `desc`
- change `introduction`

⌋*(RS_STDT_00017)*

Note that [TPS_STDT_00085] includes the ability to add text in a further language.

Note that `introduction` should not be used to describe the derivation of objects from the blueprint. See [TPS_STDT_00048] for details.

[TPS_STDT_00086] Specify a name pattern or a blueprint value in blueprints ⌈For each blueprint, a `namePattern` or a `blueprintValue` shall be specified if the `shortName` respectively a `symbol` is not fixed but intended to be defined when objects are derived from a blueprint. This is used to verify the appropriate naming of the derived objects ([constr_2553]).⌋*(RS_STDT_00017)*

[constr_2553] `shortName` shall follow the pattern defined in the Blueprint ⌈The `shortName` respectively `symbol` of the derived objects shall follow the pattern defined in `namePattern` or `blueprintValue` of the blueprint according to [TPS_STDT_00086]⌋*()*

[constr_2570] No Blueprints in system descriptions ⌈There shall be no blueprints in system descriptions. In consequence of this blueprint elements shall be referenced only from blueprints and `AtpBlueprintMapping`s. Due to ≪`atpUriDef`≫, the references from `AtpBlueprintMapping` do not need to be resolved in system descriptions.⌋*()*

[constr_2571] Outgoing references from Blueprints ⌈Note that outgoing references from Blueprints are basically not limited. Practically, references to objects living in a package of category EXAMPLE should not occur.⌋*()*

Reason for [constr_2571] is the fact that these examples then also shall exist in the target system description but not as example. In such a case the example would take the role of a blueprint.

Figure 4.6 illustrates a scenario with standardized objects, blueprints and project related objects.

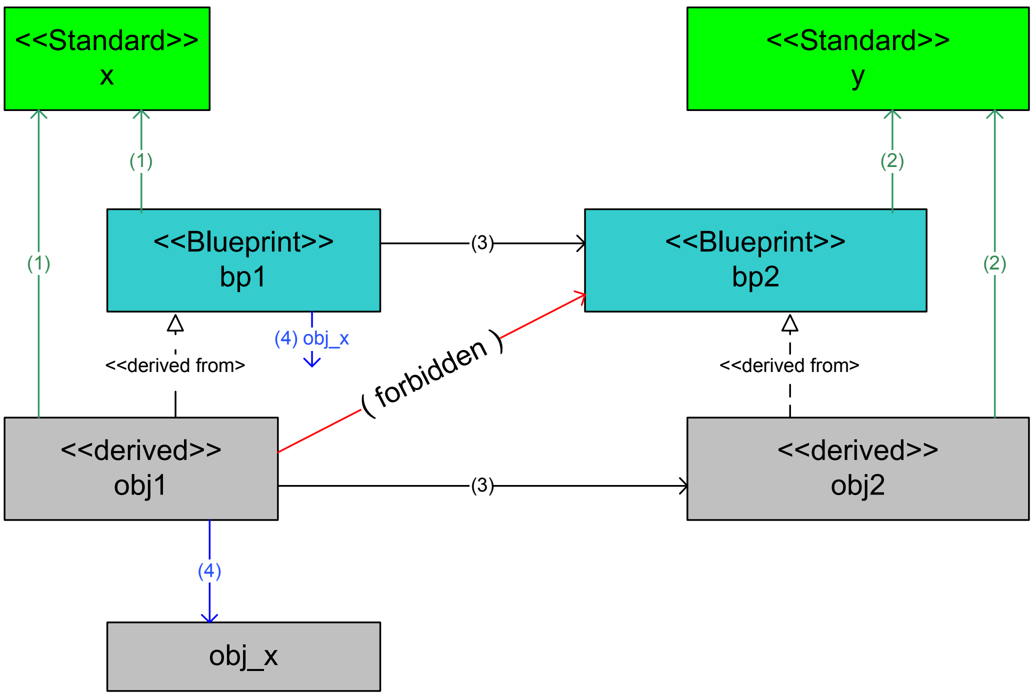


Figure 4.6: Relations between Blueprints, "Derived Objects" and "Standardized Objects"

This diagram in particular illustrates how references in blueprints shall be handled:

[TPS_STDT_00051] Handling references when deriving objects from blueprints ⌈

- Blueprints may reference standardized objects. These references also exist in the derived objects (1), (2).
- Blueprints may reference other blueprints (3). These references need to be replaced in order to meet [constr_2546]. Therefore a reference from a derived object to a blueprint is not allowed.
- Blueprints may contain references to arbitrary objects (4). According to [TPS_STDT_00041] it is allowed that these objects even do not exist. Nevertheless to meet [constr_2554] such references shall be copied to the derived objects and the referenced objects shall exist in the target system description.

⌋*(RS_STDT_00013, RS_STDT_00017)*

[TPS_STDT_00034] Integrity of Blueprints ⌈The integrity of blueprints can be established by applying references to blueprints of related objects. For example, a blueprint of a `BswModuleDescription` may refer to a blueprint of `BswModuleEntry`.⌋*(RS_STDT_00027)*

[constr_2546] References in derived model elements ⌈Model elements derived from blueprints shall never refer to model elements that are blueprints.⌋*()*

Note: A blueprint may refer to another blueprint. When deriving objects such a reference shall be replaced such that the new reference target is an object derived from the corresponding reference target in the blueprint.

[TPS_STDT_00065] Nested Blueprint Can be Used as Blueprint of its own ⌈If specialization of AtpBlueprint aggregates specialization of AtpBlueprint, then the such aggregated specialization of AtpBlueprint acts as a blueprint on its own and can be derived beyond the context of objects derived from the aggregating specialization of AtpBlueprint. This definition allows to create blueprints which are not specializations of ARElement.

In other words, If a blueprint contains blueprints, the "inner" blueprints can be derived independent from derived objects of the "outer" blueprint.⌋*(RS_STDT_00001, RS_STDT_00033)*

See chapter 5.8 for an use case of [TPS_STDT_00065].

[TPS_STDT_00047] Ignore Blueprint Attributes in Non Blueprints ⌈AUTOSAR Tools which do not process blueprints such as RTE-generator shall ignore Identifier.namePattern resp. CIdentifier.namePattern and blueprintValue.

The attributes Identifier.namePattern resp. CIdentifier.namePattern and blueprintValue should be removed when deriving objects from blueprints.⌋*(RS_STDT_00003, RS_STDT_00004, RS_STDT_00006, RS_STDT_00007)*

[TPS_STDT_00048] Express Decisions when Deriving Objects ⌈Applying VariationPoint is a suitable way to express intended decisions to be made when deriving objects from blueprints. In this case the value of the UML tag vh.latestBindingTime is blueprintDerivationTime and VariationPoint.blueprintCondition, VariationPoint.formalBlueprintGenerator respectively AttributeValueVariationPoint.blueprintValue shall be used to express the intended derivation.⌋*(RS_STDT_00008, RS_STDT_00018, RS_STDT_00019)*

[TPS_STDT_00028] Resolving VariationPoint in Blueprints ⌈If a VariationPoint has only blueprintValue respectively blueprintCondition, formalBlueprintGenerator but not swSyscond nor postBuildVariantCondition it shall be resolved when deriving elements.⌋*(RS_STDT_00014, RS_STDT_00015, RS_STDT_00019, RS_STDT_00020)*

Please refer to Generic Structure Template [7] for the following aspects:

- Even if BindingTimeEnum does not contain the value blueprintDerivationTime, there are still VariationPoints which shall be bound on blueprint derivation. This is specified as blueprintDerivationTime in the UML tag vh.latestBindingTime at the variation point in the meta model.
- In [constr_2537] VariationPoint is limited to SwComponentType, BswModuleDescription, Documentation, even if the meta model supports variation point on any PackageableElement.

[constr_2564] VariationPoint in Blueprints of PackageableElement ⌈To support standardization, constraint [constr_2537] in [7] is relaxed for blueprints. This means in particular, that all PackageableElements which inherit from AtpBlueprint and live in a package of category BLUEPRINT may have a VariationPoint.

In this case vh.latestBindingTime is considered as blueprintDerivationTime even if the meta model still states systemDesignTime for PackageableElement.⌋*()*

See chapter 5 for such elements.

- See [constr_2557]: System configurations shall not contain VariationPoints with vh.latestBindingTime set to blueprintDerivationTime.
- [constr_2558]: If vh.latestBindingTime is blueprintDerivationTime then there shall only be blueprintCondition, formalBlueprintGenerator respectively blueprintValue.
- See [constr_2559]: VariationPoints shall not be nested. In particular this means that there shall not exist a VariationPoint within the DocumentationBlock in the role blueprintCondition in a VariationPoint.
- See [constr_2567]: Attribute Value Blueprints should contain undefined.

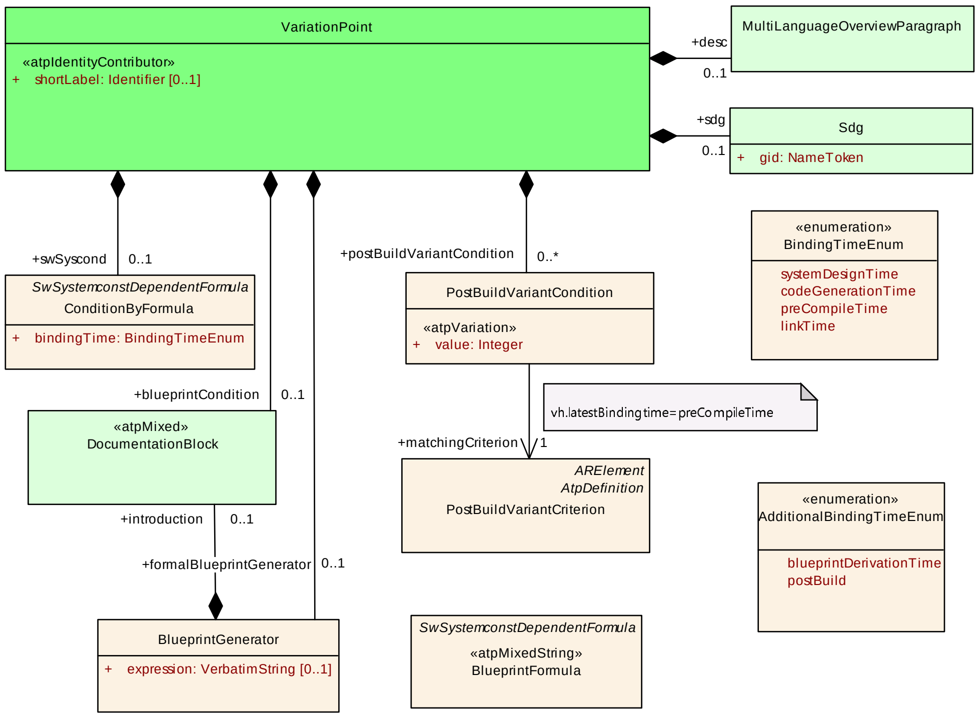


Figure 4.7: Variation Point

Class	**VariationPoint**			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
Note	This meta-class represents the ability to express a "structural variation point". The container of the variation point is part of the selected variant if swSyscond evaluates to true and each postBuildVariant Criterion is fulfilled.			
Base	*ARObject*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
blueprint Condition	DocumentationBlock	0..1	aggr	This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint. Note that variationPoints are not allowed within a blueprintCondition. **Tags:**xml.sequenceOffset=28
desc	MultiLanguageOverview Paragraph	0..1	aggr	This allows to describe shortly the purpose of the variation point. **Tags:**xml.sequenceOffset=20
formalBlueprint Generator	BlueprintGenerator	0..1	aggr	This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint by using ARMQL. Note that variationPoints are not allowed within a formal BlueprintGenerator. **Tags:** atp.Status=draft xml.sequenceOffset=30
postBuildVariant Condition	PostBuildVariant Condition	*	aggr	This is the set of post build variant conditions which all shall be fulfilled in order to (postbuild) bind the variation point. **Tags:**xml.sequenceOffset=40
sdg	Sdg	0..1	aggr	An optional special data group is attached to every variation point. These data can be used by external software systems to attach application specific data. For example, a variant management system might add an identifier, an URL or a specific classifier. **Tags:**xml.sequenceOffset=50
shortLabel	Identifier	0..1	attr	This provides a name to the particular variation point to support the RTE generator. It is necessary for supporting splitable aggregations and if binding time is later than codeGenerationTime, as well as some RTE conditions. It needs to be unique with in the enclosing Identifiables with the same ShortName. **Stereotypes:** atpIdentityContributor **Tags:**xml.sequenceOffset=10
swSyscond	ConditionByFormula	0..1	aggr	This condition acts as Binding Function for the Variation Point. Note that the mulitplicity is 0..1 in order to support pure postBuild variants. **Tags:**xml.sequenceOffset=30

Table 4.11: VariationPoint

[TPS_STDT_00030] Blueprint of `VariationPoint` ⌈A blueprint may contain `VariationPoint` with `vh.latestBindingTime` set to `blueprintDerivationTime`. These are considered as kind of blueprint of variation points which shall be handled when deriving objects. The following options apply for the container of the `VariationPoint` according to chosen approach for blueprint derivation:

1. If `blueprintCondition` is specified: resolved manually

2. If formalBlueprintGenerator is specified: resolved by a module generator. The resolver approach is formalized using ARMQL. Note that in this case it is also likely that multiple objects are created by the module generator.

After resolving the VariationPoint by one of these conditions the remaining variation is converted to a subsequent VariationPoint.⌋*(RS_STDT_00020)*

[TPS_STDT_00044] Transferring VariationPoint ⌈Unless specified explicitly otherwise, VariationPoints with vh.latestBindingTime **not** set to BlueprintDerivationTime should be transferred to the derived objects (see also [TPS_STDT_00087]). Thereby the shortLabel of the VariationPoint may be adapted according to the specification in the blueprintCondition and formalBlueprintGenerator.⌋*(RS_STDT_00020)*

[constr_2556] No Blueprint Motivated VariationPoints in AUTOSAR Descriptions ⌈AUTOSAR descriptions which are not blueprints shall not have blueprintCondition, formalBlueprintGenerator nor blueprintValue.⌋*()*

[constr_2569] Purely Blueprint Motivated VariationPoints ⌈VariationPoints with vh.latestBindingTime set to blueprintDerivationTime shall have only blueprintCondition or formalBlueprintGenerator respectively blueprintValue.⌋*()*

[TPS_STDT_00045] Transferring Objects in General ⌈Objects resp. references without VariationPoint shall be transferred to the derived objects. Thereby the namePatterns and the blueprintValues of the referenced Blueprints also apply for rewriting the shortName path in the reference.⌋*(RS_STDT_00020)*

For more details about VariationPoint refer to [7], as all constraints are summarized there.

[TPS_STDT_00046] Configuration dependent properties ⌈Some data types specify configuration-dependent properties like limits, base types etc. This is supported by an additional attribute blueprintValue in the AttributeValueVariationPoint.⌋*(RS_STDT_00020)*

An example for [TPS_STDT_00046] is:

```
NvM_BlockIdType Range:  0..2\^(16- NvMDatasetSelectionBits)-1
Dem_RatioIdType Type: uint8, uint16
```

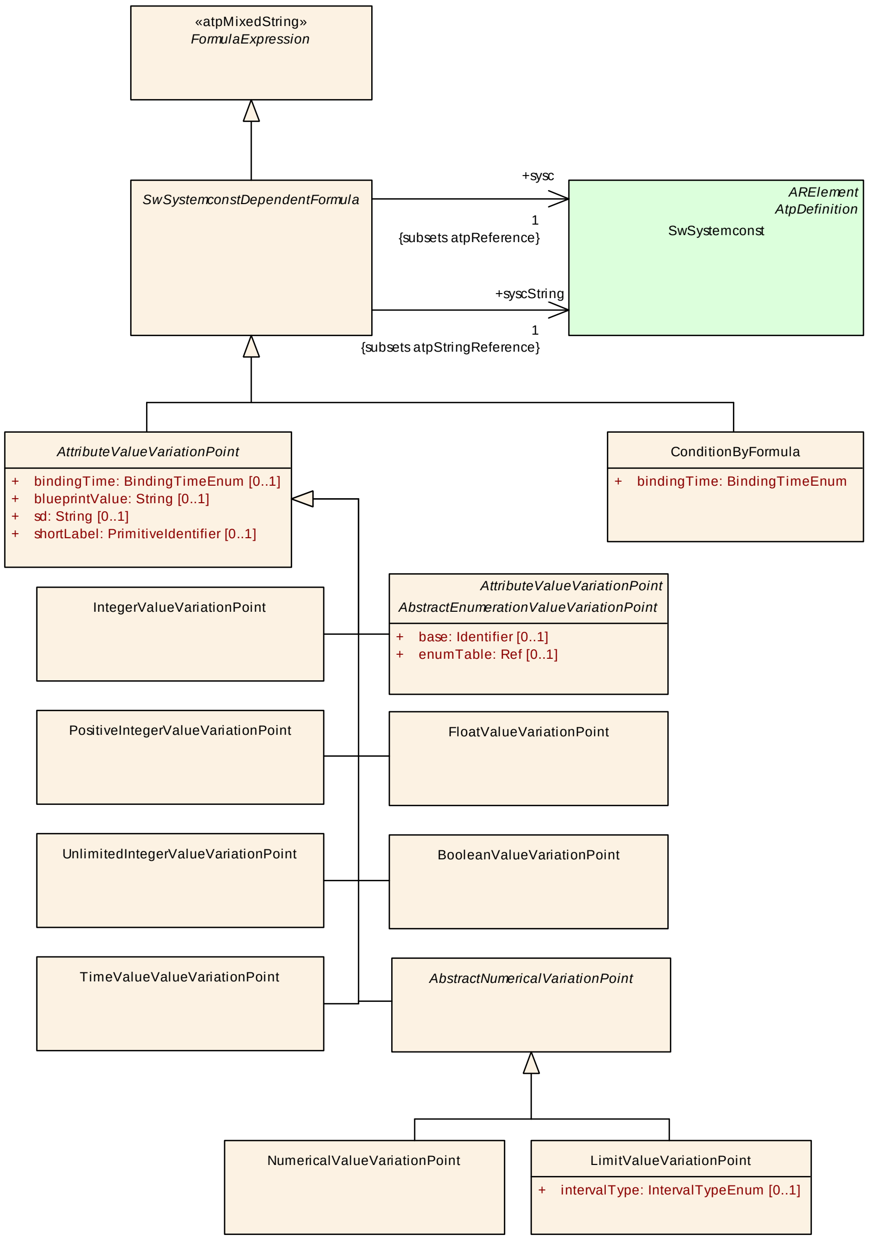

«atpMixedString»
FormulaExpression
SwSystemconstDependentFormula
+sysc
1
{subsets atpReference}
+syscString
1
{subsets atpStringReference}
ARElement
AtpDefinition
SwSystemconst
AttributeValueVariationPoint
+ bindingTime: BindingTimeEnum [0..1]
+ blueprintValue: String [0..1]
+ sd: String [0..1]
+ shortLabel: PrimitiveIdentifier [0..1]
ConditionByFormula
+ bindingTime: BindingTimeEnum
IntegerValueVariationPoint
AttributeValueVariationPoint
AbstractEnumerationValueVariationPoint
+ base: Identifier [0..1]
+ enumTable: Ref [0..1]
PositiveIntegerValueVariationPoint
FloatValueVariationPoint
UnlimitedIntegerValueVariationPoint
BooleanValueVariationPoint
TimeValueValueVariationPoint
AbstractNumericalVariationPoint
NumericalValueVariationPoint
LimitValueVariationPoint
+ intervalType: IntervalTypeEnum [0..1]

Figure 4.8: Attribute Value Variation Point

Class	<<atpMixedString>> ***AttributeValueVariationPoint*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling::AttributeValueVariationPoints			
Note	This class represents the ability to derive the value of the Attribute from a system constant (by Sw SystemconstDependentFormula). It also provides a bindingTime.			
Base	*ARObject*, *FormulaExpression*, *SwSystemconstDependentFormula*			
Subclasses	*AbstractEnumerationValueVariationPoint*, *AbstractNumericalVariationPoint*, BooleanValueVariationPoint, FloatValueVariationPoint, IntegerValueVariationPoint, PositiveIntegerValueVariationPoint, TimeValue ValueVariationPoint, UnlimitedIntegerValueVariationPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	<<atpMixedString>> ***AttributeValueVariationPoint*** (abstract)			
bindingTime	BindingTimeEnum	0..1	attr	This is the binding time in which the attribute value needs to be bound. If this attribute is missing, the attribute is not a variation point. In particular this means that It needs to be a single value according to the type specified in the pure model. It is an error if it is still a formula. **Tags:**xml.attribute=true
blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint. **Tags:**xml.attribute=true
sd	String	0..1	attr	This special data is provided to allow synchronization of Attribute value variation points with variant management systems. The usage is subject of agreement between the involved parties. **Tags:**xml.attribute=true
shortLabel	PrimitiveIdentifier	0..1	attr	This allows to identify the variation point. It is also intended to allow RTE support for CompileTime Variation points. **Tags:**xml.attribute=true

Table 4.12: AttributeValueVariationPoint

4.4 Applicable patterns to define attributes when deriving objects from blueprints

4.4.1 Name Patterns

[TPS_STDT_00003] Applying namePattern ⌈When deriving an element from a blueprint it is often the case that a particular pattern shall be used to determine the shortName respectively the symbol of the object. This use case is supported by the attribute namePattern in Identifier resp. CIdentifier.⌋*(RS_STDT_00004, RS_STDT_00008, RS_STDT_00019, RS_STDT_00021)*

Primitive	**Identifier**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes

Note	An Identifier is a string with a number of constraints on its appearance, satisfying the requirements typical programming languages define for their Identifiers.
	This datatype represents a string, that can be used as a c-Identifier.
	It shall start with a letter, may consist of letters, digits and underscores.
	Tags:
	xml.xsd.customType=IDENTIFIER
	xml.xsd.maxLength=128
	xml.xsd.pattern=[a-zA-Z][a-zA-Z0-9_]*
	xml.xsd.type=string

▽

△

Primitive	**Identifier**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint.
				Tags:
				atp.Status=draft
				xml.attribute=true

namePattern	String	0..1	attr	This attribute represents a pattern which shall be used to define the value of the identifier if the identifier in question is part of a blueprint. For more details refer to TPS_StandardizationTemplate. **Tags:**xml.attribute=true

Table 4.13: Identifier

Primitive	**CIdentifier**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			

Note	This datatype represents a string, that follows the rules of C-identifiers.
	Tags:
	xml.xsd.customType=C-IDENTIFIER
	xml.xsd.pattern=[a-zA-Z_][a-zA-Z0-9_]*
	xml.xsd.type=string			

Attribute	***Type***	***Mult.***	***Kind***	***Note***

blueprintValue	String	1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint.
				Tags:
				atp.Status=draft
				xml.attribute=true

namePattern	String	0..1	attr	This attribute represents a pattern which shall be used to define the value of the identifier if the CIdentifier in question is part of a blueprint. For more details refer to TPS_StandardizationTemplate. **Tags:**xml.attribute=true

Table 4.14: CIdentifier

[TPS_STDT_00055] General Syntax for Name Patterns ⌈The name pattern uses the syntax described in Listing 4.8 defined according to ANTLR [9].⌋*(RS_STDT_00004)*

Listing 4.8: Grammar for name pattern

```
grammar NamePattern;

options { language = Ruby;
          output = AST;}

namePattern
        : (fixedName | placeholder | separator)+ ;

subPattern
  : '('  (fixedName | placeholder | separator )+  ')' ('?' | '*' | '+')? ;

placeholder  : '{'
```

```
                    ('anyName' |
                     'anyNamePart' |
                     'blueprintName' |
                     'capitalizedCallbackName' |
                     'capitalizedMip' |
                     'codePeriode' |
                     'componentName' |
                     'componentTypeName' |
                     'componentPrototypeName' |
                     'ecucValue' '(' ecucName ')' |
                     'index' |
                     'initPolicy' |
                     'keyword' '(' kwClass ')' |
                     'Mip' |
                     'modeName' |
                     'nameSpace' |
                     'portDir' |
                     'typeId'   |
                     subPattern
                    )
                 '}' ;

fixedName :  MyName;

kwClass :    MyName;

separator
  :    Separator ;

pathSeparator
        : PathSeparator ;

ecucName:    ( anyNamePart | pathSeparator)+;

anyNamePart :    MyName (separator MyName)*;

MyName     :   ( 'a'..'z' | ('A'..'Z') | ('0'..'9') | '-' )*;


Separator :  '_' ;

PathSeparator : '/' ;
```

Example 4.1 illustrates valid name patterns. Note that `{blueprintName}` etc. denotes a placeholder.

Example 4.1

```
{blueprintName}_{anyName}

{portDir}_{blueprintName}_{keyword(Qualifier)}_{componentName}_{index}
  --> example for a match:  R_EngN_Max_Dem_3

{componentName}_{ecucValue(item1)}
```

```
h_b_{(a_{index}_b_{componentName}_{(x_{ecucValue(hugo)})*})*}
```

The semantics of the placeholder is defined as follows:

anyName This represents a string which is valid shortName according to Identifier

anyNamePart This represents a string [a-zA-Z0-9_]* which is valid part of a shortName.

Hint: The place holder "anyNamePart" shall not be used at the beginning of a shortName pattern to avoid invalid shortNames.

blueprintName This represents the shortName / shortLabel / symbol of the applied blueprint

capitalizedCallbackName This represents the name of the callback function including module prefix, but written in upper case.

capitalizedMip This represents the capitalized module implementation prefix according to [SWS_BSW_00102]. All characters are converted to uppercase.

codePeriode This represents the period time value and unit. Units are: US micro seconds, MS milli seconds, S second. For example: 100US, 10MS, 1S.

componentName This represents the shortName of the BSW module resp. ASW SwComponentType / ASW component prototype related to the derived object. "Related" mainly could be both, aggregating or referencing.

[TPS_STDT_00036] Placeholder for Module / Component ⌈The placeholder componentName in particular supports multiple derivation of a PortPrototypeBlueprint in the context of different software component types resp. modules.⌋*(RS_STDT_00021)*

componentTypeName This represents the shortName of the dedicated SwComponentType.

componentPrototypeName This represents the shortName of the dedicated SwComponentPrototype.

ecucValue [TPS_STDT_00040] Influence of ECUC ⌈This indicates an influence of the ECU configuration. This placeholder takes an argument which is intended as a keyword reflecting the kind of influence. More details shall be specified in the blueprintCondition where the argument mentioned before can be taken for reference.⌋*(RS_STDT_00002, RS_STDT_00010)*

index This represents a numerical index applicable for example to arrays.

initPolicy This represents the initialization policy of variables according to SectionInitializationPolicyType where the dashes are replaced by underscores, e.g. NO_INIT, CLEARED, POWER_ON_CLEARED, INIT, POWER_ON_INIT.

keyword [TPS_STDT_00004] Abbreviated Name ⌈This represents the abbrName of a keyword acting as a name part of the short name. The eligible keywords can be classified (using the argument kwClass). This classification shall match with one of the classification of the applied keyword.⌋*(RS_STDT_00005, RS_-STDT_00042)*

Mip This represents the module implementation prefix according to [SWS_BSW_-00102].

modeName This represents the shortName of the mode e.g. Dcm_{modeName}ModeEntry

portDir This represents the direction of a port.

[TPS_STDT_00037] Port Direction ⌈The placeholder portDir in particular supports the case that the same blueprint is used for P-Port as well as for an R-Port. The values represented by this placeholder is P for P-Port respectively R for R-Port.⌋*(RS_STDT_00021)*

typeId This represents an indicator based on the type of the object.

4.4.2 Blueprint Formula

[TPS_STDT_00006] Applying Expression Pattern ⌈When deriving an element from a blueprint it is often the case that a particular pattern shall be used to determine the value and or the condition of the object. This use case is supported by the attribute blueprintValue.⌋*(RS_STDT_00019)*

[TPS_STDT_00010] General Syntax for Expression Patterns ⌈The expression pattern uses the syntax of the Formula Language as defined in [TPS_GST_00012].⌋*(RS_-STDT_00019)*

[TPS_STDT_00092] Return values of the BlueprintFormula.ecuc query ⌈The return values are defined in Table 4.15. In the case several EcucContainerValue(s) or EcucParameterValue(s) are asigned to the EcucContainerDef / EcucParameterDef the return value is undefined.⌋*()*

Return values	Description
EcucContainerDef	Ecuc returns the value of the shortName of the EcucContainerValue
EcucBooleanParamDef	Ecuc returns the assigned value of the EcucNumericalParamValue
EcucIntegerParamDef	Ecuc returns the assigned value of the EcucNumericalParamValue

EcucFloatParamDef	Ecuc returns the assigned value of the EcucNumerical-ParamValue
EcucEnumerationParamDef	Ecuc returns the assigned value of the EcucTextual-ParamValue
EcucAbstractStringParamDef	Ecuc returns the assigned value of the EcucTextual-ParamValue
EcucReferenceDef	Ecuc returns the referenced container object qualified by the destination attribute.
EcucChoiceReferenceDef	Ecuc returns the referenced container objects (list) qualified by the destination attributes.
EcucUriReferenceDef	Ecuc returns the referenced container objects (list) qualified by the destinationUri attribute.

Table 4.15: Return values of the BlueprintFormula.ecuc query

[TPS_STDT_00021] Specialization of BlueprintFormula ⌈These specialization(s) express the extension of the Formula Language to provide formalized blueprint-Value:

- ecuc: queries to the values described for ECUC-DEFINITION-ELEMENT. Depending on the ECUC-DEFINITION-ELEMENT a value or a string or an object is the result, see [TPS_GST_00094]
- sysc: queries to the values assigned to SW-SYSTEMCONST
- syscString: indicates that the referenced system constant shall be evaluated as a string according to [TPS_SWCT_01431]
- <VERBATIM>: defines the ability to specify non formula parts
- ->: Reference Operator; a -> b the value of object 'b' as specified in [TPS_GST_-00094] which is pointed to by 'a'

⌋*(RS_STDT_00019)*

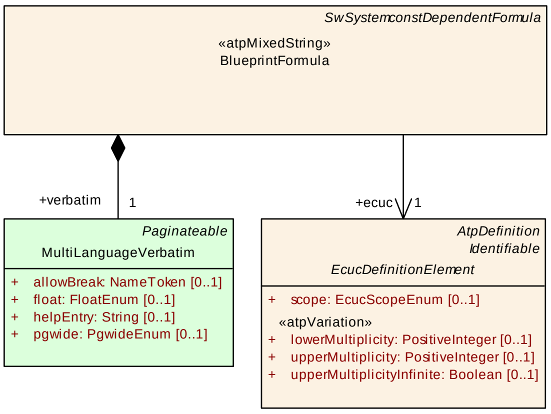

SwSystemconstDependentFormula
«atpMixedString»
BlueprintFormula
+verbatim
1
+ecuc
1
Paginateable
MultiLanguageVerbatim
+ allowBreak: NameToken [0..1]
+ float: FloatEnum [0..1]
+ helpEntry: String [0..1]
+ pgwide: PgwideEnum [0..1]
AtpDefinition
Identifiable
EcucDefinitionElement
+ scope: EcucScopeEnum [0..1]
«atpVariation»
+ lowerMultiplicity: PositiveInteger [0..1]
+ upperMultiplicity: PositiveInteger [0..1]
+ upperMultiplicityInfinite: Boolean [0..1]

Figure 4.9: Blueprint Formula

Listing 4.9 illustrates valid expression patterns. Note that blueprintValue denotes a placeholder.

Listing 4.9: Use of Logical Expression

```
<FORMAL-BLUEPRINT-CONDITION>
  (<ECUC-REF DEST="ECUC-ENUMERATION-PARAM-DEF">/AUTOSAR/EcucDefs/NvM/
     NvMCommon/NvMApiConfigClass</ECUC-REF>== "NVM_API_CONFIG_CLASS_2")
  ||
  (<ECUC-REF DEST="ECUC-ENUMERATION-PARAM-DEF">/AUTOSAR/EcucDefs/NvM/
     NvMCommon/NvMApiConfigClass</ECUC-REF>== "NVM_API_CONFIG_CLASS_3")
  <VERBATIM>
    <L-5 L="EN" xml:space="preserve">only permanent RAM block or explicit
       synchronization is used</L-5>
  </VERBATIM>
</FORMAL-BLUEPRINT-CONDITION>
```

In listing 4.10 the use of the Reference Operator is illustrated. The Reference Operator is inserted as a XML entity.

Listing 4.10: Use of Reference Operator

```
<FORMAL-BLUEPRINT-CONDITION>
  (<ECUC-REF DEST="ECUC-ENUMERATION-PARAM-DEF">/AUTOSAR/EcucDefs/Dcm/
     DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort</ECUC-REF> ==
     USE_DATA_SYNCH_CLIENT_SERVER) &amp;&amp;
  (<ECUC-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/EcucDefs/Dcm/DcmConfigSet/
     DcmDsp/DcmDspData/DcmDspDataInfoRef</ECUC-REF>-&gt;<ECUC-REF DEST="
     ECUC-BOOLEAN-PARAM-DEF">/AUTOSAR/EcucDefs/Dcm/DcmConfigSet/DcmDsp/
     DcmDspDataInfo/DcmDspDataFixedLength</ECUC-REF> == false)
</FORMAL-BLUEPRINT-CONDITION>
```

4.5 Ecu Configuration Parameters and Blueprints

[TPS_STDT_00025] Deriving VSMD from STMD Uses its own Mechanism ⌈Basically the Standard Module Definitions (STMD) specified by AUTOSAR according to [10] could also be considered as blueprints. On the other hand, the relationship between vendor specific module definitions (VSMD) is a very strict one and was there before the general concept of Blueprints was introduced. Therefore for sake of compatibility this relationship is still maintained using `refinedModuleDef`.

Nevertheless for company specific applications there is some support for ECU configuration in Standardization Template.⌋*(RS_STDT_00022, RS_STDT_00010)*

See chapter 5.13 resp. chapter 5.14 for more details.

5 Blueprintables defined in AUTOSAR Meta Model

The following sub chapters specify the particular model elements for which blueprints are supported.

5.1 Blueprinting AccessControl

[TPS_STDT_00062] Blueprinting Elements of AccessControl ⌈AclObjectSet, AclOperation, AclPermission, AclRole can be blueprinted.⌋*(RS_STDT_00032)*

5.2 Blueprinting AliasNameSet

[TPS_STDT_00011] Blueprinting AliasNameSet ⌈AliasNameSet can be blueprinted.⌋*(RS_STDT_00023)*

5.3 Blueprinting ApplicationDataType

[TPS_STDT_00023] Blueprinting ApplicationDataType ⌈ApplicationDataType can be blueprinted.⌋*(RS_STDT_00028, RS_STDT_00029)*

5.4 Blueprinting ARPackage

[TPS_STDT_00013] Blueprinting ARPackage ⌈ARPackage can be blueprinted. Main use case is to support predefined package structures, e.g. those specified in [7].⌋*(RS_STDT_00013, RS_STDT_00030)*

5.5 Blueprinting BswModuleDescription

[TPS_STDT_00027] Blueprinting BswModuleDescription ⌈BswModuleDescription can be blueprinted.⌋*(RS_STDT_00001)*

Blueprints for BswModuleDescription are used in particular to describe dependencies to other modules. Note that in this case all references to other modules and module entries are targeting blueprints of the intended module. These references need to be replaced when deriving objects from the blueprint of BswModuleDescription.

A blueprint of BswModuleDescription shall specify the references to the standard- or blueprint- API elements, in particular

- BswModuleDescription.implementedEntry

- BswModuleDescription.expectedEntry

Nevertheless, it is allowed that derived BswModuleDescription adds further ones of these references.

Furthermore, optional elements like callbacks often come in 0..* multiplicity. In this case, the blueprint should specify one callback reference (to one blueprint BswModuleEntry) and express the open multiplicity in its namePattern respectively in the VariationPoint.blueprintCondition or VariationPoint.formalBlueprintGenerator as illustrated in Figure 5.1.

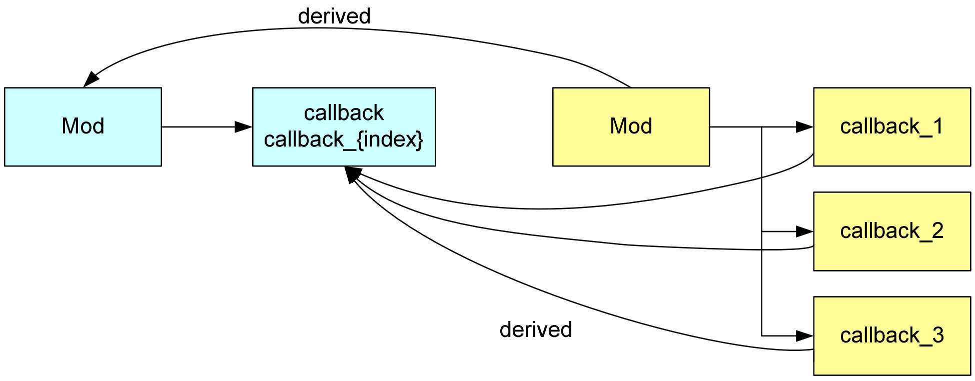


Figure 5.1: Multiply derived Objects

[constr_2563] BswModuleDescription blueprints should not have a BswInternalBehavior ⌈A BswModuleDescription blueprint should not have a BswInternalBehavior since this is a matter of implementation and not subject to standardization. Exceptions might exist in vendor internal applications.⌋*()*

5.6 Blueprinting BswModuleEntry

[TPS_STDT_00014] Blueprinting BswModuleEntry ⌈BswModuleEntry can be blueprinted.⌋*(RS_STDT_00002, RS_STDT_00018, RS_STDT_00029)*

The meta-class BswModuleEntry and its composites (SwServiceArg) contain optional as well as mandatory elements which are never or only sometimes standardized, e.g. executionContext, swServiceImplPolicy, parts of SwServiceArg.swDataDefProps. Nevertheless Standardization Template does not explicitly specify constraint which attributes shall, may or shall not be defined in the blueprint (see also [TPS_STDT_00049]).

5.7 Blueprinting BswEntryRelationshipSet

[TPS_STDT_00090] Blueprinting BswEntryRelationshipSet ⌈BswEntryRelationshipSet can be blueprinted.⌋*(RS_STDT_00002, RS_STDT_00018, RS_STDT_00029)*

[TPS_STDT_00091] Blueprinting `BswEntryRelationshipSet` ⌈The `BswEntryRelationshipSet` describes a collection of `BswEntryRelationship`s. A `BswEntryRelationship` describes a relationship between two `BswModuleEntry`s and the type of relationship. This is typically used to express that a concrete `BswModuleEntry` is derived from an abstract `BswModuleEntry`. In this case the `bswEntryRelationshipType` is set to `derivedFrom`, the `BswEntryRelationship.from` references the abstract `BswModuleEntry` and the `BswEntryRelationship.to` references the concrete `BswModuleEntry`.⌋*(RS_STDT_00002, RS_STDT_00018)*

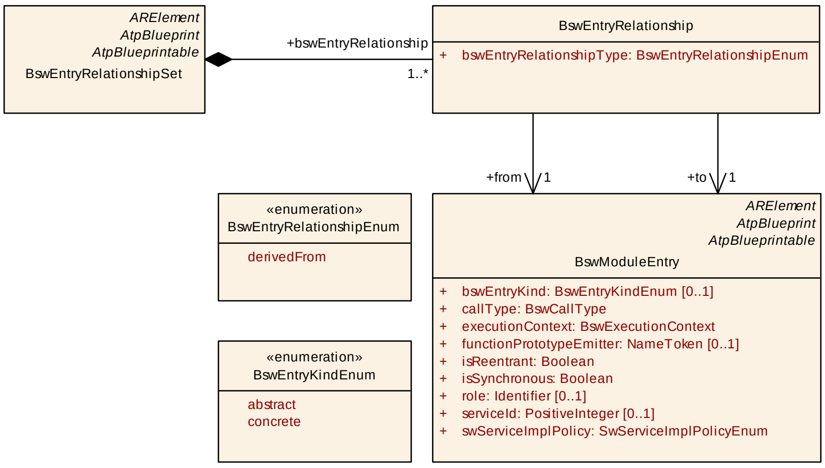


Figure 5.2: BswEntryRelationshipSet

Class	**BswEntryRelationshipSet**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces			
Note	Describes a set of relationships between two BswModuleEntrys. **Tags:**atp.recommendedPackage=BswEntryRelationshipSets			
Base	*ARElement, ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bswEntry Relationship	BswEntryRelationship	1..*	aggr	Relationship between two BswModuleEntrys.

Table 5.1: BswEntryRelationshipSet

Class	**BswEntryRelationship**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces			
Note	Describes a relationship between two BswModuleEntrys and the type of relationship.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bswEntry Relationship Type	BswEntryRelationship Enum	1	attr	Denotes the type of the relationship. **Tags:**xml.sequenceOffset=5

▽

△

Class	**BswEntryRelationship**			
from	BswModuleEntry	1	ref	Type of relationship that refers to the abstract BswModule Entry. Please notice that in this case the bswEntry RelationshipType shall be set to drivedFrom.
to	BswModuleEntry	1	ref	Type of relationship that refers to the concrete Bsw ModuleEntry

Table 5.2: BswEntryRelationship

Enumeration	**BswEntryRelationshipEnum**
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces
Note	
Literal	*Description*
derivedFrom	Describes that the BswModuleEntry referenced as "to" needs to have the same signature as the "abstract" BswModuleEntry referenced as "from". **Tags:**atp.EnumerationLiteralIndex=0

Table 5.3: BswEntryRelationshipEnum

5.8 Blueprinting BuildActionManifest

[TPS_STDT_00063] Blueprinting `BuildActionManifest` ⌈`BuildActionManifest` can be blueprinted. [TPS_STDT_00065] applies such that blueprints of `BuildAction` and `BuildActionEnvironment`s are aggregated in a blueprint of `BuildActionManifest`.⌋*(RS_STDT_00033)*

5.9 Blueprinting CompuMethod

[TPS_STDT_00015] Blueprinting `CompuMethod` ⌈`CompuMethod` can be blueprinted.⌋*(RS_STDT_00029)*

Sometimes it is required to extend a standardized enumeration with vendor specific elements.

For example [SWS_RamTst_00192] states: If vendor specific algorithms were defined the enumeration fields of RamTst_AlgorithmType should be extended accordingly.

[TPS_STDT_00049] Blueprinting Enumerators ⌈Extensions of enumerator values shall be expressed in the blueprint of the related `CompuMethod` by the `VariationPoint` at `CompuScale`.⌋*(RS_STDT_00002, RS_STDT_00029)*

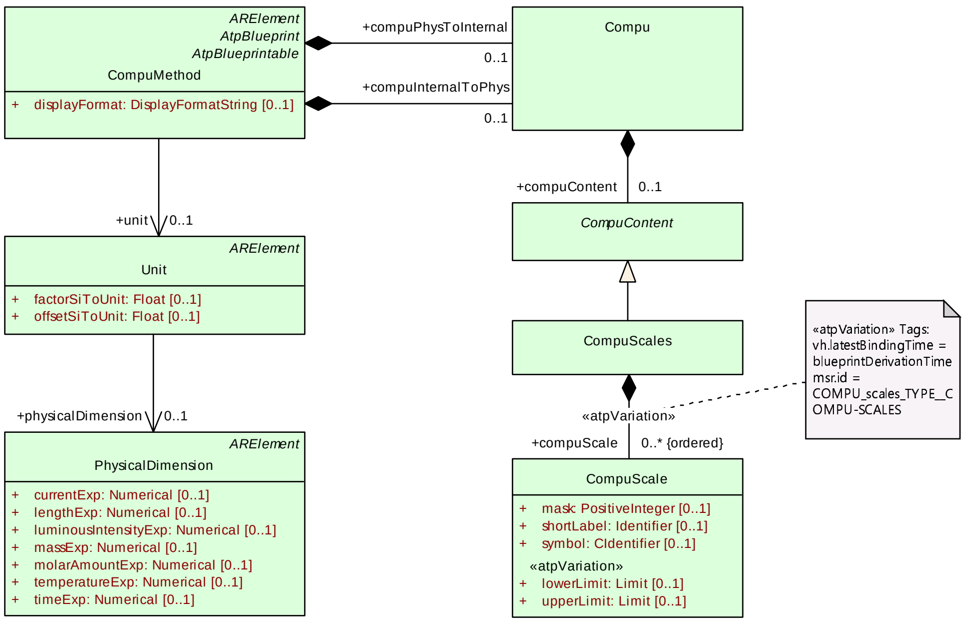

ARElement
AtpBlueprint
AtpBlueprintable
CompuMethod
+ displayFormat: DisplayFormatString [0..1]
+compuPhysToInternal
0..1
+compuInternalToPhys
0..1
Compu
+compuContent
0..1
CompuContent
CompuScales
«atpVariation» Tags:
vh.latestBindingTime =
blueprintDerivationTime
msr.id =
COMPU_scales_TYPE__COMPU-SCALES
«atpVariation»
+compuScale
0..* {ordered}
CompuScale
+ mask: PositiveInteger [0..1]
+ shortLabel: Identifier [0..1]
+ symbol: CIdentifier [0..1]
«atpVariation»
+ lowerLimit: Limit [0..1]
+ upperLimit: Limit [0..1]
+unit
0..1
ARElement
Unit
+ factorSiToUnit: Float [0..1]
+ offsetSiToUnit: Float [0..1]
+physicalDimension
0..1
ARElement
PhysicalDimension
+ currentExp: Numerical [0..1]
+ lengthExp: Numerical [0..1]
+ luminousIntensityExp: Numerical [0..1]
+ massExp: Numerical [0..1]
+ molarAmountExp: Numerical [0..1]
+ temperatureExp: Numerical [0..1]
+ timeExp: Numerical [0..1]

Figure 5.3: A CompuMethod and its attributes define data semantics

5.10 Blueprinting ConsistencyNeeds

[TPS_STDT_00071] Blueprinting ConsistencyNeeds ⌈ConsistencyNeeds can be blueprinted. But as it is not derived from ARElement, all such blueprints are aggregated by ConsistencyNeedsBlueprintSet. This allows to apply [TPS_STDT_00072].⌋*(RS_STDT_00034)*

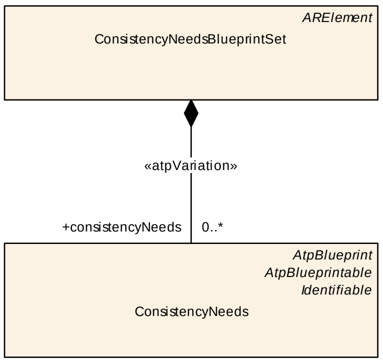


Figure 5.4: Blueprinting ConsistencyNeeds

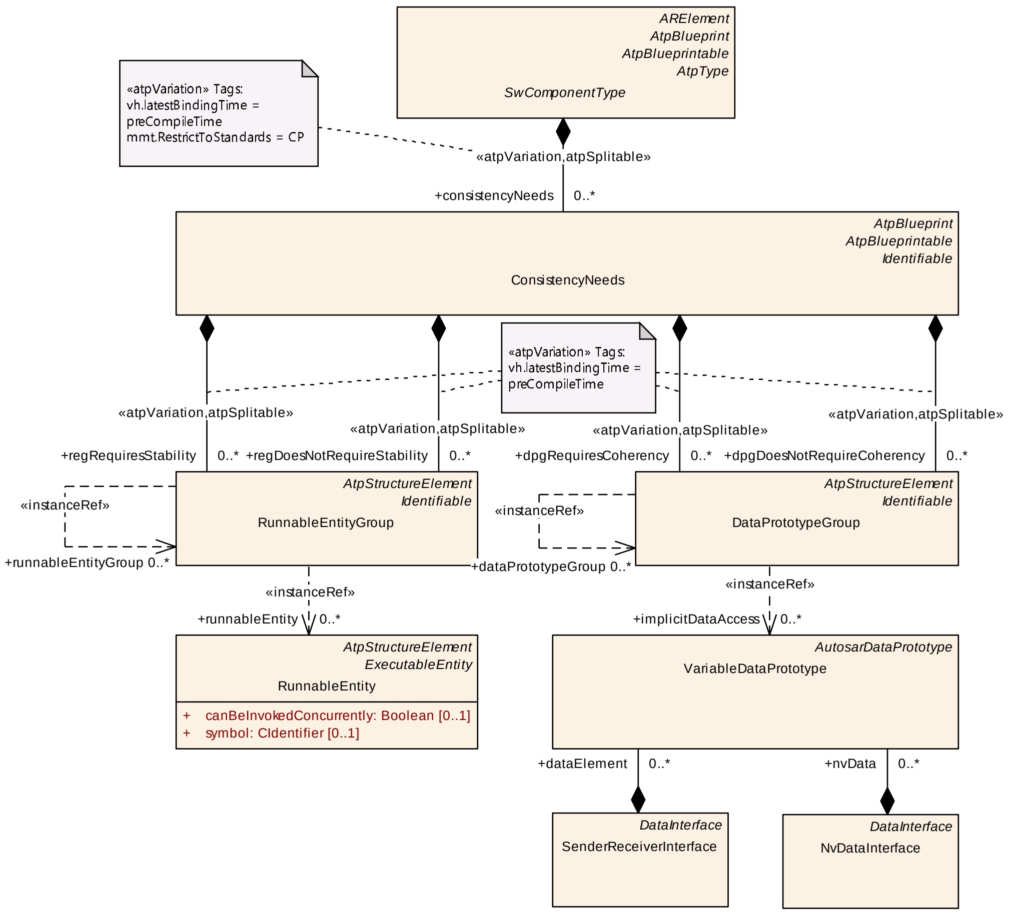


Figure 5.5: ConsistencyNeeds

[TPS_STDT_00073] Early definition of ConsistencyNeeds ⌈Grouping of Data shall be possible before the RunnableEntitys with all the details (data access points) are known. In a top down approach the grouping of DataPrototypes can already be used to design the system in a way that consistency properties are guaranteed and that consistency is not required for unrelated DataPrototypes.

Therefore the DataPrototypeGroup in a ConsistencyNeeds (Blueprint) can reference VariableDataPrototypes of PortInterfaces without any further context information.⌋*(RS_STDT_00034)*

[TPS_STDT_00074] Categorization of Blueprints of ConsistencyNeeds ⌈Since a ConsistencyNeeds(Blueprint) can be designed before the software component is known in all details it is required to denote the purpose of the DataPrototypeGroup and the RunnableEntityGroup of a ConsistencyNeeds(Blueprint). Therefore a set of category values is predefined which supports the "abstract" blueprinting of ConsistencyNeeds.⌋*(RS_STDT_00034)*

[TPS_STDT_00075] Categories for DataPrototypeGroup in a Blueprint of ConsistencyNeeds ⌈

ALL_PROVIDE_DATA_OF_COMPONENT DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes instantiated in provide ports of the software component.

ALL_REQUIRE_DATA_OF_COMPONENT DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes instantiated in require ports of the software component.

ALL_PROVIDE_AND_REQUIRE_DATA_OF_COMPONENT DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes instantiated in provide and require ports of the software component.

ALL_PROVIDE_DATA_OF_RUNNABLE_GROUP DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes where any RunnableEntity in the attached RunnableEntityGroup has a implicit write access to it.

ALL_REQUIRE_DATA_OF_RUNNABLE_GROUP DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes where any RunnableEntity in the attached RunnableEntityGroup has a implicit read access to it.

ALL_PROVIDE_AND_REQUIRE_PORTS_OF_RUNNABLE_GROUP DataPrototypeGroup of the ConsistencyNeeds shall contain all VariableDataPrototypes where any RunnableEntity in the attached RunnableEntityGroup has a implicit write or read access to it.

EXPLICIT_DATA_PROTOTYPE_GROUP DataPrototypeGroup of the ConsistencyNeeds shall contain VariableDataPrototypes according functional requirements

⌋*(RS_STDT_00034)*

[TPS_STDT_00076] Categories for RunnableEntityGroup in a Blueprint of ConsistencyNeeds ⌈

ALL_RUNNABLES_OF_COMPONENT RunnableEntityGroup of the ConsistencyNeeds shall contain all RunnableEntitys of the software component.

ALL_RUNNABLES_WRITING_TO_DATA_PROTOTYP_GROUP RunnableEntityGroup of the ConsistencyNeeds shall contain all RunnableEntitys with a implicit write access to any of the VariableDataPrototypes in the attached DataPrototypeGroup.

ALL_RUNNABLES_READING_FROM_DATA_PROTOTYPE_GROUP RunnableEntityGroup of the ConsistencyNeeds shall contain all RunnableEntitys with a implicit read access to any of the VariableDataPrototypes in the attached DataPrototypeGroup.

ALL_RUNNABLES_WRITING_TO_OR_READING_FROM_DATA_PROTOTYPE_GROUP RunnableEntityGroup of the ConsistencyNeed shall contain all RunnableEntitys with a implicit write or read access to any of the VariableDataPrototypes in the attached DataPrototypeGroup.

EXPLICIT_RUNNABLE_ENTITY_GROUP RunnableEntityGroup of the ConsistencyNeeds shall contain RunnableEntitys according functional requirements

⌋*(RS_STDT_00034)*

5.11 Blueprinting DataConstr

[TPS_STDT_00016] Blueprinting DataConstr ⌈DataConstr can be blueprinted.⌋*(RS_STDT_00029)*

5.12 Blueprinting DataTypeMappingSet

[TPS_STDT_00017] Blueprinting DataTypeMappingSet ⌈DataTypeMappingSet can be blueprinted.⌋*(RS_STDT_00029)*

5.13 Blueprinting EcucDefinitionCollection

[TPS_STDT_00018] Blueprinting EcucDefinitionCollection ⌈EcucDefinitionCollection can be blueprinted.⌋*(RS_STDT_00029)*

5.14 Blueprinting EcucModuleDef

[TPS_STDT_00019] Blueprinting EcucModuleDef ⌈EcucModuleDef can be blueprinted.⌋*(RS_STDT_00029)*

Note that this is intended for company internal use. Please refer to chapter 4.5.

5.15 Blueprinting FlatMap

[TPS_STDT_00035] Blueprinting FlatMap ⌈FlatMap can be blueprinted.⌋*(RS_STDT_00029)*

Usecase for blueprints of FlatMap is given in [11].

5.16 Blueprinting ImplementationDataType

[TPS_STDT_00020] Blueprinting ImplementationDataType ⌈ImplementationDataType can be blueprinted.⌋*(RS_STDT_00029)*

5.17 Blueprinting KeywordSet

[TPS_STDT_00077] Blueprinting KeywordSet ⌈KeywordSet can be blueprinted. The following derivation rules apply:

- No keywords may be removed from or added to the KeywordSet
- The shortName of Keyword shall not be changed or extended
- [TPS_STDT_00085] applies except that longName of Keyword shall not be changed, but it is allowed to add representations in further languages.
- The abbrName shall not be changed or extended(AbbrName)
- The classification of a Keyword shall not be changed but it is allowed to provide additional classification.

⌋*(RS_STDT_00035)*

5.18 Blueprinting LifeCycleStateDefinitionGroups and LifeCycleStates

[TPS_STDT_00043] Blueprinting LifeCycleStateDefinitionGroup ⌈LifeCycleStateDefinitionGroup and LifeCycleState can be blueprinted. [TPS_STDT_00065] applies such that blueprints of LifeCycleState are aggregated in a blueprint of LifeCycleStateDefinitionGroup.⌋*(RS_STDT_00025)*

5.19 Blueprinting ModeDeclarationGroup

[TPS_STDT_00031] Blueprinting ModeDeclarationGroup ⌈ModeDeclarationGroup can be blueprinted.⌋*(RS_STDT_00024)*

5.20 Blueprinting PortPrototype

One of the major activities of the AUTOSAR initiative is the standardization of application interfaces. That is, in terms of the AUTOSAR meta-model the standardization mainly applies to the definition of PortPrototypes for specific purposes.

Due to the structure of the AUTOSAR meta-model it is not possible to merely express a standardized PortPrototype because for good reasons the latter does not exist on its own but is always owned by a SwComponentType.

Therefore, in the past the standardization of "application interfaces" involuntarily also involved the creation of SwComponentTypes. This unnecessary complexity can be overcome by the usage of a PortPrototypeBlueprint.

[TPS_STDT_00007] Blueprinting PortPrototype ⌈PortPrototype can be blueprinted by the specific meta class PortPrototypeBlueprint.⌋*(RS_STDT_-00003)*

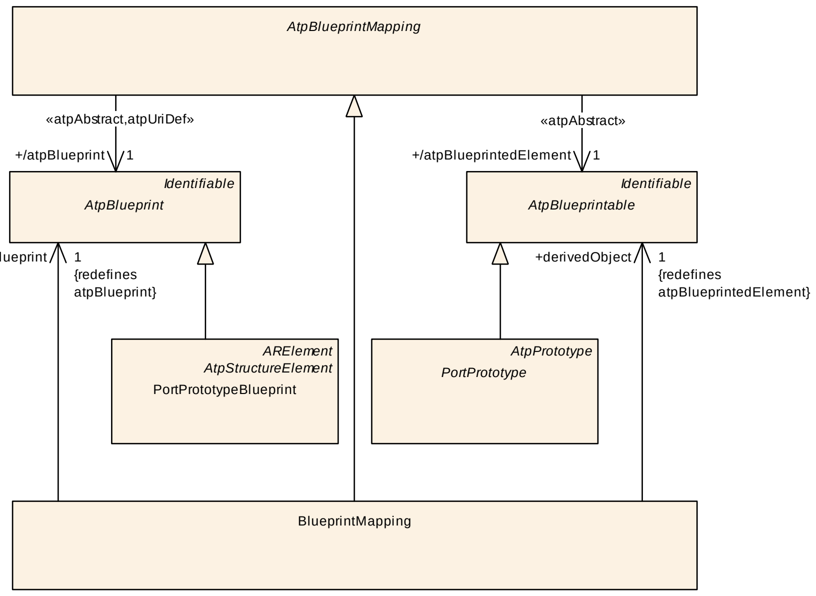


Figure 5.6: Mapping of Port Prototype Blueprints

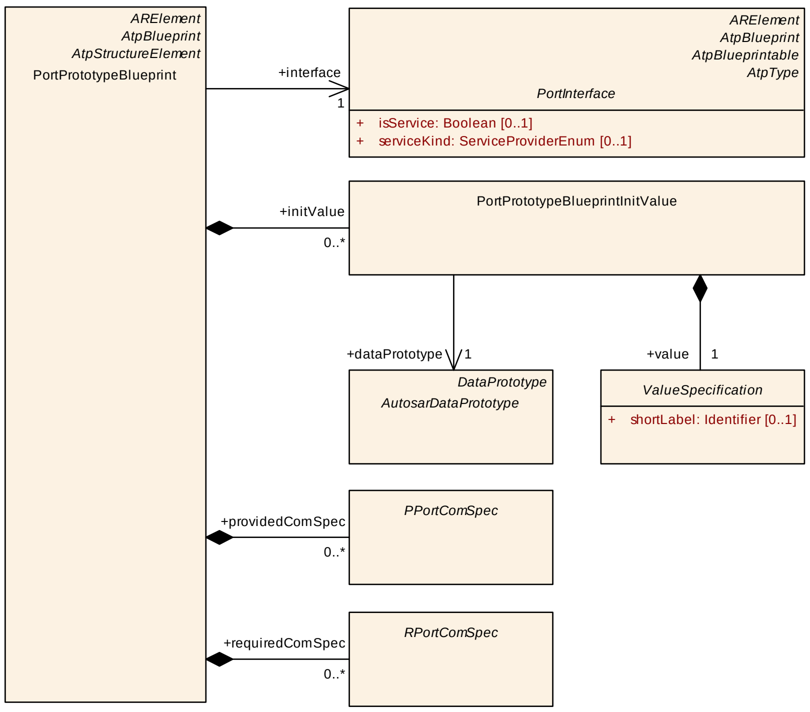


Figure 5.7: Blueprinting Port Prototype

A PortPrototypeBlueprint has the following characteristics:

- It is an ARElement and does therefore not require any element other than an ARPackage as context. It is therefore not necessary to involve “auxiliary” model elements into the definition of a standardized “application interface” for the mere purpose of conforming to the AUTOSAR meta-model.
- It acts as a “blueprint” for the creation of PortPrototypes. That is, probably supported by the used authoring tool, the user picks a specific PortPrototypeBlueprint and creates a PortPrototype out of it. The structure of the created PortPrototype is indistinguishable from a PortPrototype created without taking a PortPrototypeBlueprint as a blueprint. An PortPrototypeBlueprint can be taken as the blueprint for as many PortPrototypes as required.
- It is possible to define additional attributes that are taken over to the created PortPrototype. For example, in some cases the definition of an initial value[1] is part of the definition of a standardized "application interface". Therefore, PortPrototypeBlueprint also supports the definition of an initValue, which needs to be moved to the appropriate ComSpecs.
- It has a reference to the corresponding PortInterface. If the referenced PortInterface is not a blueprint, it can directly be taken over by the PortPrototype created out of the PortPrototypeBlueprint such that the new PortPrototype references the PortInterface. If the referenced PortIn-

[1]AUTOSAR does not standardize init values for application interfaces, but it is supported for vendor internal use.

terface is a blueprint, it is necessary to derive a PortInterface and reference this in the PortPrototype.

- It does not make any assumptions whether the PortPrototype created out of it will be a PPortPrototype or an RPortPrototype.
- It can basically be used for all kinds of PortInterfaces, i.e. it is not constrained to e.g. SenderReceiverInterfaces although this kind of PortInterface will most likely get a significant share of the usage of PortPrototypeBlueprint
- It can only be used for the standardization of “application interfaces”. A PortPrototypeBlueprint does not play any role in the formal description of any SwComponentType or related model artifacts (see also [TPS_STDT_00044]).

[TPS_STDT_00061] PortPrototypeBlueprint can own both RPortComSpecs and PPortComSpecs ⌈PortPrototypeBlueprint can own both RPortComSpecs and PPortComSpecs at the same time. The different ComSpecs are applicable for the derived PPortPrototypes, RPortPrototypes and PRPortPrototypes according the given communication direction. The [constr_1043] (PortInterface vs. ComSpec) in Software Component Template ([1]) is also applicable in this context.⌋*(RS_STDT_-00003)*

[TPS_STDT_00082] Multiple existence of initValue in the context of a PortPrototypeBlueprint ⌈If an initValue exists on the NonqueuedReceiverComSpec or at the NonqueuedSenderComSpec the initValues at PortPrototypeBlueprint shall be ignored.⌋*(RS_STDT_00003)*

In this context [TPS_SWCT_01219] needs also be respected for a valid blueprint.

Listing 5.1: PortPrototypeBlueprint with ProvidedComSpecs

```
<PORT-PROTOTYPE-BLUEPRINT>
  <SHORT-NAME NAME-PATTERN="{anyName}">ALgtOnDoorAtFrntLe</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">Acceleration Longitudinal on Door at Front Left</L-4>
  </LONG-NAME>
  <DESC>
    <L-2 L="EN">Longitudinal high-g acceleration measured in front left
        door of vehicle (locking in driving direction)</L-2>
  </DESC>
  <INTERFACE-REF DEST="SENDER-RECEIVER-INTERFACE">/AUTOSAR/AISpecification
      /PortInterfaces_Blueprint/AExtForOccptPedSfty1</INTERFACE-REF>
  <PROVIDED-COM-SPECS>
    <NONQUEUED-SENDER-COM-SPEC>
      <NETWORK-REPRESENTATION>
        <SW-DATA-DEF-PROPS-VARIANTS>
          <SW-DATA-DEF-PROPS-CONDITIONAL>
            <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
                BaseTypes_Blueprint/uint8</BASE-TYPE-REF>
            <COMPU-METHOD-REF DEST="COMPU-METHOD">/AUTOSAR/Example/
                CompuMethods_Blueprint/AccelerationOnBus</COMPU-METHOD-REF
                >
          </SW-DATA-DEF-PROPS-CONDITIONAL>
        </SW-DATA-DEF-PROPS-VARIANTS>
      </NETWORK-REPRESENTATION>
      <INIT-VALUE>
        <APPLICATION-VALUE-SPECIFICATION>
          <CATEGORY>VALUE</CATEGORY>
          <SW-VALUE-CONT>
            <SW-VALUES-PHYS>
              <V>42</V>
            </SW-VALUES-PHYS>
          </SW-VALUE-CONT>
        </APPLICATION-VALUE-SPECIFICATION>
      </INIT-VALUE>
    </NONQUEUED-SENDER-COM-SPEC>
  </PROVIDED-COM-SPECS>
</PORT-PROTOTYPE-BLUEPRINT>
```

Class	**PortPrototypeBlueprint**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::BlueprintDedicated::Port ProtoypeBlueprint			
Note	This meta-class represents the ability to express a blueprint of a PortPrototype by referring to a particular PortInterface. This blueprint can then be used as a guidance to create particular PortPrototypes which are defined according to this blueprint. By this it is possible to standardize application interfaces without the need to also standardize software-components with PortPrototypes typed by the standardized Port Interfaces. **Tags:**atp.recommendedPackage=PortPrototypeBlueprints			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Collectable Element*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initValue	PortPrototypeBlueprint InitValue	*	aggr	This specifies the init values for the dataElements in the particular PortPrototypeBlueprint.
interface	PortInterface	1	ref	This is the interface for which the blueprint is defined. It may be a blueprint itself or a standardized PortInterface
providedCom Spec	PPortComSpec	*	aggr	Provided communication attributes per interface element (data element or operation).
requiredCom Spec	RPortComSpec	*	aggr	Required communication attributes, one for each interface element.

Table 5.4: PortPrototypeBlueprint

Class	**PortPrototypeBlueprintInitValue**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::BlueprintDedicated::Port ProtoypeBlueprint			
Note	This meta-class represents the ability to express init values in PortPrototypeBlueprints. These init values act as a kind of blueprint from which for example proper ComSpecs can be derived.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototype	AutosarDataPrototype	1	ref	This is the data prototype for which the init value applies **Tags:**xml.sequenceOffset=30

▽

△

Class	**PortPrototypeBlueprintInitValue**			
value	ValueSpecification	1	aggr	This is the init value for the particular data prototype. **Tags:**xml.sequenceOffset=40

Table 5.5: PortPrototypeBlueprintInitValue

As an AUTOSAR model taken for downstream model handling (e.g. generation of an RTE) requires the usage of complete PortInterfaces it is necessary to derive an “actual” PortInterface out of a blueprinted PortInterface defined in the standardization process.

[TPS_STDT_00008] Compatibility of PortPrototype with Blueprint ⌈[constr_2526], [constr_2527], [constr_2528] and [constr_2529] apply for the compatibility of PortPrototypes and PortPrototypeBlueprints⌋*(RS_STDT_00017)*

[constr_2526] PortInterface need to be compatible to the blueprints ⌈PortInterface shall be compatible to their respective blueprints according to the compatibility rules.⌋*()*

[constr_2527] Blueprints shall live in package of a proper category ⌈As explained in detail in the [7], model artifacts (in this case PortPrototypeBlueprint and incompletely specified PortInterfaces) created for the purpose of becoming blueprints shall reside in an ARPackage of category BLUEPRINT.⌋*()*

[constr_2528] PortPrototypes shall not refer to blueprints of a PortInterface ⌈A port PortPrototype shall not reference a PortInterface which lives in a package of category BLUEPRINT.⌋*()*

[constr_2529] PortPrototypeBlueprints and derived PortPrototypes shall reference proper PortInterfaces ⌈A PortPrototypeBlueprint may reference a blueprint of PortInterface. According to [constr_2570], a system description shall not contain blueprints. Therefore the reference to the PortInterface may need to be rewritten when a PortPrototype is derived from the blueprint.

In this case the PortInterface referenced by the derived PortPrototype shall be compatible to the PortInterface (which is a blueprint) referenced by the PortPrototypeBlueprint.

According to [constr_2526] this can be ensured if the PortInterface referenced by the PortPrototypeBlueprint is the blueprint of the PortInterface referenced by the respective PortPrototype.⌋*()*

Note that [constr_2529] is obviously also fulfilled if the PortPrototypeBlueprint and the derived PortPrototype reference a STANDARD PortInterface (which lives in a ARPackage of category "STANDARD").

5.21 Blueprinting PortInterface

[TPS_STDT_00066] Blueprinting PortInterface ⌈PortInterface can be blueprinted.⌋*(RS_STDT_00026)*

[constr_2500] PortInterfaces shall be of same kind ⌈Both objects (PortInterfaces) referenced by a blueprint mapping for port interfaces (represented by BlueprintMapping) shall be of the same kind (e.g. both shall be SenderReceiverInterfaces). In other words both interfaces shall be instances of the same meta class.⌋*()*

Note that [constr_2500] is a special case of [constr_2566].

5.22 Blueprinting PortInterfaceMapping and PortInterfaceMappingSet

[TPS_STDT_00009] Blueprinting PortInterfaceMapping and PortInterfaceMappingSet ⌈PortInterfaceMapping can be blueprinted. [TPS_STDT_00065] applies such that the blueprints of PortInterfaceMapping are aggregated in a blueprint of PortInterfaceMappingSet.⌋*(RS_STDT_00026)*

The intended use cases for blueprinting PortInterfaceMapping are illustrated by figure 5.8. This diagram shows an PortInterface(Blueprint) (*M*), and two ports typed by PortInterface (*S*) respectively by PortInterface(*R*). (*S*) and (*R*) are mapped to the blueprint (*M*) by a PortInterfaceMapping(Blueprint) (*SMMap* and *RMMap*). From this, it is possible to

1. derive PortInterfaceMapping (*SRMap*) between (*S* and *R*) which is then derived from two blueprints (*SMMap* and *RMMap*)
2. propose connectors between two components using the interfaces (*S* and *R*)

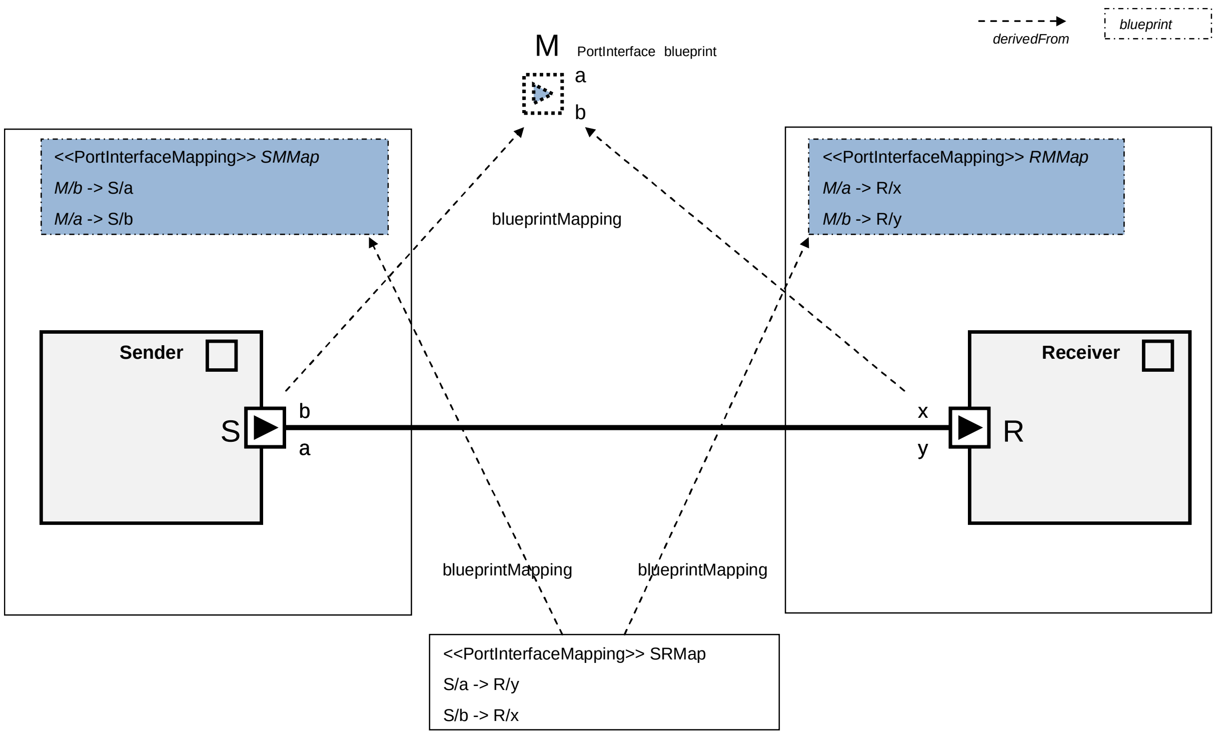


Figure 5.8: Deriving PortInterfaceMapping (1)

The intended derived objects can be determined according to the following steps:

1. find all PortInterface(blueprint)s within the BlueprintMappings of PortInterfaces containing *S* or *R* (in our example it would be *M*)
2. find all PortInterfaceMapping(Blueprint)s containing one of the PortInterface(Blueprint)s from step 1 and one of the PortInterfaces *S* and *R* (in our example it would be *SMMap* and *RMMap*)
3. derive a non blueprint PortInterfaceMapping between *S* and R from the ones found in step 2. Note that all PortInterfaceMappings found so far have a "blueprint reference" and a "non blueprint reference".

   Take one of the PortInterfaceMapping(Blueprint)s from step 2 and replace the "blueprint reference" by the corresponding "non blueprint reference" of the other PortInterfaceMapping(Blueprint)

   ```
   M/b  (blueprint in SMMap) -> S/a  <->  M/b (blueprint in RMmap) -> R/y
   M/a  (blueprint in SMMap) -> S/b  <->  M/a (blueprint in RMmap) -> R/x
   ```

   For example *M/b* would be substituted by *R/y* and *M/a* by *R/x* resulting in the final mapping (*S/a* → *R/y*, *S/b* → *R/x*).

   Same result is achieved if *M/b* would be substituted by *S/a* and *M/a* by *S/b* resulting in the final mapping (*S/a* → *R/y*, *S/b* → *R/x*).

   Implicit mappings (i.e. if data element names between PortInterface and PortInterface(blueprint) are identical then no PortInterfaceMapping(blueprint) is needed) have to be considered too (for example by creating "temporary" mappings).

4. Create BlueprintMappings for the created PortInterfaceMapping (*SRMap*) in step 3 to the involved PortInterfaceMapping(blueprints) (*SMMap* and *RMMap*).

The scenario is shown in the now following listings:

- Listing 5.2 shows the definitions e.g. given by AUTOSAR.
- Listing 5.3 shows the part of LeftCompany
- Listing 5.4 shows the part of RightCompany
- Listing 5.5 shows the part of the integration in a Project

Listing 5.2: Scenario for Blueprints of PortInterfaceMapping (1)

```
<AR-PACKAGE>
  <SHORT-NAME>AUTOSAR</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>PortInterfaces_Blueprint</SHORT-NAME>
      <CATEGORY>BLUEPRINT</CATEGORY>
      <ELEMENTS>
        <SENDER-RECEIVER-INTERFACE>
          <SHORT-NAME NAME-PATTERN="{anyName}">M</SHORT-NAME>
          <DATA-ELEMENTS>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME NAME-PATTERN="{anyName}">a</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME NAME-PATTERN="{anyName}">b</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
          </DATA-ELEMENTS>
        </SENDER-RECEIVER-INTERFACE>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>
```

Listing 5.3 shows that "LeftCompany" has created the PortInterface named *S* derived from the PortInterface(Blueprint) *M*. Thereby the description **how** this takes place is given in the blueprint of an appropriate PortInterfaceMapping named *SMMap*.

Listing 5.3: Scenario for Blueprints of PortInterfaceMapping (2)

```
<AR-PACKAGE>
  <SHORT-NAME>LeftCompany</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>PortInterfaces</SHORT-NAME>
      <ELEMENTS>
        <SENDER-RECEIVER-INTERFACE>
          <SHORT-NAME>S</SHORT-NAME>
          <DATA-ELEMENTS>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME>b</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME>a</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
          </DATA-ELEMENTS>
        </SENDER-RECEIVER-INTERFACE>
      </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
      <SHORT-NAME>BlueprintMappingSets</SHORT-NAME>
      <ELEMENTS>
        <BLUEPRINT-MAPPING-SET>
          <SHORT-NAME>S_isDerivedFrom_M</SHORT-NAME>
          <DESC>
            <L-2 L="EN">This states <E>that</E> S is derived from M</L
               -2>
          </DESC>
          <BLUEPRINT-MAPS>
            <BLUEPRINT-MAPPING>
              <BLUEPRINT-REF DEST="PORT-INTERFACE">/AUTOSAR/
                 PortInterfaces_Blueprint/M</BLUEPRINT-REF>
              <DERIVED-OBJECT-REF DEST="PORT-INTERFACE">/LeftCompany/
                 PortInterfaces/S</DERIVED-OBJECT-REF>
            </BLUEPRINT-MAPPING>
          </BLUEPRINT-MAPS>
        </BLUEPRINT-MAPPING-SET>
      </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
      <SHORT-NAME>PortInterfaceMappingSets_Blueprint</SHORT-NAME>
      <CATEGORY>BLUEPRINT</CATEGORY>
      <ELEMENTS>
        <PORT-INTERFACE-MAPPING-SET>
          <SHORT-NAME NAME-PATTERN="{anyName}">BP</SHORT-NAME>
          <DESC>
            <L-2 L="EN"></L-2>
          </DESC>
          <PORT-INTERFACE-MAPPINGS>
            <VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
              <SHORT-NAME NAME-PATTERN="{anyName}">SMMap</SHORT-NAME>
              <DESC>
                <L-2 L="EN">This defines <E>how</E> S is derived (and
                   therefore mapped to) from M</L-2>
              </DESC>
              <DATA-MAPPINGS>
                <DATA-PROTOTYPE-MAPPING>
                  <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                     PROTOTYPE">/AUTOSAR/PortInterfaces_Blueprint/M/a</
                     FIRST-DATA-PROTOTYPE-REF>
                  <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                     PROTOTYPE">/LeftCompany/PortInterfaces/S/b</SECOND
                     -DATA-PROTOTYPE-REF>
```

```
                  </DATA-PROTOTYPE-MAPPING>
                  <DATA-PROTOTYPE-MAPPING>
                    <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                        PROTOTYPE">/AUTOSAR/PortInterfaces_Blueprint/M/b</
                        FIRST-DATA-PROTOTYPE-REF>
                    <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                        PROTOTYPE">/LeftCompany/PortInterfaces/S/a</SECOND
                        -DATA-PROTOTYPE-REF>
                  </DATA-PROTOTYPE-MAPPING>
                </DATA-MAPPINGS>
              </VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
            </PORT-INTERFACE-MAPPINGS>
          </PORT-INTERFACE-MAPPING-SET>
        </ELEMENTS>
      </AR-PACKAGE>
    </AR-PACKAGES>
  </AR-PACKAGE>
```

Listing 5.4 shows that "RightCompany" has crated the `PortInterface` named *R* derived from the `PortInterface`(Blueprint) *M*. Thereby the description **how** this takes place is given in the blueprint of an appropriate `PortInterfaceMapping` named *RMMap*.

Listing 5.4: Scenario for Blueprints of PortInterfaceMapping (3)

```
<AR-PACKAGE>
  <SHORT-NAME>RightCompany</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>PortInterfaces</SHORT-NAME>
      <ELEMENTS>
        <SENDER-RECEIVER-INTERFACE>
          <SHORT-NAME>R</SHORT-NAME>
          <DATA-ELEMENTS>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME>x</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
            <VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME>y</SHORT-NAME>
            </VARIABLE-DATA-PROTOTYPE>
          </DATA-ELEMENTS>
        </SENDER-RECEIVER-INTERFACE>
      </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
      <SHORT-NAME>BlueprintMappingSets</SHORT-NAME>
      <ELEMENTS>
        <BLUEPRINT-MAPPING-SET>
          <SHORT-NAME>R_isDerivedFrom_M</SHORT-NAME>
          <DESC>
            <L-2 L="EN">This states <E>that</E> S is derived from M</L
                -2>
          </DESC>
          <BLUEPRINT-MAPS>
            <BLUEPRINT-MAPPING>
```

```
                <BLUEPRINT-REF DEST="PORT-INTERFACE">/AUTOSAR/
                   PortInterfaces_Blueprint/M</BLUEPRINT-REF>
                <DERIVED-OBJECT-REF DEST="PORT-INTERFACE">/RightCompany/
                   PortInterfaces/R</DERIVED-OBJECT-REF>
              </BLUEPRINT-MAPPING>
            </BLUEPRINT-MAPS>
          </BLUEPRINT-MAPPING-SET>
        </ELEMENTS>
      </AR-PACKAGE>
      <AR-PACKAGE>
        <SHORT-NAME>PortInterfaceMappingSets_Blueprint</SHORT-NAME>
        <CATEGORY>BLUEPRINT</CATEGORY>
        <ELEMENTS>
          <PORT-INTERFACE-MAPPING-SET>
            <SHORT-NAME NAME-PATTERN="{anyName}">BP</SHORT-NAME>
            <PORT-INTERFACE-MAPPINGS>
              <VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
                <SHORT-NAME NAME-PATTERN="{anyName}">RMMap</SHORT-NAME>
                <DESC>
                  <L-2 L="EN">This defines <E>how</E> R is derived (and
                     therefore mapped to) from M</L-2>
                </DESC>
                <DATA-MAPPINGS>
                  <DATA-PROTOTYPE-MAPPING>
                    <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                       PROTOTYPE">/AUTOSAR/PortInterfaces_Blueprint/M/a</
                       FIRST-DATA-PROTOTYPE-REF>
                    <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                       PROTOTYPE">/RightCompany/PortInterfaces/R/x</
                       SECOND-DATA-PROTOTYPE-REF>
                  </DATA-PROTOTYPE-MAPPING>
                  <DATA-PROTOTYPE-MAPPING>
                    <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                       PROTOTYPE">/AUTOSAR/PortInterfaces_Blueprint/M/b</
                       FIRST-DATA-PROTOTYPE-REF>
                    <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                       PROTOTYPE">/RightCompany/PortInterfaces/R/y</
                       SECOND-DATA-PROTOTYPE-REF>
                  </DATA-PROTOTYPE-MAPPING>
                </DATA-MAPPINGS>
              </VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
            </PORT-INTERFACE-MAPPINGS>
          </PORT-INTERFACE-MAPPING-SET>
        </ELEMENTS>
      </AR-PACKAGE>
    </AR-PACKAGES>
  </AR-PACKAGE>
```

Listing 5.5 shows that "Project" used contributions from "RightCompany" and "LeftCompany". Thereby it maps *S* to *R* in `PortInterfaceMapping` *SRMap*. This is derived from two blueprints (*SMMap* and *SRMap*).

Listing 5.5: Scenario for Blueprints of PortInterfaceMapping (4)

```
<AR-PACKAGE>
  <SHORT-NAME>Project</SHORT-NAME>
```

```
<AR-PACKAGES>
  <AR-PACKAGE>
    <SHORT-NAME>PortInterfaceMappingSets</SHORT-NAME>
    <ELEMENTS>
      <PORT-INTERFACE-MAPPING-SET>
        <SHORT-NAME>Set1</SHORT-NAME>
        <PORT-INTERFACE-MAPPINGS>
          <VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
            <SHORT-NAME>SRMap</SHORT-NAME>
            <DESC>
              <L-2 L="EN">This defines <E>how</E> S is mapped R</L-2>
            </DESC>
            <DATA-MAPPINGS>
              <DATA-PROTOTYPE-MAPPING>
                <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                   PROTOTYPE">/LeftCompany/PortInterfaces/S/b</FIRST-
                   DATA-PROTOTYPE-REF>
                <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                   PROTOTYPE">/RightCompany/PortInterfaces/R/x</
                   SECOND-DATA-PROTOTYPE-REF>
              </DATA-PROTOTYPE-MAPPING>
              <DATA-PROTOTYPE-MAPPING>
                <FIRST-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                   PROTOTYPE">/LeftCompany/PortInterfaces/S/a</FIRST-
                   DATA-PROTOTYPE-REF>
                <SECOND-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                   PROTOTYPE">/RightCompany/PortInterfaces/R/y</
                   SECOND-DATA-PROTOTYPE-REF>
              </DATA-PROTOTYPE-MAPPING>
            </DATA-MAPPINGS>
          </VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
        </PORT-INTERFACE-MAPPINGS>
      </PORT-INTERFACE-MAPPING-SET>
    </ELEMENTS>
  </AR-PACKAGE>
  <AR-PACKAGE>
    <SHORT-NAME>BlueprintMappingSets</SHORT-NAME>
    <ELEMENTS>
      <BLUEPRINT-MAPPING-SET>
        <SHORT-NAME>ProjectMap1</SHORT-NAME>
        <DESC>
          <L-2 L="EN">This states <E>that</E> SRMap is derived from
              SMMap and RMMap simultaneously</L-2>
        </DESC>
        <BLUEPRINT-MAPS>
          <BLUEPRINT-MAPPING>
            <BLUEPRINT-REF DEST="PORT-INTERFACE-MAPPING">/LeftCompany
               /PortInterfaceMappingSets_Blueprint/BP/SMMap</
               BLUEPRINT-REF>
            <DERIVED-OBJECT-REF DEST="PORT-INTERFACE-MAPPING">/
               Project/PortInterfaceMappingSets/Set1/SRMap</DERIVED-
               OBJECT-REF>
          </BLUEPRINT-MAPPING>
          <BLUEPRINT-MAPPING>
```

```
                    <BLUEPRINT-REF DEST="PORT-INTERFACE-MAPPING">/
                        RightCompany/PortInterfaceMappingSets_Blueprint/BP/
                        RMMap</BLUEPRINT-REF>
                    <DERIVED-OBJECT-REF DEST="PORT-INTERFACE-MAPPING">/
                        Project/PortInterfaceMappingSets/Set1/SRMap</DERIVED-
                        OBJECT-REF>
                </BLUEPRINT-MAPPING>
            </BLUEPRINT-MAPS>
        </BLUEPRINT-MAPPING-SET>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>
```

5.23 Blueprinting SwBaseType

[TPS_STDT_00022] Blueprinting SwBaseType ⌈SwBaseType can be blueprinted.⌋ *(RS_STDT_00029)*

5.24 Blueprinting SwComponentType

[TPS_STDT_00024] Blueprinting SwComponentType ⌈SwComponentType can be blueprinted.⌋ *(RS_STDT_00011, RS_STDT_00012)*

[constr_2568] SwComponentTypes shall be of same kind ⌈Both objects (SwComponentTypes) referenced by a blueprint mapping for port interfaces (represented by BlueprintMapping) shall be of the same kind (e.g. both shall be AtomicSwComponentTypes). In other words both components shall be instances of the same meta class.⌋ *()*

Note that [constr_2568] is a special case of [constr_2566].

5.25 Blueprinting SwAddrMethods

[TPS_STDT_00026] Blueprinting SwAddrMethod ⌈SwAddrMethod can be blueprinted.⌋ *(RS_STDT_00029)*

5.26 Blueprinting VfbTiming

[TPS_STDT_00079] Blueprinting VfbTiming ⌈VfbTiming can be blueprinted.⌋ *(RS_STDT_00029)*

One of the essential purposes of blueprinting VFB Timing is enabling one to specify temporal characteristics of interfaces specified in the AUTOSAR Application Interface

Table [12]. In particular, one likes to specify timing constraints imposed on sampling rate, recurrence, age, latency, etc. for such interfaces.

Figure 5.9 shows the basic structure of a VFB Timing Blueprint and how the specified timing elements reference other blueprint elements, specifically the elements PortPrototypeBlueprint and port interface elements which are referenced by the element PortInterface; like variable data prototypes (data elements), client-server operations, mode declarations, and triggers.

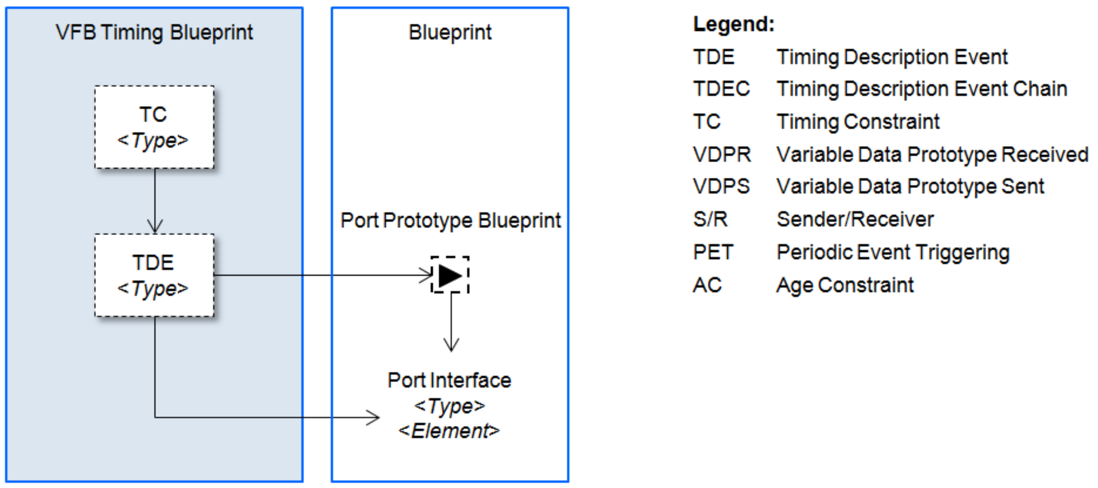

VFB Timing Blueprint
TC
<Type>
TDE
<Type>
Blueprint
Port Prototype Blueprint
Port Interface
<Type>
<Element>
Legend:
TDE Timing Description Event
TDEC Timing Description Event Chain
TC Timing Constraint
VDPR Variable Data Prototype Received
VDPS Variable Data Prototype Sent
S/R Sender/Receiver
PET Periodic Event Triggering
AC Age Constraint

Figure 5.9: VFB Timing Blueprint

A VFB Timing Blueprint consists of timing descriptions events related to the AUTOSAR VFB view, timing description event chains, and timing constraints as defined in the “AUTOSAR Specification of Timing Extensions” [13].

A VFB Timing references the software component it is associated with. In case of a VFB Timing Blueprint this reference need not to be set, but in the derived VFB Timing the VfbTiming.component shall be set properly. In addition, any reference to PortPrototypeBlueprint shall be replaced by the corresponding reference to the PortPrototype.

The following constraints apply to VFB Timing Blueprints and shall be considered when creating such blueprints.

[constr_2589] In VFB Timing Blueprint TDEventVfbPort shall reference PortPrototypeBlueprint ⌈In a VFB Timing Blueprint TDEventVfbPort shall reference PortPrototypeBlueprint. In other words, a VFB Timing Description Event specified in a VFB Timing Blueprint shall always reference a Port Prototype Blueprint.⌋*()*

5.26.1 Example

In this subsection an example for a VFB Timing Blueprint is given. It is based on contents of the AUTOSAR document “Explanation of Application Interfaces of the Powertrain Domain” [14].

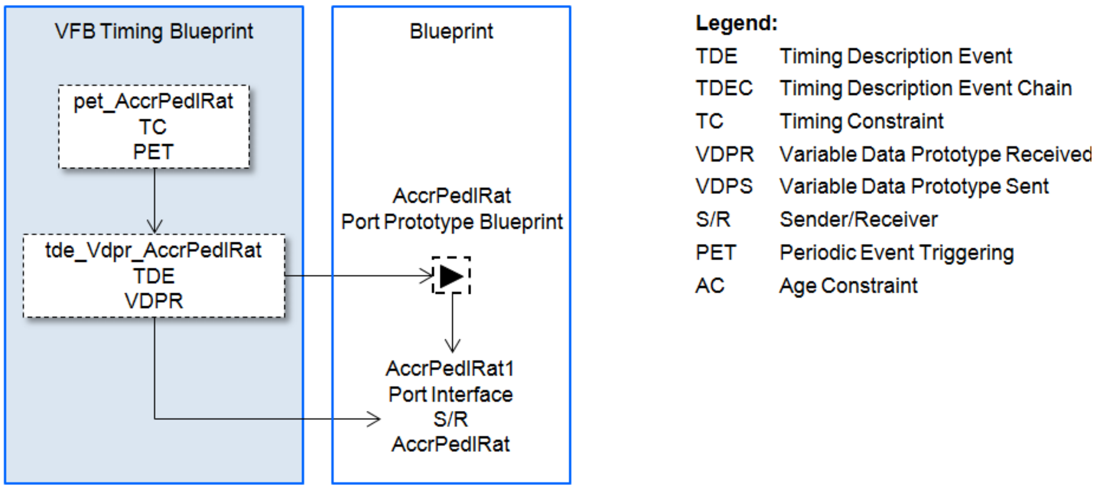


Figure 5.10: VFB Timing Blueprint Simple Example

As sketched in Figure 5.10 a VFB Timing Blueprint is specified. This blueprint consists of a timing description event called “tde_Vdpr_AccrPedlRat” that references the port prototype blueprint called “AccrPedlRat”; and also references the variable data prototype called “AccrPedlRat” of the port interface called “AccrPedlRat1”. The latter is referenced by the mentioned port prototype blueprint, too. In addition, a timing constraint, specifically a periodic event triggering constraint, is imposed on the timing description event. In essence, this timing model specifies that the variable data prototype called “AccrPedlRat” shall be received at a rate given by the periodic event triggering constraint.

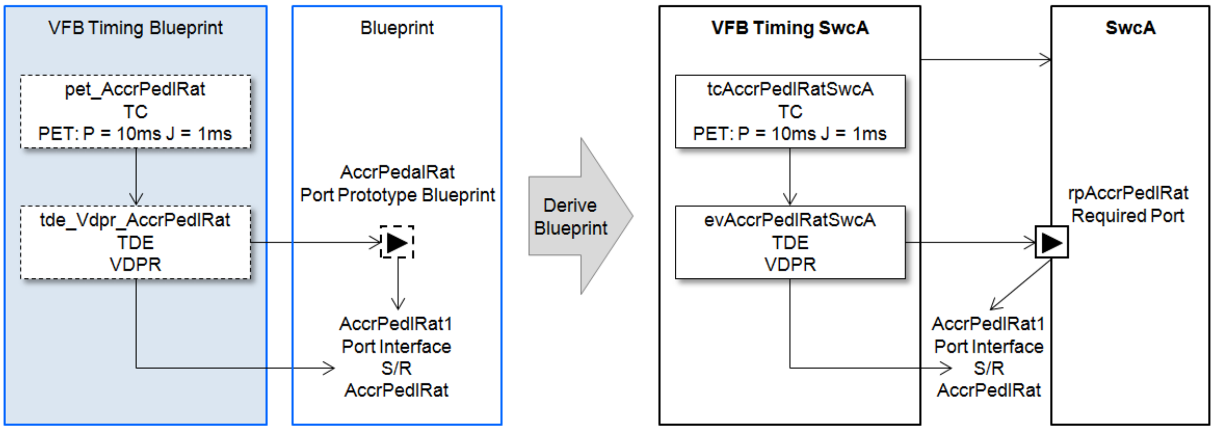
The listing 5.6 provides the corresponding contents of the ARXML file related to the example shown in Figure 5.10, but contains further timing description events and an additional age timing constraint imposed on the receiption of the specific variable data prototype.

Listing 5.6: Example for VFB Timing Blueprint

```
<AR-PACKAGES>
  <AR-PACKAGE>
    <SHORT-NAME NAME-PATTERN="{anyName}">VfbTimingBlueprint</SHORT-NAME>
    <CATEGORY>BLUEPRINT</CATEGORY>
    <ELEMENTS>
      <VFB-TIMING>
        <SHORT-NAME>vfbTiming_AccrPedlRat</SHORT-NAME>
        <TIMING-DESCRIPTIONS>
          <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>tde_Vdps_AccrPedlRat</SHORT-NAME>
            <IS-EXTERNAL>false</IS-EXTERNAL>
            <PORT-PROTOTYPE-BLUEPRINT-REF DEST="PORT-PROTOTYPE-BLUEPRINT"
               >/AUTOSAR/AISpecification/
               PortPrototypeBlueprints_Blueprint/AccrPedlRat</PORT-
               PROTOTYPE-BLUEPRINT-REF>
            <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/AUTOSAR/
               AISpecification/PortInterfaces_Blueprint/AccrPedlRat1/
               AccrPedlRat</DATA-ELEMENT-REF>
            <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
               PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
          </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
          <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>tde_Vdpr_AccrPedlRat</SHORT-NAME>
            <IS-EXTERNAL>false</IS-EXTERNAL>
            <PORT-PROTOTYPE-BLUEPRINT-REF DEST="PORT-PROTOTYPE-BLUEPRINT"
               >/AUTOSAR/AISpecification/
               PortPrototypeBlueprints_Blueprint/AccrPedlRat</PORT-
               PROTOTYPE-BLUEPRINT-REF>
            <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/AUTOSAR/
               AISpecification/PortInterfaces_Blueprint/AccrPedlRat1/
               AccrPedlRat</DATA-ELEMENT-REF>
            <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
               PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
          </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
          <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>tde_Vdp_AccrPedlRat</SHORT-NAME>
            <IS-EXTERNAL>false</IS-EXTERNAL>
            <PORT-PROTOTYPE-BLUEPRINT-REF DEST="PORT-PROTOTYPE-BLUEPRINT"
               >/AUTOSAR/AISpecification/
               PortPrototypeBlueprints_Blueprint/AccrPedlRat</PORT-
               PROTOTYPE-BLUEPRINT-REF>
            <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/AUTOSAR/
               AISpecification/PortInterfaces_Blueprint/AccrPedlRat1/
               AccrPedlRat</DATA-ELEMENT-REF>
          </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
        </TIMING-DESCRIPTIONS>
        <TIMING-REQUIREMENTS>
          <PERIODIC-EVENT-TRIGGERING>
            <SHORT-NAME>pet_AccrPedlRat</SHORT-NAME>
            <EVENT-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/
               VfbTimingBlueprint/vfbTiming_AccrPedlRat/
               tde_Vdp_AccrPedlRat</EVENT-REF>
            <JITTER>
              <CSE-CODE>0</CSE-CODE>
              <CSE-CODE-FACTOR>1</CSE-CODE-FACTOR>
            </JITTER>
            <PERIOD>
              <CSE-CODE>0</CSE-CODE>
              <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
            </PERIOD>
          </PERIODIC-EVENT-TRIGGERING>
          <AGE-CONSTRAINT>
            <SHORT-NAME>ac_AccrPedlRat</SHORT-NAME>
            <MAXIMUM>
              <CSE-CODE>0</CSE-CODE>
              <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
            </MAXIMUM>
            <SCOPE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/
               VfbTimingBlueprint/vfbTiming_AccrPedlRat/
               tde_Vdpr_AccrPedlRat</SCOPE-REF>
          </AGE-CONSTRAINT>
        </TIMING-REQUIREMENTS>
      </VFB-TIMING>
    </ELEMENTS>
```

```
    </AR-PACKAGE>
  </AR-PACKAGES>
```

Figure 5.11 shows the VFB Timing Blueprint and the derived VFB Timing for a specific software component called “SW-C_A”.



Legend:

TDE	Timing Description Event
TDEC	Timing Description Event Chain
TC	Timing Constraint
VDPR	Variable Data Prototype Received
VDPS	Variable Data Prototype Sent
S/R	Sender/Receiver
PET	Periodic Event Triggering
AC	Age Constraint
P	Period
J	Jitter

Figure 5.11: Deriving a VFB Timing Blueprint

5.27 Blueprinting ClientServerInterfaceToBswModuleEntryBlueprintMapping

[TPS_STDT_00083] Blueprinting ClientServerInterfaceToBswModuleEntryBlueprintMapping ⌈ClientServerInterfaceToBswModuleEntryBlueprintMapping can be blueprinted.⌋*(RS_STDT_00029)*

[TPS_STDT_00084] ClientServerOperationBlueprintMapping predetermines the implementation of an ClientServerOperation ⌈A ClientServerOperationBlueprintMapping expresses the intended implementation of a ClientServerOperation by a specific BswModuleEntry under consideration of the expected usage of PortDefinedArgumentValues.⌋*(RS_STDT_00029)*

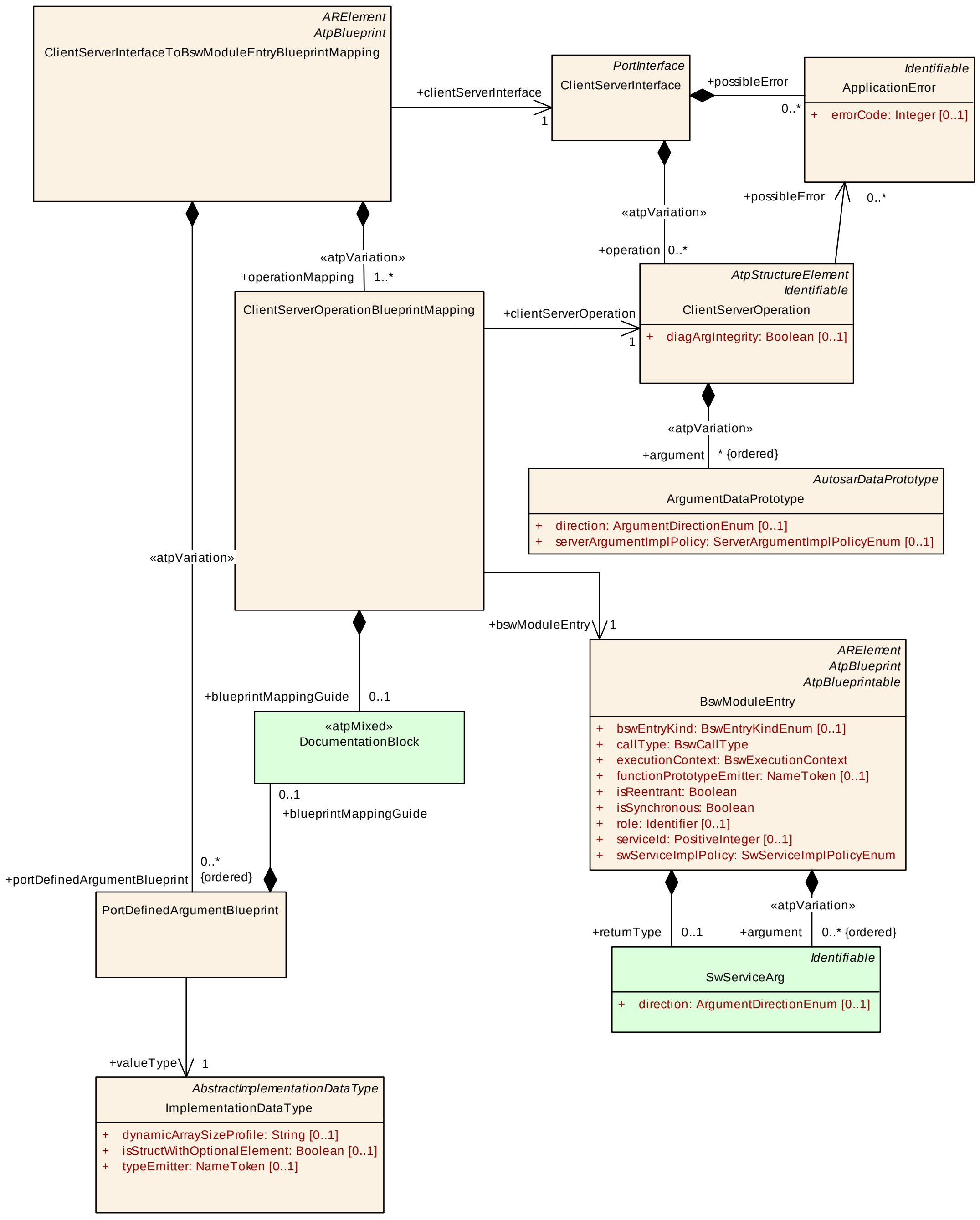

ARElement
AtpBlueprint
ClientServerInterfaceToBswModuleEntryBlueprintMapping
+clientServerInterface
1
PortInterface
ClientServerInterface
+possibleError
0..*
Identifiable
ApplicationError
+ errorCode: Integer [0..1]
+possibleError
0..*
«atpVariation»
+operation
0..*
AtpStructureElement
Identifiable
ClientServerOperation
+ diagArgIntegrity: Boolean [0..1]
«atpVariation»
+operationMapping
1..*
ClientServerOperationBlueprintMapping
+clientServerOperation
1
«atpVariation»
+argument
* {ordered}
AutosarDataPrototype
ArgumentDataPrototype
+ direction: ArgumentDirectionEnum [0..1]
+ serverArgumentImplPolicy: ServerArgumentImplPolicyEnum [0..1]
«atpVariation»
+bswModuleEntry
1
ARElement
AtpBlueprint
AtpBlueprintable
BswModuleEntry
+ bswEntryKind: BswEntryKindEnum [0..1]
+ callType: BswCallType
+ executionContext: BswExecutionContext
+ functionPrototypeEmitter: NameToken [0..1]
+ isReentrant: Boolean
+ isSynchronous: Boolean
+ role: Identifier [0..1]
+ serviceId: PositiveInteger [0..1]
+ swServiceImplPolicy: SwServiceImplPolicyEnum
+blueprintMappingGuide
0..1
«atpMixed»
DocumentationBlock
0..1
+blueprintMappingGuide
0..*
{ordered}
+portDefinedArgumentBlueprint
PortDefinedArgumentBlueprint
«atpVariation»
+returnType
0..1
+argument
0..* {ordered}
Identifiable
SwServiceArg
+ direction: ArgumentDirectionEnum [0..1]
+valueType
1
AbstractImplementationDataType
ImplementationDataType
+ dynamicArraySizeProfile: String [0..1]
+ isStructWithOptionalElement: Boolean [0..1]
+ typeEmitter: NameToken [0..1]

Figure 5.12: Client Server Operation Blueprint Mapping

Class	**ClientServerOperationBlueprintMapping**
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::ClientServerInterfaceToBswModuleEntryMapping
Note	This class describes a specific mapping between a ClientServerOperation in a ClientServerInterface blueprint and a BswModuleEntry blueprint.
Base	*ARObject*

▽

△

Class	ClientServerOperationBlueprintMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
blueprint MappingGuide	DocumentationBlock	0..1	aggr	This attribute offers the possibility to provide additional information with respect to the mapping.
bswModule Entry	BswModuleEntry	1	ref	The referenced BswModuleEntry represents the Bsw ModuleEntry the mapping is dedicated to.
clientServer Operation	ClientServerOperation	1	ref	The referenced ClientServerOperation represents the client server operation the mapping is dedicated to.

Table 5.6: ClientServerOperationBlueprintMapping

The ClientServerOperationBlueprintMapping can be used to ensure and/or track the compatibility of BswModuleEntrys which are supposed to implement ClientServerOperations. It can already be defined in an early phase of the methodology when interfaces are defined. Thereby the ClientServerOperationBlueprintMapping can already be defined without all implementation details of the later required SwComponentType, SwcInternalBehavior, BswModuleDescription, BswInternalBehavior and SwcBswMapping.

Please note that the ClientServerInterfaceToBswModuleEntryBlueprintMapping has no direct impact to the later generated RTE. The setup of the RTE is solely determined by the derived objects of ClientServerOperation, BswModuleEntry and the completed software component descriptions and basic software module descriptions respectively.

Such a mapping enables the formal check whether the number of arguments and the data types of arguments of the operation + additional PortDefinedArgumentValues matches the signature of the BswModuleEntry.

[constr_2597] ClientServerOperationBlueprintMapping constraints number of arguments ⌈The number of arguments of the BswModuleEntry referenced by a bswModuleEntry shall be identical to the number of portDefinedArgumentBlueprints of the owning ClientServerInterfaceToBswModuleEntryBlueprintMapping plus the number of ArgumentDataPrototypes aggregated in the role argument of the clientServerOperation⌋*()*

[constr_2598] ClientServerOperationBlueprintMapping constraints the types of arguments ⌈The arguments in the ordered lists bswModuleEntry and the matching arguments in the set union of the ordered lists portDefinedArgumentBlueprint plus clientServerOperation shall result in the identical C data type definitions.⌋*()*

6 Keywords

[TPS_STDT_00012] Defining Keywords ⌈The meta-class KeywordSet can be used to define sets of Keywords. The purpose of a Keyword is to contribute parts of names for AUTOSAR model elements.⌋*(RS_STDT_00005, RS_STDT_00008, RS_-STDT_00042)*

Keywords are referenced to be part of name pattern as specified in Chapter 4.4.1.

As an example, the shortName “CmftMngt” is composed out of two Keywords with the abbrName “Cmft” and “Mngt”.

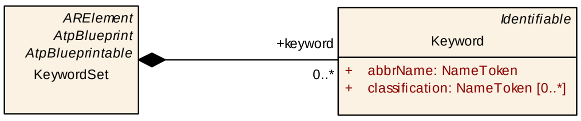


Figure 6.1: Keyword and KeywordSet

[TPS_STDT_00069] Attributes of Keyword ⌈The meta-class Keyword is derived from Identifiable. The attributes of Identifiable shall be applied for Keyword as follows.

shortName represents the unique name of the keyword. In the example above it would be “Cmft”. Note that this is used only for identifying the keyword. The contributed name part is taken from abbrName.

longName represents the long form of the keyword, typically its an unabbreviated technical term. In the example above it would be “Comfort”.

desc represents the definition of the keyword in terms of a verbal description allowing to identify whether the keyword applies for a specific case. In the example above the description would be “This keyword is used to express something as comfortable or convenient”.

introduction represents a verbal description of a use case. This can be used for additional explanations or examples.

⌋*(RS_STDT_00005, RS_STDT_00042)*

Class	**KeywordSet**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::Keyword			
Note	This meta–class represents the ability to collect a set of predefined keywords. **Tags:**atp.recommendedPackage=KeywordSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keyword	Keyword	*	aggr	This is one particular keyword in the keyword set.

Table 6.1: KeywordSet

Class	**Keyword**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::Keyword			
Note	This meta-class represents the ability to predefine keywords which may subsequently be used to construct names following a given naming convention, e.g. the AUTOSAR naming conventions. Note that such names is not only shortName. It could be symbol, or even longName. Application of keywords is not limited to particular names.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
abbrName	NameToken	1	attr	This attribute specifies an abbreviated name of a keyword. This abbreviation may e.g. be used for constructing valid shortNames according to the AUTOSAR naming conventions. Unlike shortName, it may contain any name token. E.g. it may consist of digits only.
classification	NameToken	*	attr	This attribute allows to attach classification to the Keyword such as MEAN, ACTION, CONDITION, INDEX, PREPOSITION

Table 6.2: Keyword

[TPS_STDT_00070] Classification of Keywords ⌈The attribute `classification` depends on the applied naming convention.⌋*(RS_STDT_00005, RS_STDT_00042)*

For example, the values could be according to table 2 of [15] such as `Action-PhysicalType`, `Condition-Qualifier`, `Index`, `Mean-Environment-Device`, `Preposition`.

Listing 6.1 illustrates an example how to use `Keyword`. More elaborate usage can be seen in [3].

Listing 6.1: example for keywords

```
<AR-PACKAGE>
  <SHORT-NAME>AUTOSAR</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>AISpecification</SHORT-NAME>
      <AR-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>KeywordSets</SHORT-NAME>
          <ELEMENTS>
            <KEYWORD-SET>
              <SHORT-NAME>KeywordListComfort</SHORT-NAME>
              <KEYWORDS>
                <KEYWORD>
                  <SHORT-NAME>Cmft</SHORT-NAME>
                  <LONG-NAME>
                    <L-4 L="EN">Comfort</L-4>
                  </LONG-NAME>
                  <DESC>
                    <L-2 L="EN">comfort. this keyword is used to
                       express something as comfortable or convenient</
                       L-2>
                  </DESC>
                  <ABBR-NAME>Cmft</ABBR-NAME>
```

```
                                <CLASSIFICATIONS>
                                  <CLASSIFICATION>Condition-Qualifier</CLASSIFICATION
                                      >
                                </CLASSIFICATIONS>
                              </KEYWORD>
                            </KEYWORDS>
                          </KEYWORD-SET>
                        </ELEMENTS>
                      </AR-PACKAGE>
                    </AR-PACKAGES>
                  </AR-PACKAGE>
                </AR-PACKAGES>
              </AR-PACKAGE>
```

[TPS_STDT_00068] Expressing "stem"-Relation of Keywords ⌈There are keywords which basically stem from the same root. This relationship is expressed by an `Collection` where the `elementRole` is named `DECLINATION_OF`. The root is denoted `sourceElement`. The declinations are denoted in `element`. The root is not a declination of itself, and therefore is not mentioned as an `element` again.⌋*(RS_STDT_00005, RS_STDT_00042)*

As an example for [TPS_STDT_00068] the keywords `Drvr, Drvg` stem from `Drv`[1]. This is delivered according to the example in Listing 6.2

Listing 6.2: Example for Stem Relation of Keywords

```
<COLLECTION>
  <SHORT-NAME>Drv_declinations</SHORT-NAME>
  <CATEGORY>RELATION</CATEGORY>
  <ELEMENT-ROLE>DECLINATION_OF</ELEMENT-ROLE>
  <ELEMENT-REFS>
    <ELEMENT-REF BASE="KW" DEST="KEYWORD">KeywordList/Drvr</ELEMENT-REF>
    <ELEMENT-REF BASE="KW" DEST="KEYWORD">KeywordList/Drvg</ELEMENT-REF>
  </ELEMENT-REFS>
  <SOURCE-ELEMENT-REFS>
    <SOURCE-ELEMENT-REF BASE="KW" DEST="KEYWORD">KeywordList/Drv</SOURCE-
        ELEMENT-REF>
  </SOURCE-ELEMENT-REFS>
</COLLECTION>
```

7 Deriving from AUTOSAR-provided Blueprints

Model elements provided by AUTOSAR are mainly provided as blueprints. This holds true in particular for the Application Interfaces [12] but also for the Software Specifications of the BSW layer. These AUTOSAR delivered model elements follow the package structure specified in [TPS_GST_00080].

[1]Note that `Drv` is not an `element` of this `Collection` since it is not a declination of itself.

Figure 7.1 illustrates the methodology to define data types for BSW module. The `BSW Standard Package` contains blueprints. In the above scenario, [TPS_STDT_00067] shall be followed but of course also holds true for the data types of other modules.

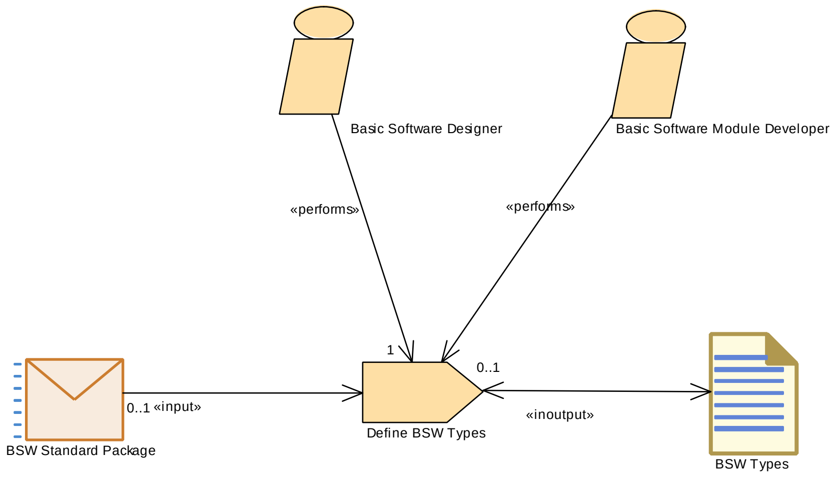


Figure 7.1: Define Bsw Types

[TPS_STDT_00067] Standardized Path for Standardized Elements ⌈Objects derived from standardized blueprints, shall follow a package path as specified in [TPS_GST_00083]. That is, providers of Software components can rely that all AUTOSAR defined model elements can be accessed through a predicable path.⌋*(RS_STDT_00001, RS_STDT_00002, RS_STDT_00014, RS_STDT_00028, RS_STDT_00030)*

For example the Platformtypes [16] blueprinted in

```
/AUTOSAR/Platform/ImplementationDatatypes_Blueprint/uint8
```

shall be implemented in (and therefore safely be accessible through)

```
/AUTOSAR_Platform/ImplementationDatatypes/uint8
```

8 Description of Data Exchange Points

8.1 Overview

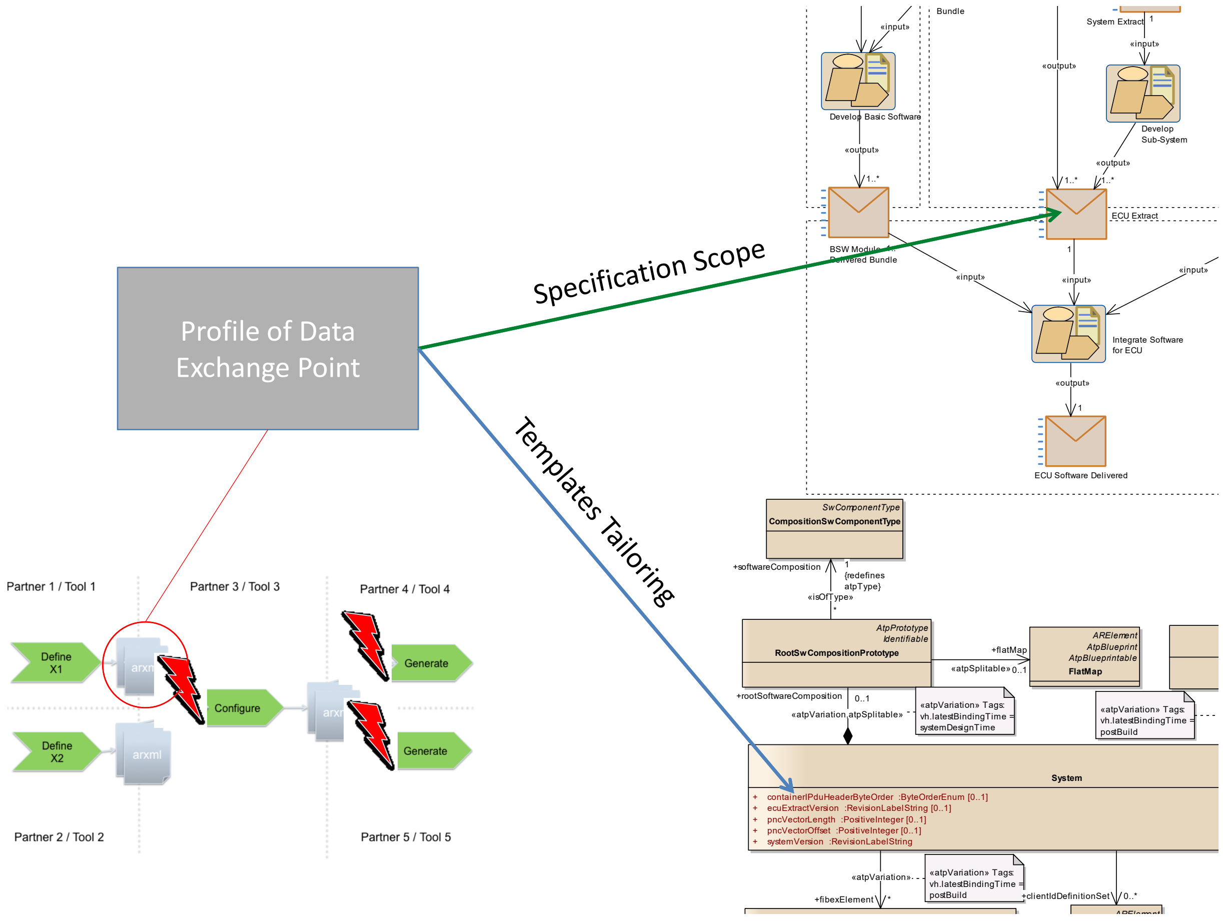


Figure 8.1: Overview of Description of Data Exchange Point

[TPS_STDT_00100] Motivation of Description of Data Exchange Points ⌈
`Profiles of Data Exchange Point`s intend to improve the interoperability between tools by describing which data is expected for a given activity or task in the methodology.⌋*(RS_STDT_00102, RS_STDT_00103)* (see figure 8.1)

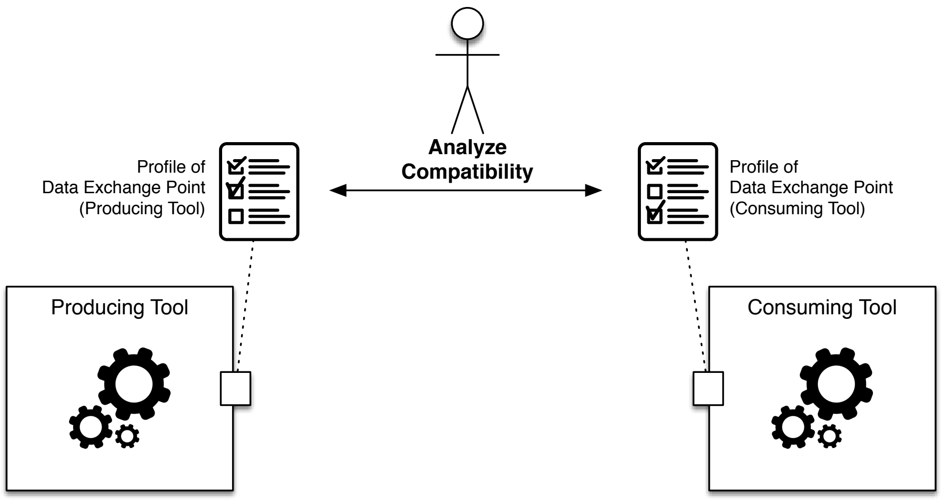


Figure 8.2: Documentation and Analysis of Data Interface of Tools

[TPS_STDT_00115] Analysis of Tool Compatibility ⌈`Profile of Data Exchange Point`s enable structured documentation of the subset of the AUTOSAR standard that is supported or explicitly not supported by a tool.
In other words: the profile describes the data interface of the tool. The availability of profiles enables initial checks of compatibility of tools before actual AUTOSAR models are available (e.g. due to IP issues, new features that are not yet implemented, ...). Commonalities usually show low risk of interoperability issues. Differences or undefined information show potential high risk of interoperability issues.
The formalized description of Data Exchange Points additionally enables tool support for finding locations with high risk of interoperability issues. However, the interpretation of the results requires engineering know how and interaction between tool vendors and users.⌋*(RS_STDT_00117, RS_STDT_00118)* (see figure 8.2)

[TPS_STDT_00116] Limitation of Analysis of `Profile of Data Exchange Point`s ⌈
The analysis of the compatibility of two or more `Profile of Data Exchange Point`s has the goal to identify potential interoperability issues. The analysis of profiles can help identifying some issues. However, the analysis cannot guarantee the absence of interoperability issues. This analysis does not replace other interoperability check mechanisms such as the creation and processing of example AUTOSAR models that make use of the intended features (reference models).⌋*(RS_STDT_00117, RS_STDT_00118)*

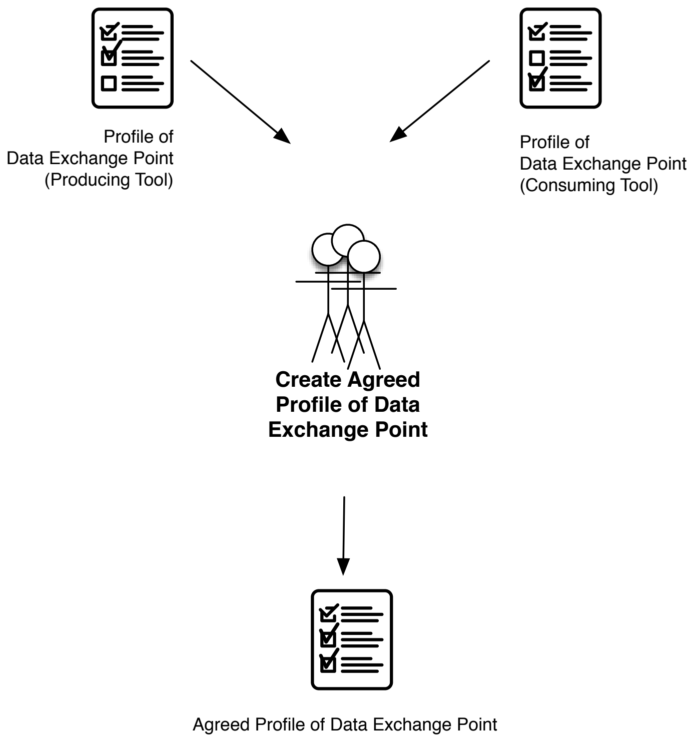


Figure 8.3: Creation of an agreed Profile of Data Exchange Point

[TPS_STDT_00117] Agreed Profile of Data Exchange Point ⌈
The result of the analysis and negotiation of `Profile of Data Exchange Point`s can be documented as an `Agreed Profile of Data Exchange Point`. Usage scenarios for the `Agreed Profile for Data Exchange` include:

- Validation of the AUTOSAR models that are created by the producing tool with respect to compliance with the agreed contract. This validation can, for instance, be used as a quality gate before the actual AUTOSAR model is passed to the consuming tool.
- A `Profile of Data Exchange Point` can be used as a specification for intended future functionality. Tool vendors can analyze the `Agreed Profile for Data Exchange` in order to identify features that are not yet implemented.

⌋*(RS_STDT_00121)* (see figure 8.3)

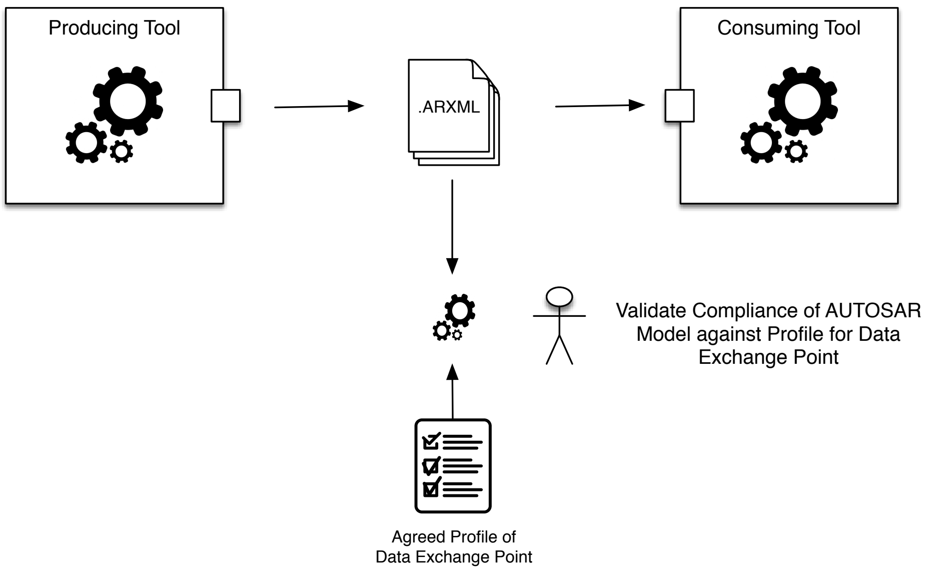

Producing Tool
.ARXML
Consuming Tool
Validate Compliance of AUTOSAR Model against Profile for Data Exchange Point
Agreed Profile of Data Exchange Point

Figure 8.4: Validation of Compliance of `AUTOSAR Models` with Profile of Data Exchange Point

[TPS_STDT_00118] Compliance with Profile of Data Exchange Point ⌈
A `Profile of Data Exchange Point` can tailor the AUTOSAR Data Format for a specific Data Exchange Point. This tailoring of `Data Format Elements` (Meta-Classes, Attributes, Constraints, Sdg usage) specifies a subset of the meta model that is relevant for this specific Data Exchange Point and defines which AUTOSAR and custom validation rules have to be evaluated. An `AUTOSAR Model` complies with a `Profile of Data Exchange Point` if all validation rules evaluate to true.⌋*(RS_-STDT_00121)* (see figure 8.4)

Note: The following patterns, meta classes and attributes focus on the description of self-contained `Profile of Data Exchange Points`. Support for authoring of profiles such as the composition of a profile out of profile assets is not yet covered.

8.2 General Patterns

8.2.1 Top Level Data Structure

[TPS_STDT_00120] Purpose of `DataExchangePoint` ⌈For a given `Data Exchange Point` the `DataExchangePoint` specifies the following aspects:

- Short description of the data exchange point using `longName`, `desc` and `introduction` (inherited from `Identifiable`).
- The `Baseline` of the AUTOSAR standard that is referenced by the profile
- High-Level specification of the data exchange point by selection of the relevant parts of the `AUTOSAR Specifications`.

- Detailed tailoring of the AUTOSAR Data Format (Meta-Classes, Attributes, Constraints, Special Data Group Definitions).

⌋*(RS_STDT_00101)*

For details see sections 8.3 and 8.4.

The aspects that are described by the DataExchangePoint are located on the Meta Level (M2 as described in [7]). On this level we can find the AUTOSAR Meta Model and the AUTOSAR XML Schema. Although a Profile of Data Exchange Point specifies information on M2 level we reuse the approach for the specification of the Profile of Data Exchange Point language that is already used by the AUTOSAR Template specifications. Using this approach, we can store a Profile of Data Exchange Point in a .arxml file and we can reuse existing meta classes such as ARPackage, Documentation, Identifiable, etc.

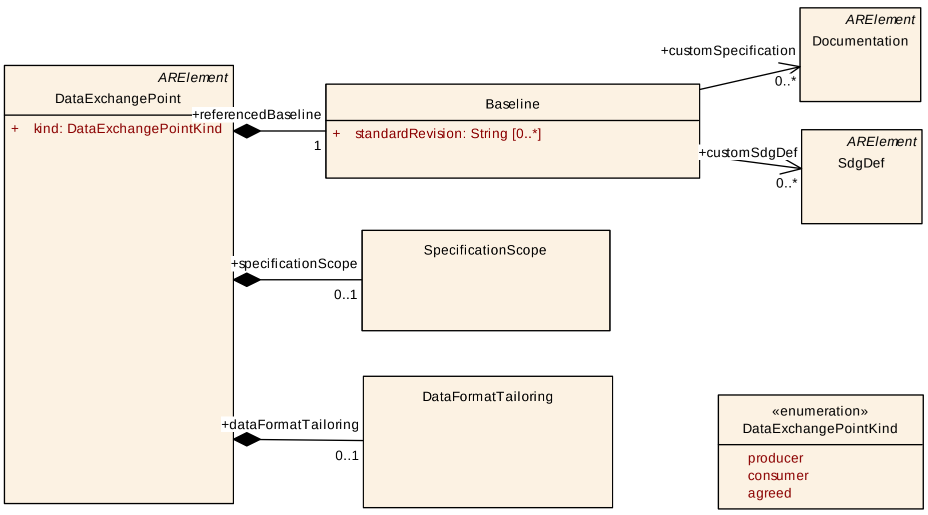


Figure 8.5: Overview of Data Exchange Point

[TPS_STDT_00121] High-level Overview Description of DataExchangePoint ⌈The high-level overview description is provided by means of its attributes longName, desc and introduction.⌋*(RS_STDT_00101, RS_STDT_00104)*

[TPS_STDT_00122] Purpose of Baseline ⌈Baseline specifies a baseline of the AUTOSAR standard that is used as a reference for all references to AUTOSAR Specification Elements in this DataExchangePoint. The baseline is specified by listing the AUTOSAR standards and their revisions. Custom defined functionality and deviations are described using the Documentation M1 Documentation capabilities.⌋*(RS_STDT_00105)*

[TPS_STDT_00211] Specification of the AUTOSAR Standards that are part of the Baseline ⌈AUTOSAR is modularised into several standards. A combination of those AUTOSAR standards in a specific version is identified by specifying the standardRevision of each included standard.⌋*(RS_STDT_00105)*

Note: the identifiers of the standards that have an impact on the XML schema are referenced in section "Covered Standards:" of the readme.txt that is part the MMOD_XMLSchema_230.

E.g.:
`standardRevision`[0]="FO 1.4.0",
`standardRevision`[1]="CP 4.4.0",
`standardRevision`[2]="AP 18-10"

[constr_2609] Single revision per AUTOSAR standard ⌈
The `standardRevision` may only contain a single revision per AUTOSAR standard. E.g. it is allowed to combine the AUTOSAR standards "Foundation" in revision 1.0.0 with the "Classic Platform" in revision 4.3.0. However, it is not allowed to reference the revisions 4.2.2 and 4.3.0 of the "Classic Platform" in the same `Baseline`.⌋*()*

Class	**DataExchangePoint**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint			
Note	The Data Exchange Point describes the relationship between a work product and its intended use in the methodology with a tailoring of the AUTOSAR templates. An informal description is provided by the 'desc' and 'introduction' attributes of the DataExchangePoint. The informal description SHOULD include the subject that is described by this data exchange point. E.g. • producible data of tool A, version x • consumable data of tool B, version y • agreed profile between partner A and partner B in project xyz **Tags:**atp.recommendedPackage=DataExchangePoints			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataFormat Tailoring	DataFormatTailoring	0..1	aggr	tailoring to the Autosar Exchange Data Format The subset and tailoring of the templates specifications (Meta-Classes, Attributes, Sdgs, Constraints, SpecItems) **Tags:**xml.sequenceOffset=40
kind	DataExchangePoint Kind	1	attr	Specifies the kind of this DataExchangePoint. It provides information if this DataExchangePoint represents • the output of a tool that produce data, • the input of a tool that consumes data or • an agreed profile
referenced Baseline	Baseline	1	aggr	The baseline of the AUTOSAR standard that is used as a reference within this Data Exchange Point. **Tags:**xml.sequenceOffset=10
specification Scope	SpecificationScope	0..1	aggr	The speficication of the relevant subset of Autosar standardized and custom specifications. **Tags:**xml.sequenceOffset=30

Table 8.1: DataExchangePoint

Class	**Baseline**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint			
Note	Specification of the baseline of the AUTOSAR standard this Data Exchange Point relates to. The baseline is specified by listing the AUTOSAR products and their revisions. Custom defined functionality and deviations to the standard can be provided as well. All references to specification elements in this Data Exchange Point refer to specification elements that are part of this specification baseline.			
Base	*ARObject*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
customSdgDef	SdgDef	*	ref	Reference to custom SdgDefs that extend the data format of this baseline, **Tags:**xml.sequenceOffset=30
custom Specification	Documentation	*	ref	Reference tof custom specifications that extend this baseline, **Tags:**xml.sequenceOffset=20
standard Revision	String	*	attr	Specifies a combination of revisions of AUTOSAR standards that are used as the specification baseline of this Data Exchange Point. All standard specification elements that are referenced by this Profile of Data Exchange Point have to be part of specifications that belong to the defined AUTOSAR standards. **Tags:**xml.sequenceOffset=10

Table 8.2: Baseline

Enumeration	**DataExchangePointKind**
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint
Note	Specifies the kind of a DataExchangePoint.
Literal	*Description*
agreed	the data exchange point description represents the agreed data exchange point that should be used during data exchange **Tags:**atp.EnumerationLiteralIndex=2
consumer	the data exchange point description represents the input of a consuming tool. **Tags:**atp.EnumerationLiteralIndex=1
producer	the data exchange point description represents the output of a producing tool. **Tags:**atp.EnumerationLiteralIndex=0

Table 8.3: DataExchangePointKind

Listing 8.1: Example of Top Level Structure

```
<AR-PACKAGE>
  <SHORT-NAME>DataExchangePoints</SHORT-NAME>
  <ELEMENTS>
    <DATA-EXCHANGE-POINT>
      <SHORT-NAME>ECU_System_Description</SHORT-NAME>
      <LONG-NAME>
        <L-4 L="EN">ECU System Description for Configuration of ComStack
             for Unsegmented Unmultiplexed Signal-Based Communication on
           CAN</L-4>
      </LONG-NAME>
      <DESC>
        <L-2 L="EN">This profile describes the data that is exchanged in
             the deliverable "ECU␣System␣Description" and focuses on data
             that is required for configuring the behavior of the ECU on
```

```
          the CAN network with respect to unsegmented signal-based
          communication. The profile shows the supported input of
          FancyCanStackConfigurator version 1.2.2</L-2>
    </DESC>
    <INTRODUCTION>
      <P>
        <L-1 L="EN">Consumer, Tool: FancyCanStackConfigurator version
            1.2.2, invoked using "fancy␣-buildCar"</L-1>
      </P>
    </INTRODUCTION>
    <KIND>CONSUMER</KIND>
    <REFERENCED-BASELINE>
      <STANDARD-REVISIONS>
        <STANDARD-REVISION>CP R4.3.0</STANDARD-REVISION>
        <STANDARD-REVISION>FO R1.0.0</STANDARD-REVISION>
      </STANDARD-REVISIONS>
        <CUSTOM-SPECIFICATION-REFS>
        <CUSTOM-SPECIFICATION-REF DEST="DOCUMENTATION">/VendorName/
            DataExchangePoints/CustomExtensions</CUSTOM-SPECIFICATION-
            REF>
      </CUSTOM-SPECIFICATION-REFS>
    </REFERENCED-BASELINE>
    <SPECIFICATION-SCOPE>
      <!-- -->
    </SPECIFICATION-SCOPE>
    <DATA-FORMAT-TAILORING>
      <!-- -->
    </DATA-FORMAT-TAILORING>
  </DATA-EXCHANGE-POINT>
  <DOCUMENTATION>
    <SHORT-NAME>CustomExtensions</SHORT-NAME>
    <DOCUMENTATION-CONTENT>
      <CHAPTER>
        <SHORT-NAME>RFCs</SHORT-NAME>
        <STRUCTURED-REQ>
          <SHORT-NAME>Example_RFC12345</SHORT-NAME>
          <DESCRIPTION>
            <P>
              <L-1 L="EN">Description of the change request</L-1>
            </P>
          </DESCRIPTION>
        </STRUCTURED-REQ>
      </CHAPTER>
    </DOCUMENTATION-CONTENT>
  </DOCUMENTATION>
</ELEMENTS>
</AR-PACKAGE>
```

8.2.2 Referencing Standardized Specification Elements

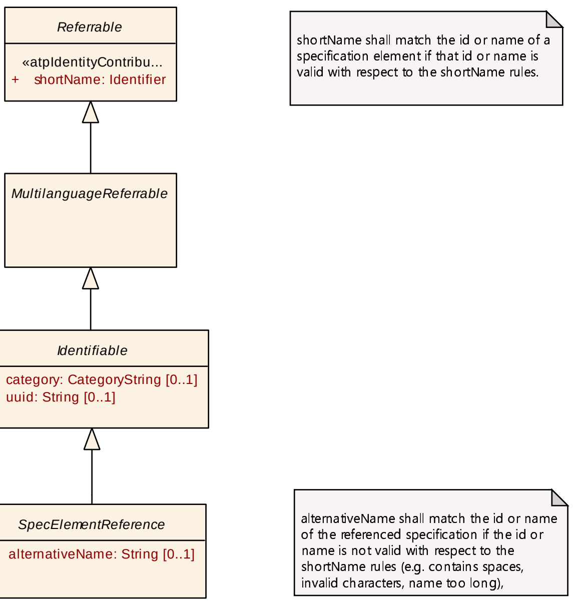

Referrable
«atpIdentityContribu...
+ shortName: Identifier
MultilanguageReferrable
Identifiable
+ category: CategoryString [0..1]
+ uuid: String [0..1]
SpecElementReference
+ alternativeName: String [0..1]
shortName shall match the id or name of a specification element if that id or name is valid with respect to the shortName rules.
alternativeName shall match the id or name of the referenced specification if the id or name is not valid with respect to the shortName rules (e.g. contains spaces, invalid characters, name too long),

Figure 8.6: Referencing Specification Elements

[TPS_STDT_00102] Referencing AUTOSAR Specification Elements via shortName ⌈If the name of the `AUTOSAR Specification Element` follows the rules of `shortName`s (see `Identifier`), then `SpecElementReference.shortName` shall describe the name of the referenced `AUTOSAR Specification Element.`⌋*(RS_STDT_00102, RS_STDT_00103, RS_STDT_00106, RS_STDT_00108, RS_STDT_00109)*

[TPS_STDT_00103] Referencing AUTOSAR Specification Elements via alternativeName ⌈If the name of the `AUTOSAR Specification Element` does not follow the rules of `shortName`s (see `Identifier`), then `alternativeName` shall describe the name of the referenced `AUTOSAR Specification Element`. The `shortName` shall contain a simplified name that is created using the following rules:

- replace all characters, that are not allowed by the shortName rules (see `Identifier`) by '_' (underline)
- If the shortName is longer than 128 characters, then the following algorithm applies:
  1. get the first 121 characters. This leaves room for a separator and a CRC number.
  2. append "_0x"
  3. append a CRC16 checksum in hex format (uppercase) of the original name. For more information about the CRC16 algorithm see [17].

⌋*(RS_STDT_00102, RS_STDT_00103, RS_STDT_00106, RS_STDT_00108, RS_STDT_00109)*

[constr_2610] No alternativeName if matching via shortName ⌈The alternativeName shall not be set if the referenced AUTOSAR Specification Element matches the rules of Identifier.⌋*()*

[constr_2611] Referenced AUTOSAR Specification Elements shall be part of the AUTOSAR Specification Baseline ⌈If the SpecElementReference references an AUTOSAR specification element then the shortName or alternativeName shall match the name of the AUTOSAR specification element in a specification that is part of the revision of the standard that is specified in Baseline.⌋*()*

See also example A.1.

Class	***SpecElementReference*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Common Patterns			
Note	This is a reference to a specification element in the Autosar standard.			
Base	*ARObject, Identifiable, MultilanguageReferrable, Referrable*			
Subclasses	*DataFormatElementReference, SpecElementScope*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
alternative Name	String	0..1	attr	Alternative name of a specification element if its name doesn't fit into the shortName. E.g. because the name contains spaces.

Table 8.4: SpecElementReference

8.2.3 Referencing Custom Specification Elements

[TPS_STDT_00104] Referencing Custom Specification Elements ⌈If it is required to define custom specification elements that go beyond the AUTOSAR standardized specification elements, then the Description of Data Exchange Points allows the referencing of those descriptions via AUTOSAR shortName path based references. If a reference to a custom Specification Element is defined, then this reference is used for identification of the Specification element. No matching of AUTOSAR defined Specification Elements via alternativeName or shortName applies.⌋*(RS_STDT_00102, RS_STDT_00103, RS_STDT_00106, RS_STDT_00108, RS_STDT_00109)*

[constr_2608] Custom extensions shall be part of the Documentation that is referenced by the Baseline ⌈If a SpecElementReference references a custom defined specification element, then this specification element shall be part of a Documentation that is referenced by the Baseline of this Profile.⌋*()*

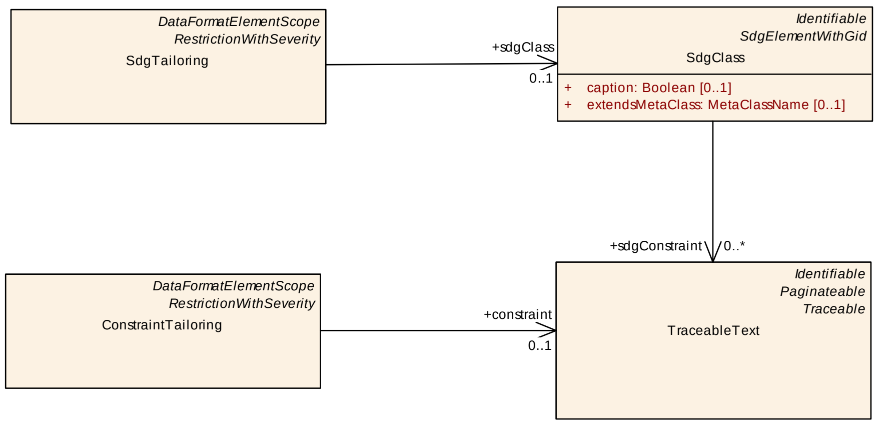


Figure 8.7: Referencing Custom Defined Constraints and Special Data Groups

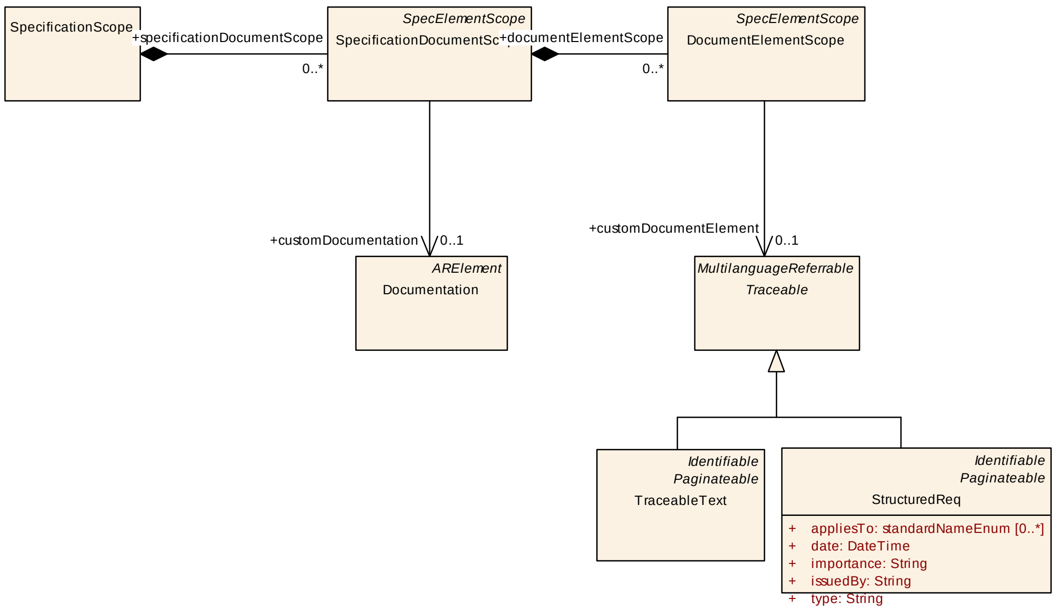


Figure 8.8: Referencing elements in custom specifications

See also example A.1.

8.2.4 Scoping of Specification Elements

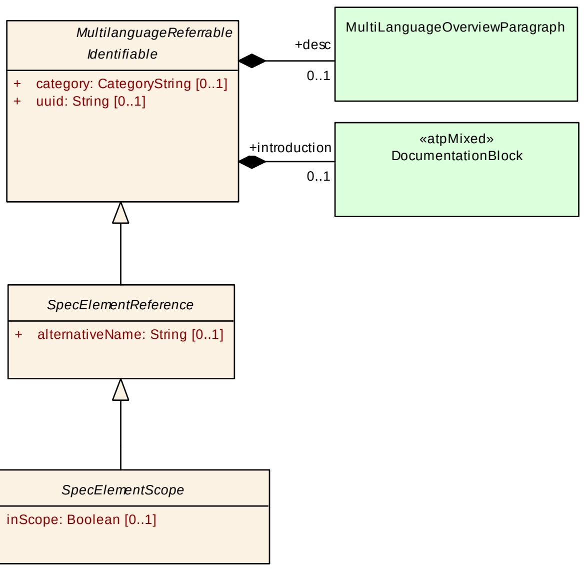


Figure 8.9: Scoping of Specification Elements

[TPS_STDT_00124] Purpose of `SpecElementScope` ⌈For all AUTOSAR specification elements and custom functionality a Data Exchange Point describes if a referenced specification element is relevant for the Data Exchange Point. If `inScope`==true, then the specification element is relevant. (e.g. a requirement needs to be fullfilled, a constraint is enabled, an attribute shall exist, ...). If `inScope`==false, then the specification element is not relevant. (e.g. a requirement does not apply, a constraint is disabled, it is not relevant if an attribute exists, ...)⌋*(RS_STDT_00103, RS_STDT_00106, RS_STDT_00108, RS_STDT_00109)*

Class	***SpecElementScope*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Common Patterns			
Note	This class defines if a specification element is relevant within the context of this data exchange point.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SpecElementReference*			
Subclasses	*DataFormatElementScope*, DocumentElementScope, SpecificationDocumentScope			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
inScope	Boolean	0..1	attr	indicates, if a specification element is relevant for this data exchange point. It is relevant if inScope==true. It is not relevant or don't care if inScope=false.

Table 8.5: SpecElementScope

8.2.5 Tailoring of Data Format Elements

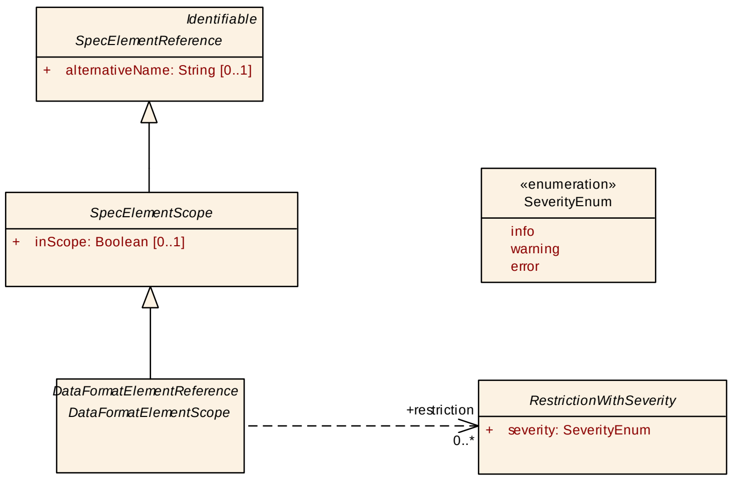


Figure 8.10: Tailoring of Data Format Elements

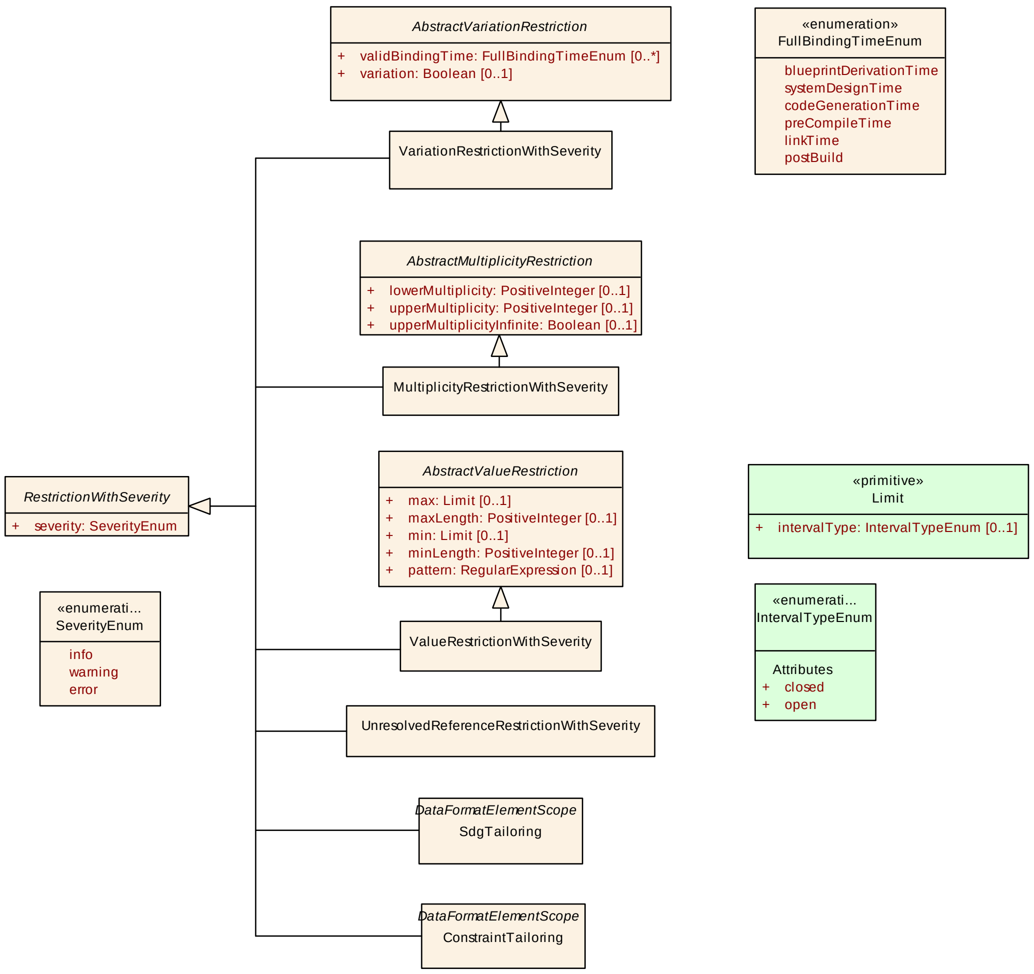

AbstractVariationRestriction
+ validBindingTime: FullBindingTimeEnum [0..*]
+ variation: Boolean [0..1]
VariationRestrictionWithSeverity
«enumeration»
FullBindingTimeEnum
blueprintDerivationTime
systemDesignTime
codeGenerationTime
preCompileTime
linkTime
postBuild
AbstractMultiplicityRestriction
+ lowerMultiplicity: PositiveInteger [0..1]
+ upperMultiplicity: PositiveInteger [0..1]
+ upperMultiplicityInfinite: Boolean [0..1]
MultiplicityRestrictionWithSeverity
RestrictionWithSeverity
+ severity: SeverityEnum
AbstractValueRestriction
+ max: Limit [0..1]
+ maxLength: PositiveInteger [0..1]
+ min: Limit [0..1]
+ minLength: PositiveInteger [0..1]
+ pattern: RegularExpression [0..1]
«primitive»
Limit
+ intervalType: IntervalTypeEnum [0..1]
«enumerati...
SeverityEnum
info
warning
error
«enumerati...
IntervalTypeEnum
Attributes
+ closed
+ open
ValueRestrictionWithSeverity
UnresolvedReferenceRestrictionWithSeverity
DataFormatElementScope
SdgTailoring
DataFormatElementScope
ConstraintTailoring

Figure 8.11: Restrictions with Severity

[TPS_STDT_00126] Definition: Data Format Elements ⌈`Data Format Elements` are `Meta-Classes`, `Meta-Attributes`, `Constraints` and `Special Data`

Group Definitions that have direct impact on the AUTOSAR data exchange format.⌋*(RS_STDT_00106, RS_STDT_00114)*

[TPS_STDT_00186] Scope and Restrictions of Data Format Elements ⌈A DataFormatElementScope defines if a Data Format Element is relevant for the Data Exchange Point. If inScope == true then restrictions with severity specify additional constraints and their severity.⌋*(RS_STDT_00106, RS_STDT_00114)*

[TPS_STDT_00172] Purpose of RestrictionWithSeverity ⌈A RestrictionWithSeverity defines constraints on the model. The severity describes the severity level that is reported in case the restriction is violated.⌋*(RS_STDT_00114)*

[TPS_STDT_00173] Purpose of ValueRestrictionWithSeverity ⌈A ValueRestrictionWithSeverity defines constraints on the value of a simple attribute (string, integer, float).⌋*(RS_STDT_00113)*

[TPS_STDT_00174] Purpose of MultiplicityRestrictionWithSeverity ⌈A MultiplicityRestrictionWithSeverity specifies the valid number of occurrences of an element in the current context.⌋*(RS_STDT_00106, RS_STDT_00110)*

[TPS_STDT_00175] Purpose of VariationRestrictionWithSeverity ⌈A VariationRestrictionWithSeverity specifies constraints on the usage of variation and on the valid binding times.⌋*(RS_STDT_00125)*

[TPS_STDT_00176] Context specific Tailoring ⌈The tailoring of a Meta Class can optionally depend on

- the role by which an object is aggregated or referenced and
- conditions that depend for instance on attribute values (e.g. the value of "category")

⌋*(RS_STDT_00125)*

Class	***RestrictionWithSeverity*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Common Patterns			
Note	A restriction that has a severity. The severity describes the severity level that is reported in case the restriction is violated.			
Base	*ARObject*			
Subclasses	ConstraintTailoring, MultiplicityRestrictionWithSeverity, SdgTailoring, UnresolvedReferenceRestriction WithSeverity, ValueRestrictionWithSeverity, VariationRestrictionWithSeverity			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
severity	SeverityEnum	1	attr	Severity level that is reported in case the restriction is violated.

Table 8.6: RestrictionWithSeverity

Enumeration	**SeverityEnum**
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint
Note	Definition of severity levels.
Literal	***Description***
error	Something is not right. High risk of interoperability issues. **Tags:**atp.EnumerationLiteralIndex=2
info	Something was found that is worth mentioning. Low risk of interoperability issues. **Tags:**atp.EnumerationLiteralIndex=0
warning	Something might be wrong depending on the context. Medium risk of interoperability issues. **Tags:**atp.EnumerationLiteralIndex=1

Table 8.7: SeverityEnum

Class	**ValueRestrictionWithSeverity**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	Specifies valid values of primitive data types. A value is valid if all rules defined by this ValueRestriction evaluate to true.			
Base	*ARObject*, *AbstractValueRestriction*, *RestrictionWithSeverity*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 8.8: ValueRestrictionWithSeverity

Class	***AbstractValueRestriction*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ModelRestrictionTypes			
Note	Restricts primitive values. A value is valid if all rules that are defined by this restriction evaluate to true.			
Base	*ARObject*			
Subclasses	PrimitiveAttributeCondition, *SdgAbstractPrimitiveAttribute*, ValueRestrictionWithSeverity			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
max	Limit	0..1	attr	Specifies the upper bounds for numeric values.
maxLength	PositiveInteger	0..1	attr	Specifies the maximum number of characters of textual values.
min	Limit	0..1	attr	Specifies the lower bounds for numeric values.
minLength	PositiveInteger	0..1	attr	Specifies the minimal number of characters of textual values.
pattern	RegularExpression	0..1	attr	Defines the exact sequence of characters that are acceptable.

Table 8.9: AbstractValueRestriction

Class	**MultiplicityRestrictionWithSeverity**
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring
Note	Restriction that specifies the valid number of occurrences of an element in the current context.
Base	*ARObject*, *AbstractMultiplicityRestriction*, *RestrictionWithSeverity*

▽

△

Class	**MultiplicityRestrictionWithSeverity**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 8.10: MultiplicityRestrictionWithSeverity

Class	***AbstractMultiplicityRestriction*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ModelRestrictionTypes			
Note	Restriction that specifies the valid number of occurrences of an element in the current context.			
Base	*ARObject*			
Subclasses	*AttributeCondition*, MultiplicityRestrictionWithSeverity, *SdgAttribute*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
lowerMultiplicity	PositiveInteger	0..1	attr	Specifies the minimal number of times an object shall occur. If this primitive attribute is not set, then the object is optional.
upperMultiplicity	PositiveInteger	0..1	attr	Specifies the maximum number of times an object may occur. If this primitive attribute is not set, then there is no limit with respect to the maximum occurrence.
upperMultiplicity Infinite	Boolean	0..1	attr	This explicitly specifies, that the upper multiplicity is NOT restricted. Note: The use of 'upperMultiplicityInfinite' and 'upperMultiplicity' is mutual exclusive.

Table 8.11: AbstractMultiplicityRestriction

Class	**VariationRestrictionWithSeverity**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	Defines constraints on the usage of variation and on the valid binding times.			
Base	*ARObject*, *AbstractVariationRestriction*, *RestrictionWithSeverity*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 8.12: VariationRestrictionWithSeverity

Class	***AbstractVariationRestriction*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ModelRestrictionTypes			
Note	Defines constraints on the usage of variation and on the valid binding times.			
Base	*ARObject*			
Subclasses	SdgAggregationWithVariation, SdgForeignReferenceWithVariation, SdgPrimitiveAttributeWithVariation, VariationRestrictionWithSeverity			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
validBinding Time	FullBindingTimeEnum	*	attr	List of valid binding times. **Tags:**xml.sequenceOffset=20
variation	Boolean	0..1	attr	Defines if the AUTOSAR model may define a Variation Point at this location. **Tags:**xml.sequenceOffset=10

Table 8.13: AbstractVariationRestriction

Enumeration	**FullBindingTimeEnum**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ModelRestrictionTypes
Note	This enumeration specifies the BindingTimes that can be used in AUTOSAR models.
Literal	***Description***
blueprintDerivation Time	The point in time when an object is created from a blueprint. **Tags:**atp.EnumerationLiteralIndex=0
codeGeneration Time	• Coding by hand, based on requirements document. • Tool based code generation, e.g. from a model. • The model may contain variants. • Only code for the selected variant(s) is actually generated. **Tags:**atp.EnumerationLiteralIndex=2
linkTime	Configure what is included in object code, and what is omitted Based on which variant(s) are selected E.g. for modules that are delivered as object code (as opposed to those that are delivered as source code) **Tags:**atp.EnumerationLiteralIndex=4
postBuild	PostBuild is the binding time which is bound latest at startup of the ECU. In other words this is everything between creation of the executable program and startup of the ECU. **Tags:**atp.EnumerationLiteralIndex=5
preCompileTime	This is typically the C-Preprocessor. Exclude parts of the code from the compilation process, e.g., because they are not required for the selected variant, because they are incompatible with the selected variant, because they require resources that are not present in the selected variant. Object code is only generated for the selected variant(s). The code that is excluded at this stage code will not be available at later stages. **Tags:**atp.EnumerationLiteralIndex=3
systemDesignTime	• Designing the VFB. • Software Component types (PortInterfaces). • SWC Prototypes and the Connections between SWCprototypes. • Designing the Topology • ECUs and interconnecting Networks • Designing the Communication Matrix and Data Mapping **Tags:**atp.EnumerationLiteralIndex=1

Table 8.14: FullBindingTimeEnum

8.2.6 Effective vs. Serialized Profile

[TPS_STDT_00105] Serialized Profile ⌈
The `Serialized Profile of Data Exchange Point` is the `ARXML Description` of a `Profile of Data Exchange Points`. This ARXML representation shall explicitly specify the parts of a Profile that deviate from the default values. It may explicitly specify values that do not deviate from the default values.⌋*(RS_STDT_00120)*
(see section 8.5)

[TPS_STDT_00106] Effective Profile ⌈
The `Effective Profile of Data Exchange Point` is a logical representation of a Profile that provides

- a `scope` for all AUTOSAR Specifications and their elements and

- a tailoring and restrictions for each Meta Class and Attribute, Constraint, etc.

of a dedicated AUTOSAR revision. It is calculated by applying the default values that are described in section 8.5 whenever the given profile does not explicitly specify a value.⌋*(RS_STDT_00120)*

For details see section 8.5.

8.2.7 Documentation of Rationales

[TPS_STDT_00170] Local documentation of Rationale ⌈
desc and introduction of the SpecElementScope objects can be used to document why something is in scope or tailored in a specific way.⌋*(RS_STDT_00115)*

[TPS_STDT_00168] Share documentation of Rationale ⌈
A DocumentElementScope can reference multiple DataFormatElementReferences in order to document that it is the rationale for the referenced tailorings.⌋*(RS_STDT_00115)*

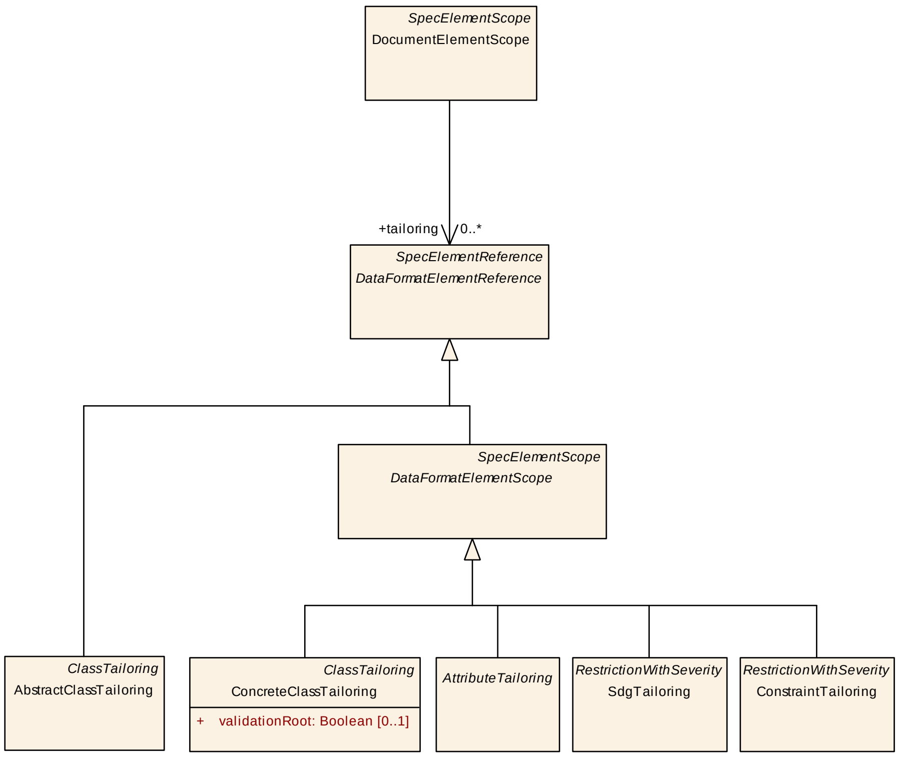


Figure 8.12: Shared Rationale

Class	DataFormatElementReference (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Common Patterns			
Note	Superclass of all references to specification elements that have direct impact on the data exchange format (Meta-Classes, Meta-Attributes, constraints, SdgDefs)			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SpecElementReference*			
Subclasses	AbstractClassTailoring, *DataFormatElementScope*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 8.15: DataFormatElementReference

Class	DataFormatElementScope (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	This class specifies if a Meta Class, Meta Attribute, Constraint or SdgDef is relevant for the Data Exchange Point.			
Base	*ARObject*, *DataFormatElementReference*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Spec ElementReference*, *SpecElementScope*			
Subclasses	*AttributeTailoring*, ConcreteClassTailoring, ConstraintTailoring, SdgTailoring			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 8.16: DataFormatElementScope

8.2.8 Validation Semantics

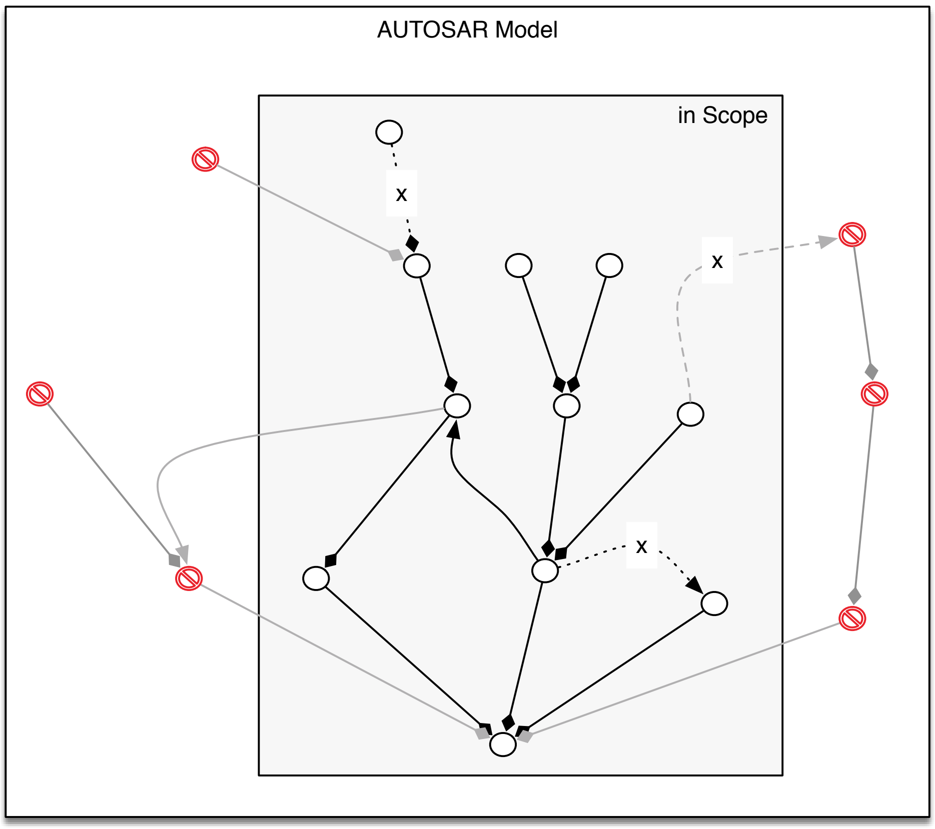
[TPS_STDT_00127] Validation Environment ⌈The evaluation of the compliance of an `AUTOSAR Model` with a profile assumes that:

- The model is loaded and merged according to the `splitable` rules as defined in [7].
- Default values that are standardized by AUTOSAR are applied according to the strategy defined in `PrimitiveAttributeTailoring.defaultValue-Handling`.
- Variation is bound (temporarily).

⌋*(RS_STDT_00111)*

[TPS_STDT_00129] Semantics of `DataFormatElementScope` with `in-Scope`==true ⌈If `inScope` of `DataFormatElementScope` is true then all attached restrictions are enabled otherwise they are disabled. E.g. If a meta class is out of scope, then the AUTOSAR model may contain an instance of that meta class. However this instance is not subject of validation.⌋*(RS_STDT_00106, RS_STDT_00121)*

Figure 8.13 visualizes the semantics of the scope of `DataFormatElementScope`.



	in Scope	out of Scope
instance of Class	○	🚫
aggregation or attribute	◆——	◆· x ··
reference	——▶	·· x ·▶

Figure 8.13: Example of AUTOSAR Model with Scoping.

[TPS_STDT_00130] Navigation strategy for validation ⌈The evaluation of the restrictions that are enabled or implied by the profile focuses on the subset of the `AUTOSAR Model` that is reachable from one or more validationRoot objects using the following navigation strategy. In contrast to the scoping based on meta classes and attributes as described in [TPS_STDT_00129] this navigation strategy allows to collect all objects in the current model that are actually used directly or indirectly by the instances of `ConcreteClassTailoring`s with `inScope`==true. E.g. It can be used to differentiate interfaces that are used from interfaces that are not used. Interfaces that are not used may contain errors while interfaces that are used should be valid.

- Start with an instance of a Meta-Class that is specified as a root element for the validation (`ConcreteClassTailoring.validationRoot`==true). If there are more than one validationRoot elements, then the validating tool should support the selection of one or more elements that are subject of validation.

- Follow the aggregations if the following preconditions evaluate to true:
  1. The aggregation is in scope and the aggregation is not explicitly excluded (AggregationTailoring.inScope==true AND AggregationTailoring.multiplicityRestriction.upperMultiplicity!=0) AND
  2. The aggregated object is in scope and it is not explicitly excluded (ConcreteClassTailoring.inScope==true AND ConcreteClassTailoring.multiplicityRestriction.upperMultiplicity!=0)
- Follow the references if the following preconditions evaluate to true:
  1. The reference is in scope and it is not explicitly excluded (ReferenceTailoring.inScope==true AND ReferenceTailoring.multiplicityRestriction.upperMultiplicity!=0) AND
  2. The referneced object is in scope and and not explicitly excluded (ConcreteClassTailoring.inScope==true AND ConcreteClassTailoring.multiplicityRestriction.upperMultiplicity!=0)

⌋*(RS_STDT_00107)*

Figure 8.14 shows an example of the application of the aforementioned navigation strategy.

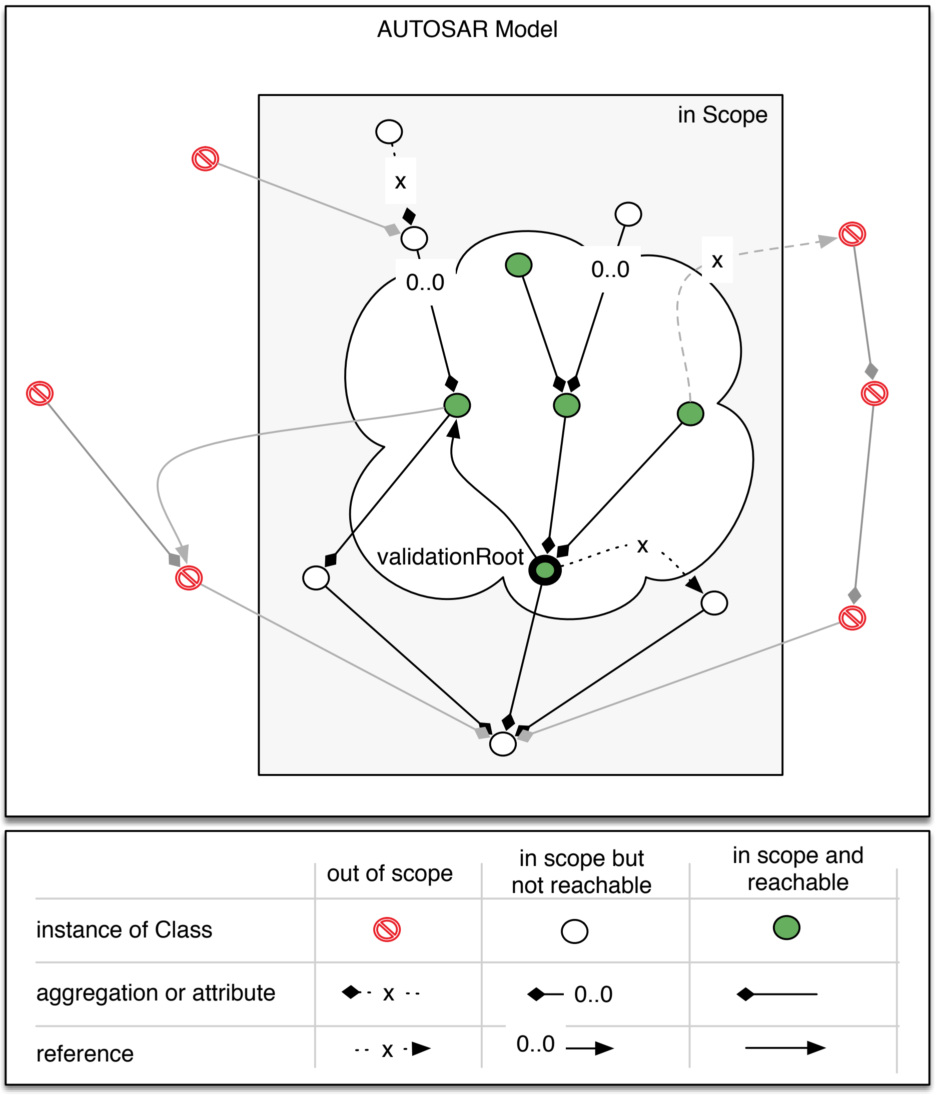


Figure 8.14: Example of AUTOSAR Model with Scoping (Elements that are reachable during validation are marked green)

8.3 Scoping of Specifications

[TPS_STDT_00156] Purpose of `SpecificationScope` ⌈The `SpecificationScope` specifies the subset of `AUTOSAR Specification`s and `AUTOSAR specification element`s that is relevant for this `DataExchangePoint`.⌋*(RS_STDT_00102, RS_STDT_00103)* (See 8.15)

[TPS_STDT_00188] Purpose of `SpecificationDocumentScope` ⌈The `SpecificationDocumentScope` if an AUTOSAR or custom specification is in scope of this `DataExchangePoint`. Autosar specifications are identified by their title. Custom specifications are referenced by `SpecificationDocumentScope.customDocumentation`.⌋*(RS_STDT_00102, RS_STDT_00103)*

[TPS_STDT_00187] Purpose of DocumentElementScope ⌈The DocumentElementScope specifies if an element in an AUTOSAR or custom specification is relevant for this DataExchangePoint. Elements of Autosar Specifications are identified by their Id (e.g. TPS_STDT_00187) that is composed according to [TPS_STDT_00042] or its name if the specification element is a SPEM Work Definition or SPEM Work Product in the Methodology specification [18]. Custom elements are referenced by DocumentElementScope.customDocumentElement.⌋*(RS_STDT_00102, RS_-STDT_00103)*

[TPS_STDT_00123] Guidance on how to specify SpecificationDocumentScope and DocumentElementScope ⌈When specifying the SpecificationDocumentScopes and DocumentElementScopes of a Data Exchange Point then the author should focus on Autosar Specifcations and Specification Elements that describe the current status of the data and on the description of how the data will be used after data exchange.⌋*(RS_STDT_00102, RS_STDT_00103)*

For example, a Profile of Data Exchange Point should refer to the Autosar Specification "Methodology" [18] and should refer to a deliverable. Additionally, it should describe which follow-up activities are intended to be performed based on that deliverable.
However, the author does not need to describe how the deliverable was produced.

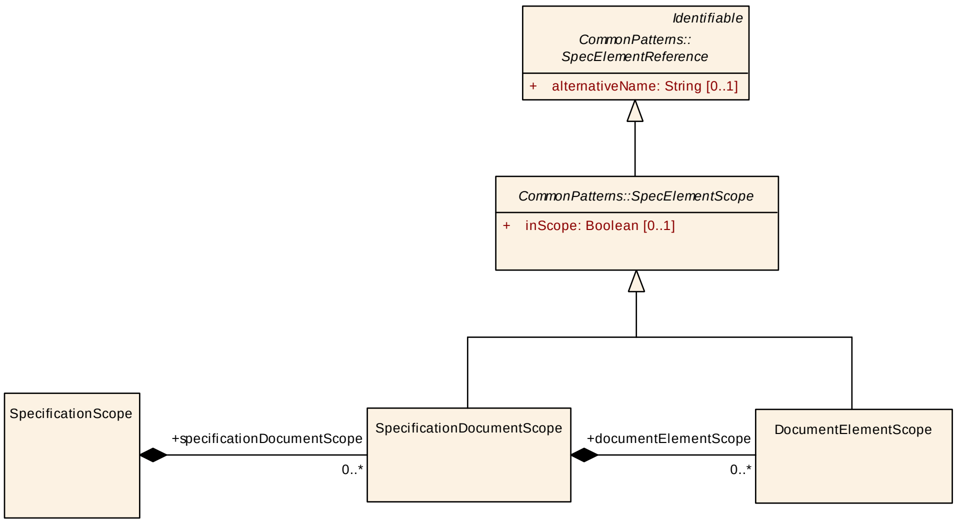

Identifiable
CommonPatterns::SpecElementReference
+ alternativeName: String [0..1]
CommonPatterns::SpecElementScope
+ inScope: Boolean [0..1]
SpecificationScope
+specificationDocumentScope
0..*
SpecificationDocumentScope
+documentElementScope
0..*
DocumentElementScope

Figure 8.15: Overview SpecificationScope

Class	**SpecificationScope**
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::SpecificationScope
Note	Specification of the relevant subset of Autosar specifications.
Base	*ARObject*

▽

△

Class	**SpecificationScope**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
specification Document Scope	SpecificationDocument Scope	*	aggr	The Autosar or custom specifications that contain that are considered in this Data Exchange Point.

Table 8.17: SpecificationScope

Class	**SpecificationDocumentScope**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchange Point::SpecificationScope			
Note	Represents a standardized or custom specification document such as Software Component Template, Main Requirements, Specification of Communication, etc. Autosar specifications are referenced via their title.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SpecElementReference*, *SpecElement Scope*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
custom Documentation	Documentation	0..1	ref	reference to a custom defined specification.
document ElementScope	DocumentElement Scope	*	aggr	An element with a name or ID that is specified in the Spcification Document.

Table 8.18: SpecificationDocumentScope

Class	**DocumentElementScope**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchange Point::SpecificationScope			
Note	Specifies if a specification element such as a requirement, specification, deliverable, artifact, task definition or activity is in scope of this data exchange point. The DocumentElementScope may reference all specification elements that have a name or ID. The only exception are Meta Classes, Meta Attribute and constraints which are handled in the Data Format Tailoring section of the Profile of Data Exchange Point. Elements of Autosar specification documents are referenced via their ID (requirement, specification items) or name (deliverable, artifact, task definition or activity)			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SpecElementReference*, *SpecElement Scope*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
custom Document Element	Traceable	0..1	ref	Reference to a custom defined specification element.
tailoring	DataFormatElement Reference	*	ref	Data Format Element that is implied by this element in the specification. Used to share one rationale for more tailorings.

Table 8.19: DocumentElementScope

8.3.1 Addition Constraints

None

8.4 Tailoring of Data Format Elements

[TPS_STDT_00157] Purpose of `DataFormatTailoring` ⌈The `DataFormatTailoring` tailors the `AUTOSAR Data Exchange Format` for a specific data exchange point. This includes:

- Identification if meta-classes, attributes, constraints, or SDGs are relevant (inScope) of the data exchange point.
- Restriction of multiplicities, attribute values and use of variation
- Specification of severities in case a rule is violated
- Declaration of completeness
- Declaration on how to handle AUTOSAR defined default values
- Specification of the structure of Special Data Group
- Definition of the start element and navigation strategy during validation

⌋*(RS_STDT_00106, RS_STDT_00107, RS_STDT_00108, RS_STDT_00109, RS_STDT_00110, RS_STDT_00111, RS_STDT_00113, RS_STDT_00114, RS_STDT_00116)*

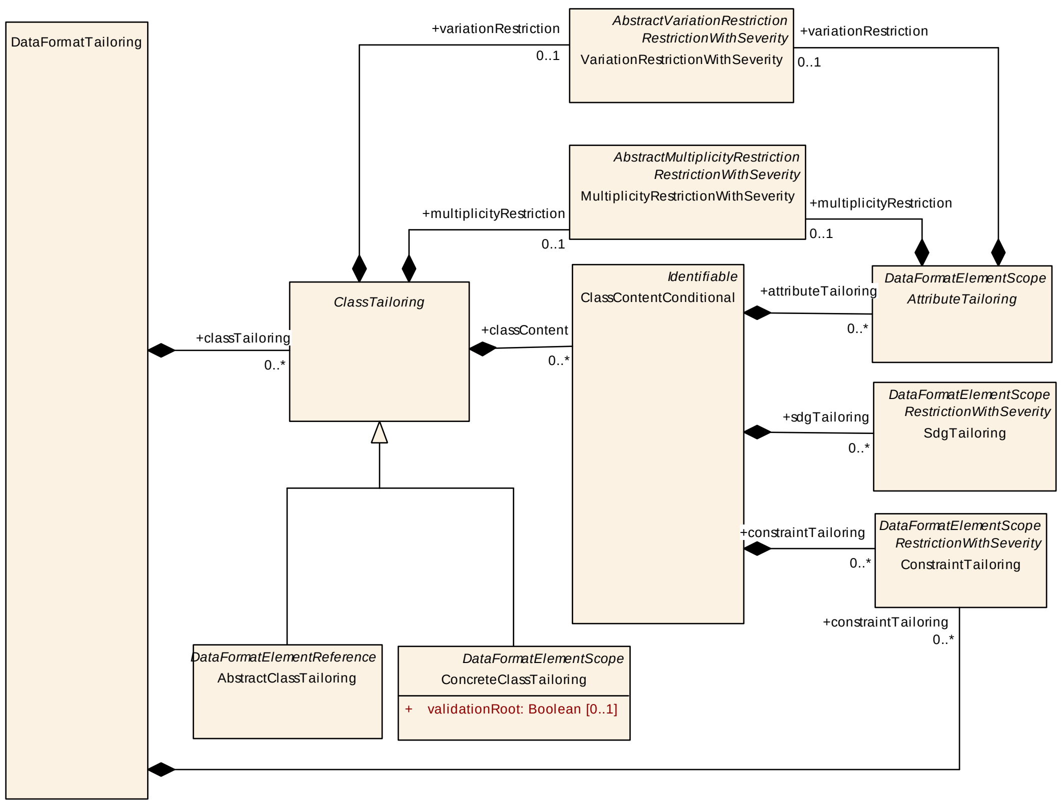


Figure 8.16: Overview of DataFormatTailoring

8.4.1 Tailoring of Classes

8.4.1.1 Description

[TPS_STDT_00145] Purpose of `ClassTailoring` ⌈The `ClassTailoring` tailors a `Meta-Class` with respect to the relevant attributes, applicable constraints, number of occurances, use of variation and the extensibility via Sdgs.⌋*(RS_STDT_00106)*

[TPS_STDT_00109] AUTOSAR Standardized Concrete Meta-Classes ⌈AUTOSAR standardized concrete meta-classes are specified by concrete UML classes (abstract=false) that are not representing primitive types (no stereotype «primitive», «enumeration») in the AUTOSAR MetaModel [19], sub-packages "M2::AUTOSAR DataFormat" or "M2::MSR".

The reference is established via the name of the UML classes.⌋*(RS_STDT_00106)*

[TPS_STDT_00146] AUTOSAR Standardized Abstract Meta-Classes ⌈AUTOSAR standardized abstract meta-classes are specified by abstract UML classes (abstract=true) that are not representing primitive types (no stereotype «primitive», «enumeration») in the AUTOSAR Meta Model [19], sub-packages "M2::AUTOSAR DataFormat" or "M2::MSR".

The reference is established via the name of the UML classes.⌋*(RS_STDT_00106)*

[TPS_STDT_00177] Global `ClassTailoring` ⌈`ClassTailoring`s that are directly contained by `DataFormatTailoring` are global `ClassTailoring`s. If a global `ConcreteClassTailoring` is `inScope` then its tailorings and restrictions apply for all reachable instances of the class.⌋*(RS_STDT_00106)*

[TPS_STDT_00178] Role Specific `ClassTailoring` ⌈`ClassTailoring`s that are contained by `AggregationTailoring.typeTailoring` or `ReferenceTailoring.typeTailoring` are context specific `ClassTailoring`s. Their tailorings and restrictions are applicable if

- `inScope` == true AND
- the object in the AUTOSAR model is aggregated or referenced by the specified role.

⌋*(RS_STDT_00106)*

See also examples A.2, A.3 and A.4.

[TPS_STDT_00179] Conditional `ClassTailoring` ⌈The content model of a meta-class is tailored via one or more `ClassContentConditional`s. Multiple `ClassContentConditional`s may apply for a single object.⌋*(RS_STDT_00106)*

[TPS_STDT_00180] Invariant Content Model ⌈If `condition` does not exist, then the tailorings and restrictions defined by this `ClassContentConditional` shall be applied for all instances within the current context.⌋*(RS_STDT_00106)*

[TPS_STDT_00181] Conditional Content Model ⌈If condition is defined, then the restrictions defined by this ClassContentConditional shall apply if that condition evaluates to true.⌋*(RS_STDT_00106)*

See also example A.5.

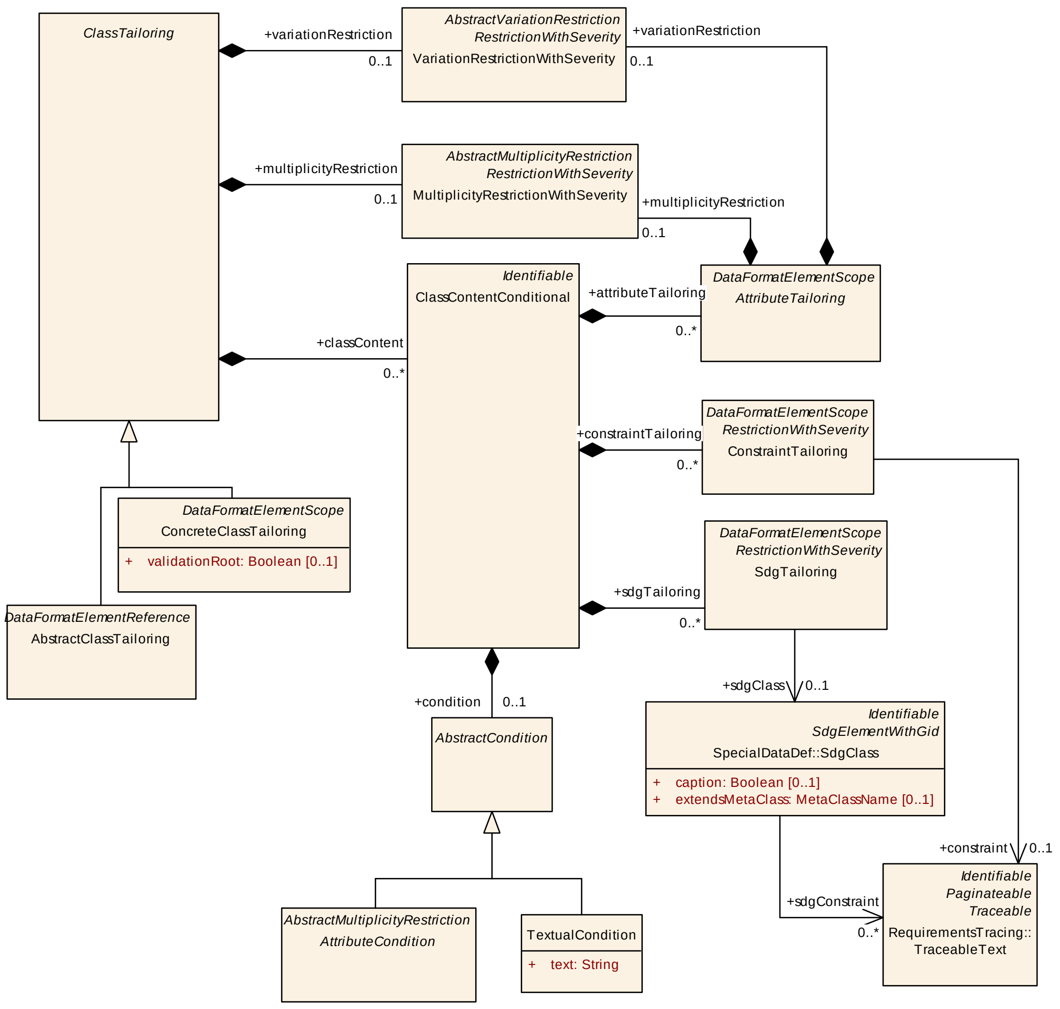


Figure 8.17: Overview of ClassTailoring

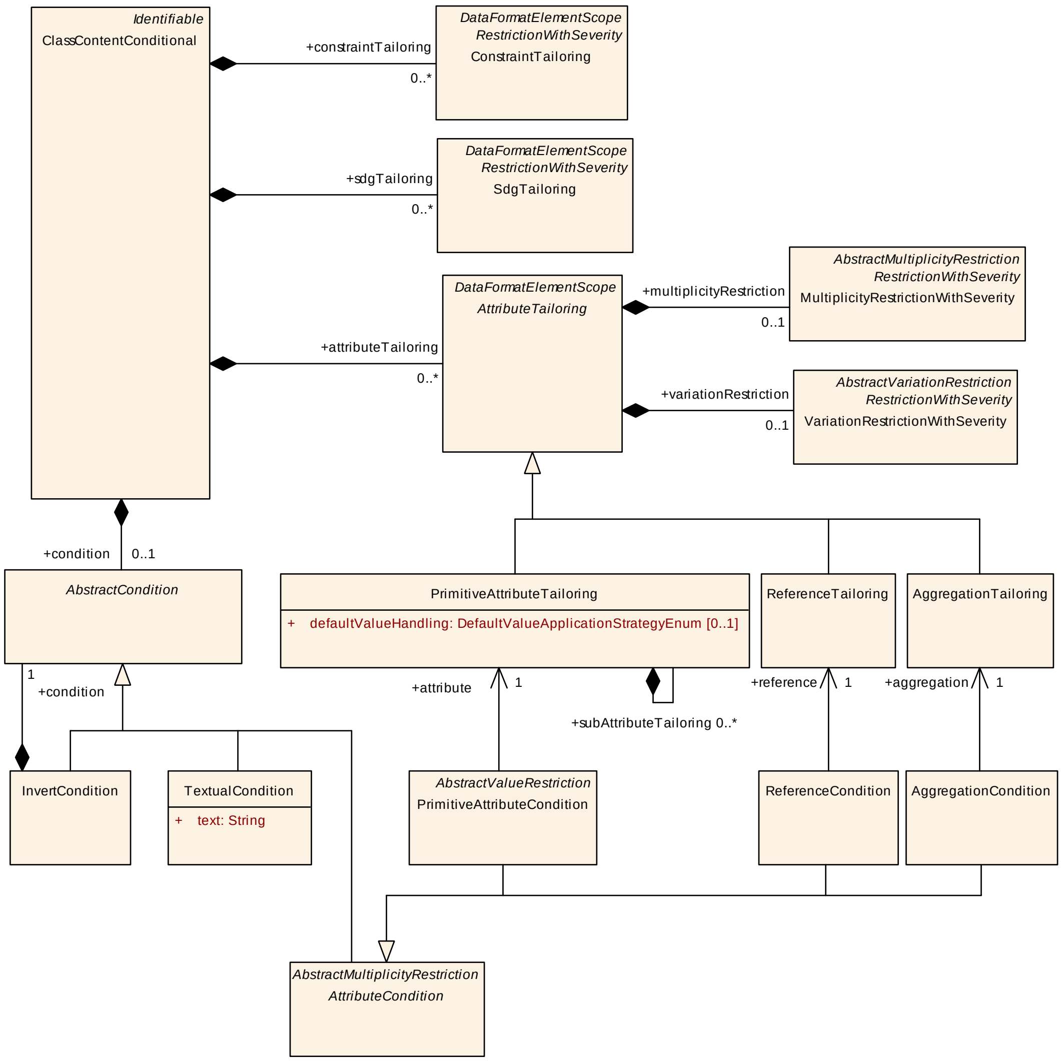


Figure 8.18: Overview of ClassContentConditional

Class	**AbstractClassTailoring**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	Tailoring of abststract classes in the AUTOSAR meta-model			
Base	*ARObject*, *ClassTailoring*, *DataFormatElementReference*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SpecElementReference*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 8.20: AbstractClassTailoring

Class	***AbstractCondition*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	A premise upon which the fulfillment of an agreement depends			
Base	*ARObject*			
Subclasses	*AttributeCondition*, InvertCondition, TextualCondition			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 8.21: AbstractCondition

Class	**AggregationCondition**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	The AggregationCondition evaluates to true, if the referenced aggregation is accepted by all rules of this condition.			
Base	*ARObject*, *AbstractCondition*, *AbstractMultiplicityRestriction*, *AttributeCondition*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
aggregation	AggregationTailoring	1	ref	The aggregation that has to be accepted by the restrictions of this AggregationCondition

Table 8.22: AggregationCondition

Class	***AttributeCondition*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	The AttributeCondition evaluates to true, if the referenced attribute is accepted by all rules of this condition.			
Base	*ARObject*, *AbstractCondition*, *AbstractMultiplicityRestriction*			
Subclasses	AggregationCondition, PrimitiveAttributeCondition, ReferenceCondition			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 8.23: AttributeCondition

Class	***ClassTailoring*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	The ClassTailoring is an abstract class that allows the tailoring of its attributes, applicable constraints and Sdgs.			
Base	*ARObject*			
Subclasses	AbstractClassTailoring, ConcreteClassTailoring			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
classContent	ClassContent Conditional	*	aggr	Specifies the accepted / not accepted content of the class. All rules apply that fullfill the condition of the Class ContentConditional **Tags:**xml.sequenceOffset=30

▽

△

Class	***ClassTailoring*** (abstract)			
multiplicity Restriction	MultiplicityRestriction WithSeverity	0..1	aggr	Specifies the multiplicity of the class in the current context. **Tags:**xml.sequenceOffset=10
variation Restriction	VariationRestrictionWith Severity	0..1	aggr	Specifies restrictions on the usage of variant handling. **Tags:**xml.sequenceOffset=20

Table 8.24: ClassTailoring

Class	**ClassContentConditional**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	Specifies the valid content of the class. The content can optionally depend on a condition. (E.g. value of attribute 'category')			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
attribute Tailoring	AttributeTailoring	*	aggr	Tailorings of the owned and inherited attributes of this Meta Classes **Tags:**xml.sequenceOffset=20
condition	AbstractCondition	0..1	aggr	The rules on the content of this class are enabled if the condition validates to true. **Tags:**xml.sequenceOffset=10
constraint Tailoring	ConstraintTailoring	*	aggr	Specification of tailorings of Constraints of that are owned by this Meta Classes **Tags:**xml.sequenceOffset=30
sdgTailoring	SdgTailoring	*	aggr	Specification of the applicable Special Data Group **Tags:**xml.sequenceOffset=40

Table 8.25: ClassContentConditional

Class	**ConcreteClassTailoring**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	Tailoring of concrete meta classes.			
Base	*ARObject*, *ClassTailoring*, *DataFormatElementReference*, *DataFormatElementScope*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SpecElementReference*, *SpecElementScope*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
validationRoot	Boolean	0..1	attr	Specification if this concrete Meta-Class is a root element for validation. I.e.: The validation starts at an object of this concrete Meta-Class and continues by following all aggregations and references that are in scope of this Data Exchange Point. **Tags:**xml.sequenceOffset=10

Table 8.26: ConcreteClassTailoring

Class	**InvertCondition**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	inverts the nested condition			
Base	*ARObject*, *AbstractCondition*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
condition	AbstractCondition	1	aggr	The inverted condition

Table 8.27: InvertCondition

Class	**PrimitiveAttributeCondition**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	The PrimitiveAttributeCondition evaluates to true, if the referenced primitive attribute is accepted by all rules of this condition.			
Base	*ARObject*, *AbstractCondition*, *AbstractMultiplicityRestriction*, *AbstractValueRestriction*, *AttributeCondition*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
attribute	PrimitiveAttribute Tailoring	1	ref	The primitive attribute that has to be accepted by the restrictions of this PrimitiveAttributeCondition

Table 8.28: PrimitiveAttributeCondition

Class	**ReferenceCondition**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	The ReferenceCondition evaluates to true, if the referenced reference is accepted by all rules of this condition.			
Base	*ARObject*, *AbstractCondition*, *AbstractMultiplicityRestriction*, *AttributeCondition*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
reference	ReferenceTailoring	1	ref	The reference that has to be accepted by the restrictions of this ReferenceCondition

Table 8.29: ReferenceCondition

Class	**TextualCondition**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	Specifies additional conditions for one or more model elements. The condition is described using human language.			
Base	*ARObject*, *AbstractCondition*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
text	String	1	attr	Human language description of the condition.

Table 8.30: TextualCondition

8.4.1.2 Additional Constraints

[constr_2612] `shortName` of `ConcreteClassTailoring` shall match the name of an AUTOSAR specified concrete meta-class ⌈`shortName` of `Concrete-`

ClassTailoring shall match the name of an AUTOSAR specified concrete meta-class).⌋*()*

[constr_2613] shortName of AbstractClassTailoring shall match the name of an AUTOSAR specified abstract meta-class ⌈shortName of AbstractClassTailoring shall match the name of an AUTOSAR specified abstract meta-class).⌋*()*

[constr_2614] PrimitiveAttributeCondition.attribute shall reference invariant owned PrimitiveAttributeTailoring, only ⌈The following conditions need to evaluate to true for PrimitiveAttributeCondition.attribute:

- The referenced PrimitiveAttributeTailoring is owned by an ClassContentConditional that has no condition (invariant class content) **AND**
- The ClassContentConditional that owns the referenced PrimitiveAttributeTailoring and the ClassContentConditional that owns this PrimitiveAttributeCondition are owned by the same ClassTailoring.

⌋*()*

[constr_2615] AggregationCondition.aggregation shall reference invariant owned AggregationTailoring, only ⌈The following conditions need to evaluate to true for AggregationCondition.aggregation:

- The referenced AggregationTailoring is owned by an ClassContentConditional that has no condition (invariant class content) **AND**
- The ClassContentConditional that owns the referenced AggregationTailoring and the ClassContentConditional that owns this AggregationCondition are owned by the same ClassTailoring.

⌋*()*

[constr_2616] ReferenceCondition.reference shall reference invariant owned ReferenceTailoring, only ⌈The following conditions need to evaluate to true for ReferenceCondition.reference:

- The referenced ReferenceTailoring is owned by an ClassContentConditional that has no condition (invariant class content) **AND**
- The ClassContentConditional that owns the referenced ReferenceTailoring and the ClassContentConditional that owns this ReferenceCondition are owned by the same ClassTailoring.

⌋*()*

[constr_2617] ClassTailoring.variationRestriction only applicable for «atpVariation» classes ⌈If the tailored meta class is not marked with stereotype «atpVariation» then ClassTailoring.variationRestriction shall not be defined.⌋
()

8.4.1.3 Additional Validation Semantics for Reachable Elements

[TPS_STDT_00163] Validation Semantics of ConcreteClassTailoring ⌈If ConcreteClassTailoring.inScope = true then the restrictions that are defined for this class are evaluated. If the restrictions are violated then a validation message with the specified severity shall be created.⌋*(RS_STDT_00106, RS_STDT_00121)*

[TPS_STDT_00182] Validation Semantics of AbstractClassTailoring ⌈AbstractClassTailorings may be used in order to define restrictions that shall apply for all instances of this class.⌋*(RS_STDT_00106)*

[TPS_STDT_00107] Validation Semantics of global ConcreteClassTailoring.multiplicityRestriction with validationRoot==true ⌈
If the ConcreteClassTailoring is directly aggregated by DataFormatTailoring.classTailoring and ConcreteClassTailoring.validationRoot==true then the MultiplicityRestrictionWithSeverity is evaluated for all instances of the concrete meta class in the context of the complete model (not only the reachable elements). This evaluation can happen before the set of reachable elements is calculated.⌋*(RS_STDT_00106)*

Example 8.2 specifies a ConcreteClassTailoring of a class that is used as validationRoot element. The Validation semantics is: the complete model shall contain exactly one System.

Listing 8.2: Example of Multiplicity Restriction of class that is used as validation root element

```
<DATA-EXCHANGE-POINT>
  <SHORT-NAME>MyExchangePointSystem</SHORT-NAME>
  <!-- -->
  <DATA-FORMAT-TAILORING>
    <CLASS-TAILORINGS>
      <CONCRETE-CLASS-TAILORING>
        <SHORT-NAME>System</SHORT-NAME>
        <DESC>
          <L-2 L="EN">The complete model shall contain exactly one 'System'
              </L-2>
        </DESC>
        <IN-SCOPE>true</IN-SCOPE>
        <MULTIPLICITY-RESTRICTION>
          <SEVERITY>ERROR</SEVERITY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
        </MULTIPLICITY-RESTRICTION>
        <VALIDATION-ROOT>true</VALIDATION-ROOT>
      </CONCRETE-CLASS-TAILORING>
    </CLASS-TAILORINGS>
  </DATA-FORMAT-TAILORING>
</DATA-EXCHANGE-POINT>
```

[TPS_STDT_00108] Validation Semantics of global ConcreteClassTailoring.multiplicityRestriction with validationRoot==false ⌈
If the ConcreteClassTailoring is directly aggregated by DataFormatTailor-

ing.classTailoring and ConcreteClassTailoring.validationRoot==false then the MultiplicityRestrictionWithSeverity is evaluated for each instance of a reference and aggregation individually. I.e. for all reachable instances of references and all reachable instances of aggregations that have a type which is identical to the tailored meta class, the number of referenced / contained objects which are an instance of the tailored meta class is determined and evaluated with respect to the MultiplicityRestrictionWithSeverity.⌋*(RS_STDT_00106)*

Example 8.3 specifies a ConcreteClassTailoring of a class that is not used as validationRoot element.
The validation semantics of the example is: In the set of reachable elements no instances of references to FlexrayFrames are allowed. Additionally, not instances of aggregations that contain FlexrayFrames are allowed.
Note that FlexrayFrames might exist in parts of the model that are not reachable from selected validation root elements.

Listing 8.3: Example of Multiplicity Restriction of class that is not used as validation root element

```
<DATA-EXCHANGE-POINT>
  <SHORT-NAME>MyExchangePointFlexray</SHORT-NAME>
  <!-- -->
  <DATA-FORMAT-TAILORING>
    <CLASS-TAILORINGS>
      <CONCRETE-CLASS-TAILORING>
        <SHORT-NAME>FlexrayFrame</SHORT-NAME>
        <DESC>
          <L-2 L="EN">
  The set of reachable elements shall not contain
  any FlexrayFrames.
  Note that FlexrayFrames might exist in parts of the model that
  are not reachable from the validation root element. </L-2>
        </DESC>
        <IN-SCOPE>true</IN-SCOPE>
        <MULTIPLICITY-RESTRICTION>
          <SEVERITY>ERROR</SEVERITY>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>0</UPPER-MULTIPLICITY>
        </MULTIPLICITY-RESTRICTION>
        <VALIDATION-ROOT>false</VALIDATION-ROOT>
      </CONCRETE-CLASS-TAILORING>
    </CLASS-TAILORINGS>
  </DATA-FORMAT-TAILORING>
</DATA-EXCHANGE-POINT>
```

[TPS_STDT_00113] Validation Semantics of AbstractClassTailoring.multiplicityRestriction ⌈
If the AbstractClassTailoring is directly aggregated by DataFormatTailoring.classTailoring then the MultiplicityRestrictionWithSeverity is evaluated for each instance of a reference and aggregation individually. I.e. for all reachable instances of references and all reachable instances of aggregations which

have a type which is a sub class of the tailored meta class, the number of referenced / contained objects which are an instance of the tailored meta class is determined and evaluated with respect to the MultiplicityRestrictionWithSeverity.⌋ *(RS_STDT_00106)*

8.4.2 Tailoring of Attributes

8.4.2.1 Description

[TPS_STDT_00144] Purpose of AttributeTailoring ⌈The AttributeTailoring specifies if an owned or inherited AUTOSAR Attribute is in scope and defines which restrictions have to be considered.⌋ *(RS_STDT_00106)*

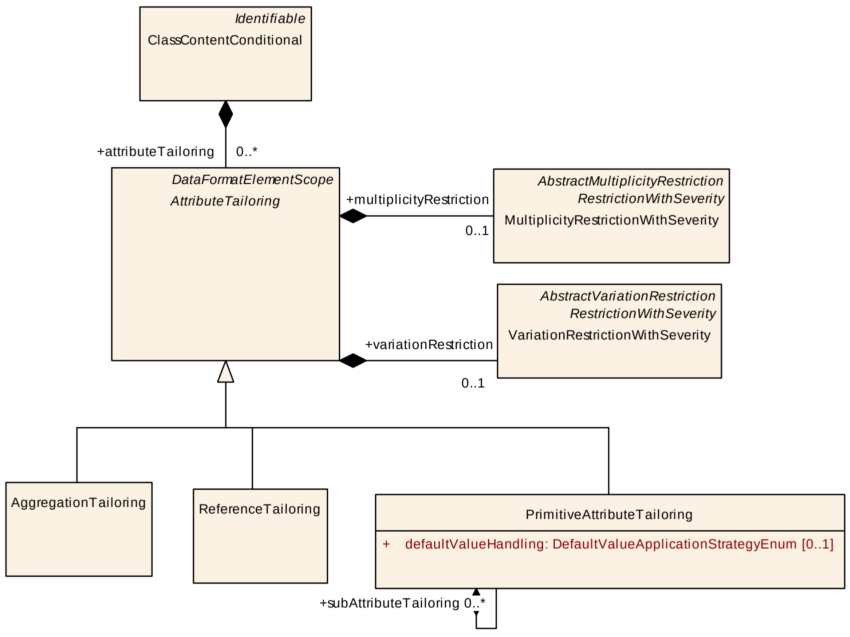

Identifiable
ClassContentConditional
+attributeTailoring
0..*
DataFormatElementScope
AttributeTailoring
+multiplicityRestriction
0..1
AbstractMultiplicityRestriction
RestrictionWithSeverity
MultiplicityRestrictionWithSeverity
AbstractVariationRestriction
RestrictionWithSeverity
VariationRestrictionWithSeverity
+variationRestriction
0..1
AggregationTailoring
ReferenceTailoring
PrimitiveAttributeTailoring
+ defaultValueHandling: DefaultValueApplicationStrategyEnum [0..1]
+subAttributeTailoring 0..*

Figure 8.19: Overview of AttributeTailoring

Class	***AttributeTailoring*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::DataFormatTailoring			
Note	Tailoring of Attributes			
Base	*ARObject*, *DataFormatElementReference*, *DataFormatElementScope*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SpecElementReference*, *SpecElementScope*			
Subclasses	AggregationTailoring, PrimitiveAttributeTailoring, ReferenceTailoring			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
multiplicityRestriction	MultiplicityRestrictionWithSeverity	0..1	aggr	Multiplicity restriction of the attribute **Tags:**xml.sequenceOffset=10

▽

△

Class	***AttributeTailoring*** (abstract)			
variation Restriction	VariationRestrictionWith Severity	0..1	aggr	Restrictions on the usage of variant handling. **Tags:**xml.sequenceOffset=20

Table 8.31: AttributeTailoring

8.4.2.2 Additional Constraints

[constr_2618] ShortName of AttributeTailoring shall match owned or inherited attributes ⌈The shortName shall match the name of an attribute that is owned or inherited by the AUTOSAR meta-class which is identified by the ClassTailoring that owns this AttributeTailoring.⌋*()*

[constr_2619] No AttributeTailoring for Derived or Abstract Attributes ⌈No AttributeTailorings are allowed for Attributes that are marked with stereotypes <<atpDerived>> or <<atpAbstract>>.⌋*()*

See [TPS_GST_00022] and [TPS_GST_00023] in [7] for more details about the stereotypes <<atpDerived>> and <<atpAbstract>>.

[constr_2624] AttributeTailoring.variationRestriction only applicable for «atpVariation» attributes ⌈If the tailored attribute is not marked with stereotype «atpVariation» then AttributeTailoring.variationRestriction shall not be defined.⌋*()*

8.4.2.3 Additional Validation Semantics for Reachable Elements

[TPS_STDT_00159] Semantics of Attribute that is in Scope ⌈If AttributeTailoring.inScope = true then the restrictions defined for the AttributeTailoring apply.⌋*(RS_STDT_00106, RS_STDT_00121)*

[TPS_STDT_00114] MultiplicityRestrictionWithSeverity in the context of ClassTailoring vs. AggregationTailoring/ReferenceTailoring ⌈

- The MultiplicityRestrictionWithSeverity that is aggregated via AggregationTailoring.multiplicityRestriction evaluates the total number of contained elements per instance of the tailored aggregation.
- The MultiplicityRestrictionWithSeverity that is aggregated via ReferenceTailoring.multiplicityRestriction evaluates the total number of referenced elements per instance of the tailored reference.
- The MultiplicityRestrictionWithSeverity that is aggregated via ClassTailoring.multiplicityRestriction evaluates the total number of

aggregated or referenced elements that are an instance of the tailored class per instance of aggregation or reference.

⌋*(RS_STDT_00106)*

See also [TPS_STDT_00108], [TPS_STDT_00112] and example A.2

8.4.3 Tailoring of Primitive Attributes

8.4.3.1 Description

[TPS_STDT_00142] Purpose of `PrimitiveAttributeTailoring` ⌈The `PrimitiveAttributeTailoring` specifies if a owned or inherited `Primitive Attribute` is in scope. Additionally, it defines the handling of AUTOSAR specified default values.⌋*(RS_STDT_00106)*

[TPS_STDT_00143] AUTOSAR Standardized Primitive Attributes of Meta-Class ⌈Within the context of a given AUTOSAR meta-class all inherited and owned primitive attributes that are not marked with `<<atpDerived>>` or `<<atpAbstract>>` may be tailored. The reference to the primitive attribute is established via the name of the primitive attribute.⌋*(RS_STDT_00106)*

Note: In the context of this specification a primitive attribute is a UML property that has a type that is marked with a stereotype `<<primitive>>` or `<<enumeration>>`.

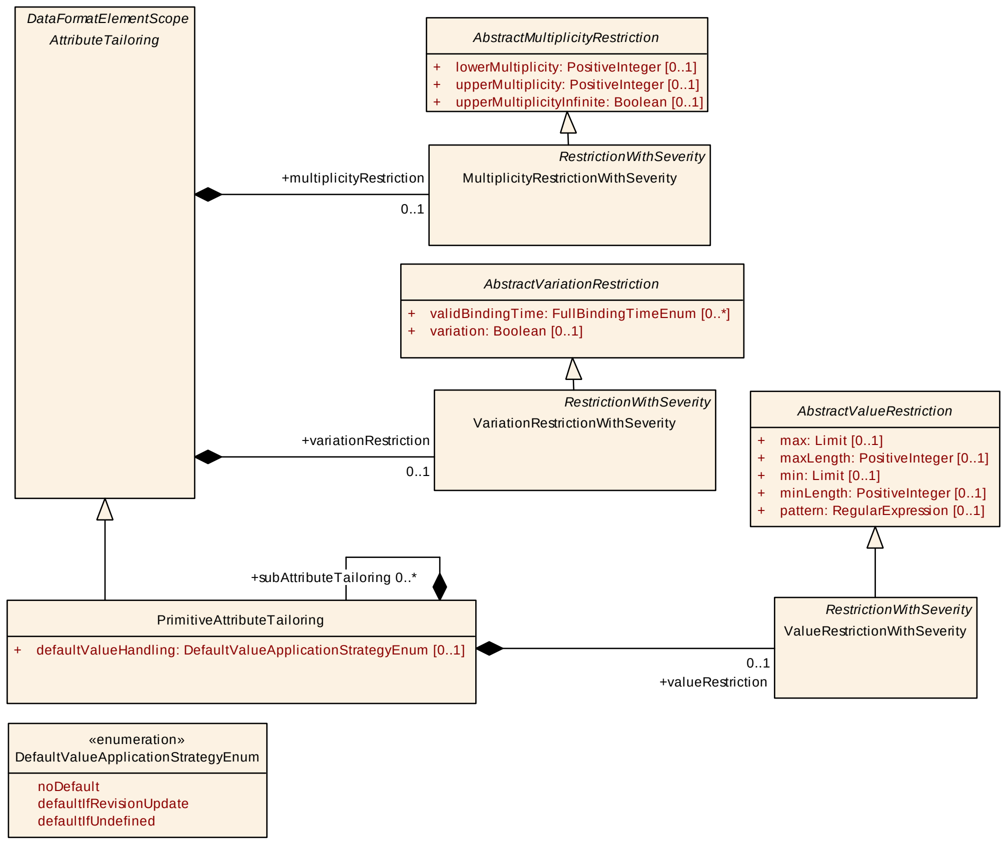

DataFormatElementScope
AttributeTailoring
AbstractMultiplicityRestriction
+ lowerMultiplicity: PositiveInteger [0..1]
+ upperMultiplicity: PositiveInteger [0..1]
+ upperMultiplicityInfinite: Boolean [0..1]
RestrictionWithSeverity
MultiplicityRestrictionWithSeverity
+multiplicityRestriction
0..1
AbstractVariationRestriction
+ validBindingTime: FullBindingTimeEnum [0..*]
+ variation: Boolean [0..1]
RestrictionWithSeverity
VariationRestrictionWithSeverity
+variationRestriction
0..1
AbstractValueRestriction
+ max: Limit [0..1]
+ maxLength: PositiveInteger [0..1]
+ min: Limit [0..1]
+ minLength: PositiveInteger [0..1]
+ pattern: RegularExpression [0..1]
+subAttributeTailoring 0..*
PrimitiveAttributeTailoring
+ defaultValueHandling: DefaultValueApplicationStrategyEnum [0..1]
RestrictionWithSeverity
ValueRestrictionWithSeverity
0..1
+valueRestriction
«enumeration»
DefaultValueApplicationStrategyEnum
noDefault
defaultIfRevisionUpdate
defaultIfUndefined

Figure 8.20: Tailoring of Primitive Attributes

Class	**PrimitiveAttributeTailoring**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	Tailoring of primitive attributes. Primitive attributes are attributes that have a type which is marked by the stereotype <<primitive>> or <<enumeration>>			
Base	*ARObject*, *AttributeTailoring*, *DataFormatElementReference*, *DataFormatElementScope*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SpecElementReference*, *SpecElementScope*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultValue Handling	DefaultValueApplication StrategyEnum	0..1	attr	Specification of how to handle AUTOSAR defined default values.
subAttribute Tailoring	PrimitiveAttribute Tailoring	*	aggr	Tailors the attribute of a <<primitive>> data type.
valueRestriction	ValueRestrictionWith Severity	0..1	aggr	The restriction of the attribute value.

Table 8.32: PrimitiveAttributeTailoring

Enumeration	**DefaultValueApplicationStrategyEnum**
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring

▽

△

Enumeration	**DefaultValueApplicationStrategyEnum**
Note	Enumeration that describes how to handle AUTOSAR defined default values. If the strategy requires application of the AUTOSAR defined default value, then the value shall be added before further validation or processing.
Literal	*Description*
defaultIfRevisionUpdate	If the AUTOSAR model is older than the Baseline of the Data Exchange Point and the older version did not yet support the attribute, then the AUTOSAR defined default value SHALL be applied before further validation or processing. **Tags:**atp.EnumerationLiteralIndex=1
defaultIfUndefined	If the AUTOSAR model does not explicitly specify a value, then the apply the AUTOSAR defined default value before further validation or processing. **Tags:**atp.EnumerationLiteralIndex=2
noDefault	do not apply the AUTOSAR defined default value **Tags:**atp.EnumerationLiteralIndex=0

Table 8.33: DefaultValueApplicationStrategyEnum

8.4.3.2 Additional Constraints

[constr_2620] `shortName` of `PrimitiveAttributeTailoring` shall be a primitive attribute in the referenced Baseline ⌈The `shortName` of `PrimitiveAttributeTailoring` shall match the name of an AUTOSAR specified primitive attribute of the Meta-Class in the referenced Baseline.⌋*()*

8.4.3.3 Additional Validation Semantics for Reachable Elements

No additional validation semantics.

8.4.4 Tailoring of Aggregations

8.4.4.1 Description

[TPS_STDT_00140] Purpose of `AggregationTailoring` ⌈The `AggregationTailoring` specifies if an owned or inherited `Aggregation` is in scope.⌋*(RS_STDT_00106)*

[TPS_STDT_00141] AUTOSAR Standardized Aggregations of Meta-Class ⌈Within the context of a given AUTOSAR meta-class all inherited and owned aggregations that are not marked with `<<atpDerived>>` or `<<atpAbstract>>` may be tailored.⌋*(RS_STDT_00106)*

Note: In the context of this specification an aggregation is a UML property that has a type that is NOT marked with a stereotype «primitive» or «enumeration» and aggregation=AggegationKind::composite.

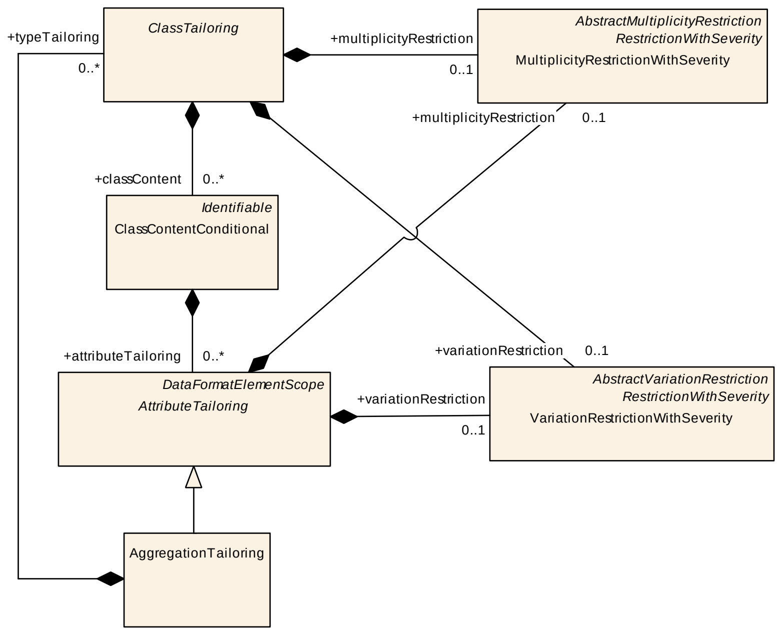


Figure 8.21: Tailoring of Aggregations

Class	**AggregationTailoring**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	Tailoring of aggregations in the AUTOSAR meta-model			
Base	*ARObject*, *AttributeTailoring*, *DataFormatElementReference*, *DataFormatElementScope*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SpecElementReference*, *SpecElementScope*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
typeTailoring	ClassTailoring	*	aggr	Local class tailoring which is applied if the content is contained by this aggregation.

Table 8.34: AggregationTailoring

8.4.4.2 Additional Constraints

[constr_2621] The `shortName` of `AggregationTailoring` shall match the name of an AUTOSAR specified aggregation of the meta-class ⌈The `shortName` of `AggregationTailoring` shall match the name of an AUTOSAR specified aggregation of the meta-class).⌋*()*

8.4.4.3 Additional Validation Semantics for Reachable Elements

[TPS_STDT_00112] Validation Semantics of `ClassTailoring.multiplicityRestriction` in the context of `AggregationTailoring.typeTailoring` ⌈
If the `ClassTailoring` is directly aggregated by `AggregationTailoring.typeTailoring` then the `MultiplicityRestrictionWithSeverity` is evaluated for each instance the tailored aggregation individually. I.e. for each instance of the

tailored aggregation, the number of contained objects which are an instance of the tailored meta class is determined and evaluated with respect to the MultiplicityRestrictionWithSeverity.⌋*(RS_STDT_00106)*

See also example A.3.

8.4.5 Tailoring of References

8.4.5.1 Description

[TPS_STDT_00138] Purpose of ReferenceTailoring ⌈The ReferenceTailoring specifies if an owned or inherited Reference is in scope.⌋*(RS_STDT_00106)*

[TPS_STDT_00139] AUTOSAR Standardized References of Meta-Class ⌈Within the context of a given AUTOSAR Meta-Class all inherited and owned references that are not marked with <<atpDerived>> or <<atpAbstract>> may be tailored. The reference to the reference is established via the name of the reference.⌋*(RS_STDT_00106)*

Note: in the context of this specification a reference is a UML property that has a type that is NOT marked with a stereotype «primitive» or «enumeration» and aggregation=AggregationKind::none.

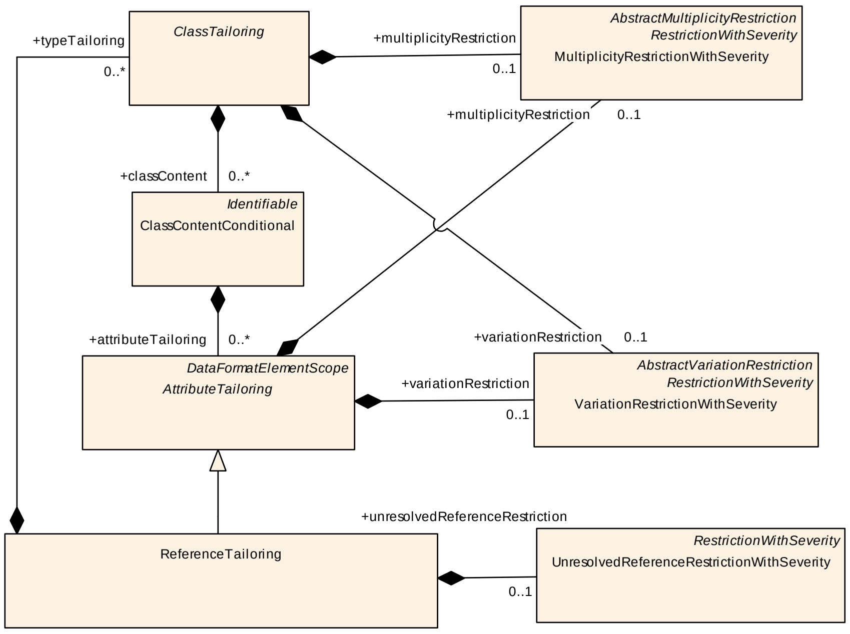


Figure 8.22: Tailoring of References

Class	ReferenceTailoring			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	Tailoring of Non-Containment References.			
Base	*ARObject*, *AttributeTailoring*, *DataFormatElementReference*, *DataFormatElementScope*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SpecElementReference*, *SpecElementScope*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
typeTailoring	ClassTailoring	*	aggr	Local class tailoring for content that is referenced by this reference.
unresolved Reference Restriction	UnresolvedReference RestrictionWithSeverity	0..1	aggr	Specifies the severity of unresolved references.

Table 8.35: ReferenceTailoring

8.4.5.2 Additional Constraints

[constr_2622] The `shortName` of `ReferenceTailoring` shall match the name of an AUTOSAR specified reference of the meta-class ⌈The `shortName` of `ReferenceTailoring` shall match the name of an AUTOSAR specified reference of the meta-class).⌋*()*

8.4.5.3 Additional Validation Semantics for Reachable Elements

[TPS_STDT_00169] Handling of unresolved references ⌈If a reachable object defines an unresolved reference that is referenced by this `ReferenceTailoring` then the `unresolvedReferenceRestriction` specifies the severity of this violation.⌋*(RS_STDT_00121)*

[TPS_STDT_00119] Validation Semantics of `ClassTailoring.multiplicityRestriction` in the context of `ReferenceTailoring.typeTailoring` ⌈
If the `ClassTailoring` is directly aggregated by `ReferenceTailoring.typeTailoring` then the `MultiplicityRestrictionWithSeverity` is evaluated for each instance of the tailored reference individually. I.e. for each instance of the tailored reference, the number of referenced objects which are an instance of the tailored meta class is determined and evaluated with respect to the `MultiplicityRestrictionWithSeverity`.⌋*(RS_STDT_00106)*

See also example A.2.

8.4.6 Tailoring of Constraints

8.4.6.1 Description

[TPS_STDT_00147] Purpose of `ConstraintTailoring` ⌈The `ConstraintTailoring` specifies if the referenced `Constraint` is enabled for this `DataExchangePoint`.⌋*(RS_STDT_00108)*

[TPS_STDT_00111] AUTOSAR Standardized Constraints ⌈Constraints are Specification Elements that have an ID that starts with 'constr_'. A complete list of constraints is available in document AUTOSARModelConstraints [20].⌋*(RS_STDT_00108, RS_STDT_00122)*

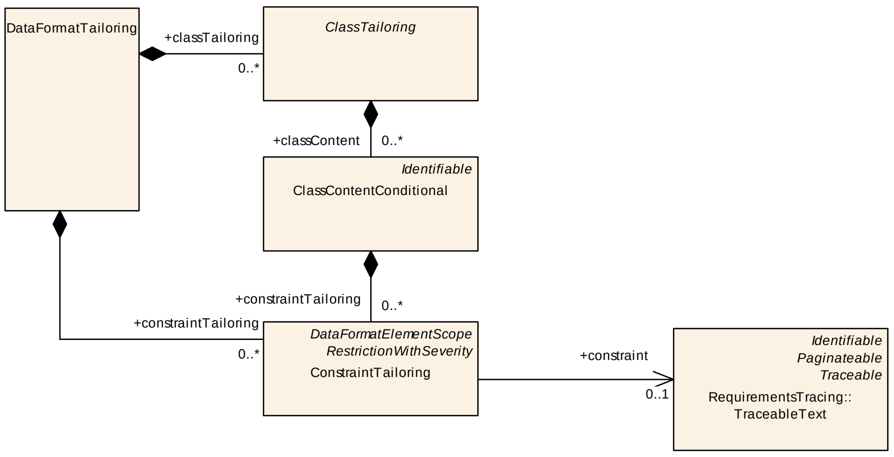


Figure 8.23: Tailoring of Constraints

Class	**ConstraintTailoring**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	Tailoring of constraints. If a constraint is in scope, then the severity defines its Error Severity Level. If it is not in scope, then the constraint is disabled.			
Base	*ARObject*, *DataFormatElementReference*, *DataFormatElementScope*, *Identifiable*, *Multilanguage Referrable*, *Referrable*, *RestrictionWithSeverity*, *SpecElementReference*, *SpecElementScope*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
constraint	TraceableText	0..1	ref	Reference to custom specification of constraint.

Table 8.36: ConstraintTailoring

8.4.6.2 Additional Constraints

none

8.4.6.3 Additional Validation Semantics for Reachable Elements

[TPS_STDT_00164] Semantics of a Constraint that is out of Scope ⌈If ConstraintTailoring.inScope = false then the referenced constraint is disabled.⌋ *(RS_STDT_00108, RS_STDT_00121)*

[TPS_STDT_00165] Semantics of Constraint that is in Scope ⌈If ConstraintTailoring.inScope = true then the referenced constraint is evaluated.⌋*(RS_STDT_00108, RS_STDT_00121)*

[TPS_STDT_00125] Trigger for Evaluation of Constraints ⌈
The context in which a ConstraintTailoring is specified defines the trigger for the evaluation of the constraint:

- If a ConstraintTailoring is aggregated via ClassContentConditional.constraintTailoring then the constraint is only evaluated for reachable instances of the tailored meta class which fullfill the condition.
- If a ConstraintTailoring is aggregated via DataFormatTailoring.constraintTailoring then no explicit hint on instances of classes which trigger the evaluation are provided. It is up to the tool implementer to decide on the correct trigger.

Therefore, the author of a Profile of Data Exchange Point should attach ConstraintTailorings to ClassTailorings whenever this is possible.⌋*(RS_STDT_00108, RS_STDT_00121)*

8.4.7 Tailoring of Special Data Groups

8.4.7.1 Description

[TPS_STDT_00132] Purpose of SdgTailoring ⌈SdgTailoring specifies if a SdgClass (Sdg with a specific gid and structure) may be added to a given MetaClass.⌋*(RS_STDT_00116)*

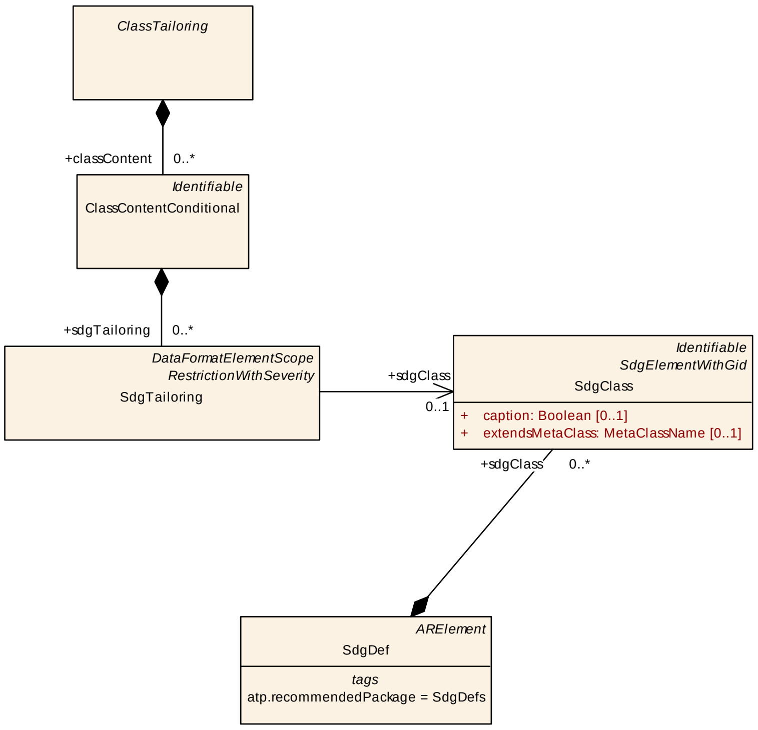


Figure 8.24: Tailoring of Usage of Special Data Groups

Class	**SdgTailoring**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	Describes if the referenced Sdg may be attached to the current class.			
Base	*ARObject*, *DataFormatElementReference*, *DataFormatElementScope*, *Identifiable*, *Multilanguage Referrable*, *Referrable*, *RestrictionWithSeverity*, *SpecElementReference*, *SpecElementScope*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sdgClass	SdgClass	0..1	ref	Specification of the structure of the Special Data Group.

Table 8.37: SdgTailoring

8.4.7.2 Additional Constraints

[constr_2623] Referenced `SdgClass` shall be part of a `SdgDef` that is referenced by the `Baseline` ⌈Referenced `SdgClass` shall be part of a `SdgDef` that is referenced by the `Baseline` of this `Profile of Data Exchange Point`.⌋*()*

8.4.7.3 Additional Validation Semantics for Reachable Elements

[TPS_STDT_00167] Semantics of SdgTailoring that is in scope ⌈If `SdgTailoring.inScope` == true then Sdg structure of instance of the tailored class shall be evaluated against the structure that is specified by the referenced `SdgClass`.⌋*(RS_STDT_00121)*

8.4.8 Description of Special Data Group Definitions

The Special Data Group Definition `SdgDef` specifies the structure of special data group extensions. For a quick overview see figure 8.25. More detailed information is described in [TPS_GST_00374], [TPS_GST_00375], [TPS_GST_00421] and [TPS_-GST_00422] in [7].

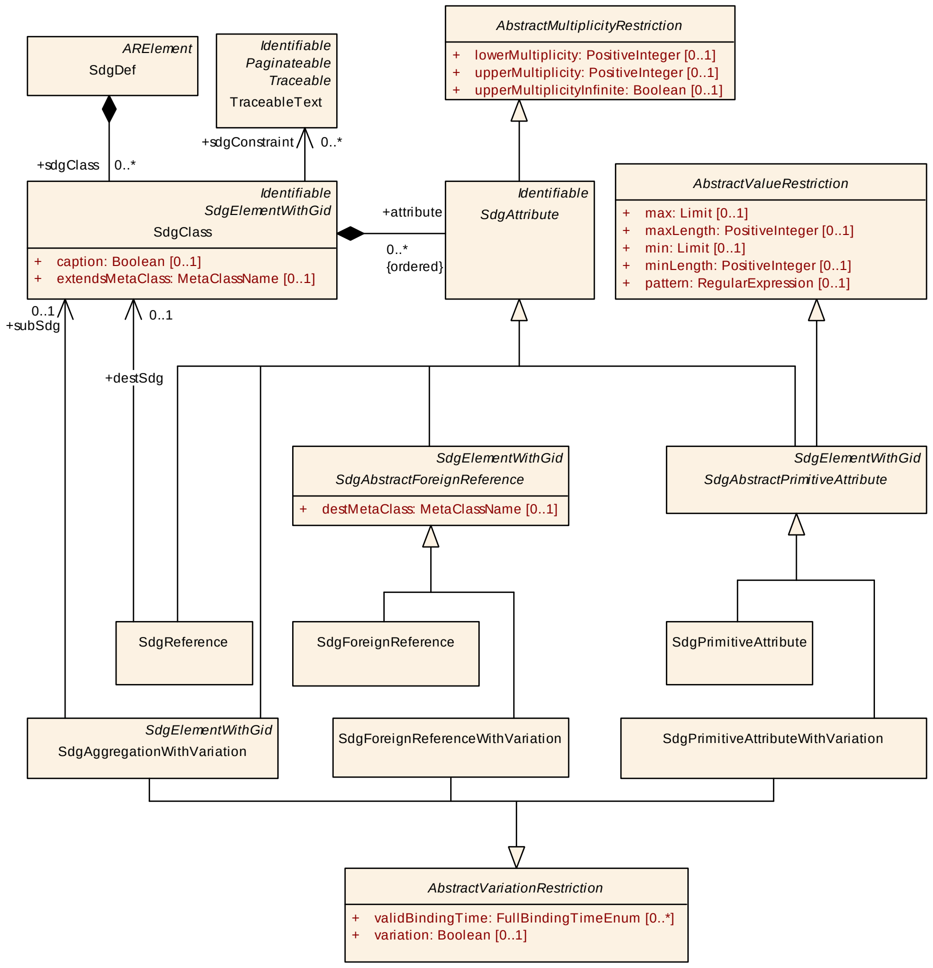

ARElement
SdgDef
Identifiable
Paginateable
Traceable
TraceableText
AbstractMultiplicityRestriction
+ lowerMultiplicity: PositiveInteger [0..1]
+ upperMultiplicity: PositiveInteger [0..1]
+ upperMultiplicityInfinite: Boolean [0..1]
+sdgClass 0..*
+sdgConstraint 0..*
Identifiable
SdgElementWithGid
SdgClass
+ caption: Boolean [0..1]
+ extendsMetaClass: MetaClassName [0..1]
+attribute
0..* {ordered}
Identifiable
SdgAttribute
AbstractValueRestriction
+ max: Limit [0..1]
+ maxLength: PositiveInteger [0..1]
+ min: Limit [0..1]
+ minLength: PositiveInteger [0..1]
+ pattern: RegularExpression [0..1]
0..1 +subSdg
0..1
+destSdg
SdgElementWithGid
SdgAbstractForeignReference
+ destMetaClass: MetaClassName [0..1]
SdgElementWithGid
SdgAbstractPrimitiveAttribute
SdgReference
SdgForeignReference
SdgPrimitiveAttribute
SdgElementWithGid
SdgAggregationWithVariation
SdgForeignReferenceWithVariation
SdgPrimitiveAttributeWithVariation
AbstractVariationRestriction
+ validBindingTime: FullBindingTimeEnum [0..*]
+ variation: Boolean [0..1]

Figure 8.25: Special Data Group Definition

8.4.9 Description of Custom Constraints

8.4.9.1 Description

Custom constraints are documented as `TraceableText` with category==`CONSTRAINT_ITEM` as defined in [TPS_STDT_00059] and [constr_2540]

8.4.9.2 Additional Constraints

none

8.4.9.3 Additional Validation Semantics for Reachable Elements

none

8.5 Default Values in Profiles of Data Exchange Point

This chapter describes rules for default values in Profile of Data Exchange Points which apply if a Profile does not explicitly specify that information. In addition to these rules AUTOSAR provides Baseline Profiles which explicity declare the default values of the latest AUTOSAR releases.

[TPS_STDT_00191] Purpose of Baseline Profile of Data Exchange Point ⌈
A Baseline Profile of Data Exchange Point is a Profile of Data Exchange Point that explicitly models the following default values of ClassTailorings:

- inScope (see [TPS_STDT_00190])
- validationRoot (see [TPS_STDT_00196])
- multiplicityRestriction (see [TPS_STDT_00197])
- variationRestriction (see [TPS_STDT_00200])

Additionally, it specifies the default values of AttributeTailorings:

- inScope (see [TPS_STDT_00195])
- multiplicityRestriction (see [TPS_STDT_00198])
- variationRestriction (see [TPS_STDT_00199])
- defaultValueHandling (see [TPS_STDT_00204])

⌋*(RS_STDT_00105, RS_STDT_00106)*

8.5.1 Default Values in SpecificationScope

The following rules define the default scope of AUTOSAR Specifications and their contained elements.

[TPS_STDT_00192] Default Scope of AUTOSAR Specifications ⌈
If SpecificationDocumentScope.inScope is not explicitly specified for an

AUTOSAR Specification then that AUTOSAR Specification is considered as not relevant for the Data Exchange Point.
Default value of SpecificationDocumentScope.inScope is false.⌋*(RS_STDT_00102, RS_STDT_00103)*

[TPS_STDT_00193] Default Scope of AUTOSAR Specification Elements ⌈
If DocumentElementScope.inScope is not explicitly specified for an element in an AUTOSAR Specification then that element has the same scope as the AUTOSAR Specification that contains it.
Default value of DocumentElementScope.inScope is SpecificationDocumentScope.inScope of the AUTOSAR Specification that contains the element.⌋
(RS_STDT_00102, RS_STDT_00103)

8.5.2 Default Values in DataFormatTailoring

The following rules define default scope and restrictions with respect to the Data Format.

[TPS_STDT_00190] Default Scope of concrete Meta Classes ⌈
If ConcreteClassTailoring.inScope is not explicitly specified for a Meta Class then instances of that Meta Class are considered as not relevant for the Data Exchange Point by default.
Default value of ConcreteClassTailoring.inScope is false.⌋*(RS_STDT_00106)*

[TPS_STDT_00196] Default Validation Root of concrete Meta Classes ⌈
If ConcreteClassTailoring.validationRoot is not explicitly specified for a Meta Class then instances of that Meta Class are no root elements for the validation by default.
Default value of ConcreteClassTailoring.validationRoot is false.⌋*(RS_STDT_00106)*

[TPS_STDT_00197] Default multiplicityRestriction of Meta Classes ⌈
If attributes of ClassTailoring.multiplicityRestriction are not explicitly specified for a Meta Class then the default values as defined in table 8.38 apply by default. Note that the default values depend on the life cycle status of the Meta Class (see also [TPS_GST_00051] in [7]).⌋*(RS_STDT_00106)*

Life cycle status of concrete `Meta Class`	Default value of `ClassTailoring.multiplicityRestriction`				Description
	`.lowerMultiplicity`	`.upperMultiplicity`	`.upperMultiplicityInfinite`	`.severity`	
valid	0	n/a	true	info	No restrictions. Any number of this class may occur.
draft	0	0	n/a	info	Info Message if draft classes are used
obsolete	0	0	n/a	warning	Warning message if obsolete classes are used
preliminary	0	0	n/a	info	Info Message if preliminary classes are used
removed	0	0	n/a	error	Error Message if removed classes are used

Table 8.38: Default Multiplicity Restrictions of Meta Classes

[TPS_STDT_00200] Default `variationRestriction` of `Meta Classes` ⌈
If a meta class is marked with stereotype `<<atpVariation>>` (see [TPS_GST_-00189] in [7]) and attributes of `ClassTailoring.variationRestriction` are not explicitly specified then the default values as defined in table 8.39 apply by default.

Note that the default values depend on the vh.latestBindingTime of the Meta Attribute (see also [TPS_GST_00182] in [7]).⌋*(RS_STDT_00106)*

vh.latestBindingTime of Meta Class	**Default value of ClassTailoring. variationRestriction**		
	.variation	**.validBindingTime**	**.severity**
blueprintDerivationTime	true	blueprintDerivationTime	error
systemDesignTime	true	blueprintDerivationTime, systemDesignTime	error
codeGenerationTime	true	blueprintDerivationTime, systemDesignTime, codeGenerationTime	error
preCompileTime	true	blueprintDerivationTime, systemDesignTime, codeGenerationTime, preCompileTime	error
linkTime	true	blueprintDerivationTime, systemDesignTime, codeGenerationTime, preCompileTime, linkTime	error
postBuild	true	blueprintDerivationTime, systemDesignTime, codeGenerationTime, preCompileTime, linkTime, postBuild	error

Table 8.39: Default Variation Restriction of Meta Classes

[TPS_STDT_00195] Default Scope of Meta Attributes ⌈
If AttributeTailoring.inScope is not explicitly specified for a Meta Attribute then occurrences of that Attribute are considered as not relevant for the Data Exchange Point by default.
Default value of AttributeTailoring.inScope is false.⌋*(RS_STDT_00106)*

[TPS_STDT_00198] Default multiplicityRestriction of Meta Attributes ⌈
If attributes of AttributeTailoring.multiplicityRestriction are not explicitly specified for a Meta Attribute then the default values as defined in table 8.40 apply by default. Note that the default values depend on the life cycle status of the Meta Attribute (see also [TPS_GST_00051] in [7]).⌋*(RS_STDT_00106)*[1]

[1]see [TPS_XMLSPR_00036], [TPS_XMLSPR_00046], [TPS_XMLSPR_00003] in [21]

Life cycle status of Meta Attribute	Default value of AttributeTailoring. multiplicityRestriction				Description
	.lowerMultiplicity	.upperMultiplicity	.upperMultiplicityInfinite	.severity	
valid	(lower)	(upper)	(upperInf)	info	Multiplicity is not restricted. Same rules apply as in AUTOSAR XSD Schema [22]: **if** attribute is tagged with 'xml.enforceMinMultiplicity=true' **then** .lowerMultiplicity = lower multiplicity of attribute as defined in the meta model. **else** .lowerMultiplicity = 0 **if** (upper multiplicity of attribute in the meta model is infinite) **or** (attribute is not marked with xml.attribute=true and it is owned by a class that is marked with stereotypes <<atpMixed>> or <<atpMixed>>) **then** .upperMultiplicity = n/a **and** .upperMultiplicityInfinite = true **else** .upperMultiplicity = upper multiplicity as defined in meta model **and** .upperMultiplicityInfinite = n/a.
draft	0	0	n/a	info	Info Message if draft attribute is used
obsolete	0	0	n/a	warning	Warning message if obsolete attribute is used
preliminary	0	0	n/a	info	Info Message if preliminary attribute is used
removed	0	0	n/a	error	Error Message if removed attribute is used

Table 8.40: Default Multiplicity Restrictions of Meta Attributes

[TPS_STDT_00199] Default variationRestriction of Meta Attributes ⌈
If attributes of AttributeTailoring.variationRestriction are not explicitly specified for a Meta Attribute then the default values as defined in table 8.41 apply by default. Note that the default values depend on the vh.latestBindingTime of the Meta Attribute (see also [TPS_GST_00182] in [7]).⌋*(RS_STDT_00106)*

vh.latestBindingTime of Meta Attribute	Default value of AttributeTailoring. variationRestriction		
	.variation	.validBindingTime	.severity
blueprintDerivationTime	true	{blueprintDerivationTime}	error
systemDesignTime	true	{blueprintDerivationTime, systemDesignTime}	error
codeGenerationTime	true	{blueprintDerivationTime, systemDesignTime, codeGenerationTime}	error
preCompileTime	true	{blueprintDerivationTime, systemDesignTime, codeGenerationTime, preCompileTime}	error
linkTime	true	{blueprintDerivationTime, systemDesignTime, codeGenerationTime, preCompileTime, linkTime}	error
postBuild	true	{blueprintDerivationTime, systemDesignTime, codeGenerationTime, preCompileTime, linkTime, postBuild}	error

Table 8.41: Default Variation Restriction of Meta Attributes

[TPS_STDT_00203] Default PrimitiveAttributeTailoring.valueRestriction ⌈
If attributes of PrimitiveAttributeTailoring.valueRestriction are not explicitly specified for a primitive Attribute then the default values as defined in table 8.42 apply by default. In other words: By default the ValueRestriction of a primitive

attribute semantically equals the declaration of its primitive type in the AUTOSAR meta model.⌋*(RS_STDT_00113)*

Attribute	Default Value
pattern	**if** the type of the primitive attribute specifies `xml.xsd.pattern` **then** value of `xml.xsd.pattern` **else** .*
max	**if** the type of the primitive attribute specifies `xml.xsd.maxInclusive` that is not +INF **then** value of `xml.xsd.maxInclusive` with `intervalType`=closed **else if** the type of the primitive attribute specifies `xml.xsd.maxExclusive` that is not +INF **then** value of `xml.xsd.maxExclusive` with `intervalType`=open **else** +INF with `intervalType`=infinite
min	**if** the type of the primitive attribute specifies `xml.xsd.minInclusive` that is not -INF **then** value of `xml.xsd.minInclusive` with `intervalType`=closed **else if** the type of the primitive attribute specifies `xml.xsd.minExclusive` that is not -INF **then** value of `xml.xsd.minExclusive` with `intervalType`=open **else** -INF with `intervalType`=infinite
maxLength	**if** the type of the primitive attribute specifies `xml.xsd.maxLength` **then** value of `xml.xsd.maxLength` **else** +INF
minLength	**if** the type of the primitive attribute specifies `xml.xsd.minLength` **then** value of `xml.xsd.minLength` **else** 0
severity	error

Table 8.42: Default Values of Value Restrictions

[TPS_STDT_00204] Default `PrimitiveAttributeTailoring.defaultValueHandling` ⌈
If attribute `PrimitiveAttributeTailoring.defaultValueHandling` is not explicitly specified for a primitive `Attribute` then the no default values are applied by default.
Default value of `PrimitiveAttributeTailoring.defaultValueHandling` is `noDefault`.⌋*(RS_STDT_00111)*

[TPS_STDT_00207] Default `ReferenceTailoring.unresolvedReferenceRestriction` ⌈

If ReferenceTailoring.unresolvedReferenceRestriction or ReferenceTailoring.unresolvedReferenceRestriction.severity are not defined then the default value of ReferenceTailoring.unresolvedReferenceRestriction.severity is error.⌋*(RS_STDT_00111)*

8.6 Compatibility

This section describes the meaning of compatibility of Profiles of Data Exchange Points and defines generic rules for evaluating the compatibility of profiles. Compatibility is a measure for the level of interoperability risk. When evaluating the compatibility of a producer's and consumer's profile, a workflow is assumed where both, the producer and consumer, validate an artifact against their individual profiles. A compatibility analysis shall answer the question "What interoperability issues might arise if an Autosar Model passes the producer's validation and is imported on the consumer's side?" Problems can occur, for example, if the consumer's profile is more restrictive than the producer's profile. On the other hand, an issue on the producer side may not necessarily result in a problem on the consumer side, for example, if only a subset of the data is consumed.

- Note, the compatibility of Profiles of Data Exchange Points does not guarantee the absence of any interoperability issues. This compatibility however is an indicator for a low overall interoperability risk.
- On the other hand, the incompatibility of Profiles of Data Exchange Points does not necessarily imply the presence of interoperability issues. This incompatibility however is an indicator for a high interoperability risk.

[TPS_STDT_00110] Identification of Potential Interoperability Issues ⌈
Potential interoperability risks are identified using the following iterative approach. The approach focuses on early identification of incompatibilities with a high risk of producing interoperability issues. The following steps refer to the effective representation of the profiles as specified in [TPS_STDT_00106].

1. Compare the high level descriptions in DataExchangePoint.longName, DataExchangePoint.desc and DataExchangePoint.introduction in order to understand if the Profile of Data Exchange Points actually fit to each other with respect to the intended step in the Autosar Methodology. This is a fully manual step. If the profiles relate to completely different steps in the methodology, then an expert discussion about the methodological integration is required.
2. Compare the Baselines according to the rules defined in [TPS_STDT_00183]. If the Baselines are compatible then we can continue with the following steps. Otherwise, special caution is required, since specification items, constraints, meta model elements, etc. might have been added, removed or changed in the Baselines. Thus, in addition to the descriptions in the following steps, it

is required to figure out if the changes between the baselines actually affect the compared Profile of Data Exchange Points.

3. Identify matching SpecElementReference elements. The key for matching of standardized Specification Elements is the relative shortName path that is relative to the DataExchangePoint. The key for matching custom Specification Elements is the absolute shortName path of the referenced custom element.
   (a) If there is no matching SpecElementReference in the other profile, then expert discussion is needed. This can for instance happen if profiles with incompatible Baselines are compared or if custom extensions are used.
   (b) Otherwise: continue with next steps.
4. Identify not relevant elements: Elements that are not relevant (SpecElementScope.inScope==false) in both profiles result in low risk for interoperability issues and are ignored in further analysis.
5. Analyze the SpecificationScope (see also section 8.6.2):
   (a) Compare SpecificationDocumentScope.inScope as defined in [TPS_STDT_00128] and [TPS_STDT_00160]. Experts should discuss the identified incompatibilities.
6. Analyze DataFormatTailoring (see also section 8.6.3):
   (a) Compare ConcreteClassTailoring.inScope as defined in [TPS_STDT_00101]. Experts should discuss the identified incompatibilities. Suspect ConcreteClassTailorings are analyzed in the next steps.
   (b) Compare ConcreteClassTailoring.validationRoots. Expert discussion is required if the values are not identical.
   (c) Compare ClassTailoring.multiplicityRestriction [TPS_STDT_00210] and ClassTailoring.variationRestrictions [TPS_STDT_00201]. Experts should discuss the identified incompatibilities. Incompatible restrictions with severity==error in the consumer's profile should be handled first, followed by restrictions with severity==warning and restrictions with severity==info.
   (d) Compare ClassTailoring.classContent [TPS_STDT_00135]. Two ClassContentConditional match if the condition is equal. Experts should discuss if there is no match in the other profile or if incompatibilities are identified.
   (e) Compare ClassContentConditional.attributeTailorings [TPS_STDT_00131][TPS_STDT_00133][TPS_STDT_00134], ClassContentConditional.constraintTailorings [TPS_STDT_00209] and ClassContentConditional.sdgTailorings [TPS_STDT_00209].

Experts should discuss the identified incompatibilities. Incompatible restrictions with severity==error in the consumer's profile should be handled first, followed by restrictions with severity==warning and restrictions with severity==info.

⌋*(RS_STDT_00118)*

8.6.1 Compatibility of Baseline

[TPS_STDT_00183] Compatibility of Baselines ⌈
Baselines are compatible if the following criteria are fulfilled

- standardRevisions specify the same revisions of Autosar standards[2] and
- customSpecifications refer to the same set of custom Documentations and
- customSdgDefs refer to the same set of SdgDefs.

Otherwise the Baselines are not compatible and discussion by engineers is required.⌋*(RS_STDT_00118)*

8.6.2 Compatibility of SpecificationScope

[TPS_STDT_00128] Compatibility of SpecificationDocumentScopes ⌈
SpecificationDocumentScopes are incompatible if the attribute inScope has different values. Further analysis of the contained DocumentElementScopes is required if the specification is relevant in both profiles (inScope==true). See also table 8.43.⌋*(RS_STDT_00118)*

	SpecificationDocumentScope.inScope of consumer x = compatible, - = incompatible, ? = suspect	
SpecificationDocumentScope.inScope of producer	**false**	**true**
false	x	-
true	-	? suspect, further analysis of contained DocumentElementScopes required => see [TPS_STDT_00160]

Table 8.43: Compatibility of SpecificationDocumentScope

[TPS_STDT_00160] Compatibility of DocumentElementScopes ⌈
DocumentElementScopes are compatible if the attribute inScope has the same value. Otherwise it is incompatible. See also table 8.44.⌋*(RS_STDT_00118)*

[2] Different standardRevisions do not automatically result in problems with respect to tool interoperability. Especially, in case the Data Exchange Point relates to parts of the standard that have not changed between the revisions

	DocumentElementScope.inScope of consumer x = compatible, - = incompatible	
DocumentElementScope.inScope of producer	false	true
false	x	-
true	-	x

Table 8.44: Compatibility of DocumentElementScope

8.6.3 Compatibility of DataFormatTailoring

[TPS_STDT_00101] Compatibility of ConcreteClassTailorings ⌈
ConcreteClassTailorings are compatible if both the consumer and producer consider the related class as not relevant. (i.e. ConcreteClassTailoring.inScope==false). They are incompatible if the values of ConcreteClassTailoring.inScope are different. Further analysis is required if both the consumer and producer consider the related class as relevant. (i.e. ConcreteClassTailoring.inScope=true). See also table 8.45.⌋*(RS_STDT_00118)*

	ConcreteClassTailoring.inScope of consumer x = compatible, - = incompatible, ? = suspect	
ConcreteClassTailoring.inScope of producer	false	true
false	x	-
true	-	? Further analysis required

Table 8.45: Compatibility of ConcreteClassTailoring

[TPS_STDT_00135] Compatibility of ClassContentConditional ⌈
Two ClassContentConditionals are considered to be incompatible if

- condition is not equivalent **OR**
- elements in attributeTailoring do not match **OR**
- elements in constraintTailoring do not match **OR**
- elements in sdgTailoring do not match

Otherwise further analysis is required. See also [TPS_STDT_00131], [TPS_STDT_00133], [TPS_STDT_00134], [TPS_STDT_00209] and [TPS_STDT_00208].⌋*(RS_STDT_00118)*

[TPS_STDT_00136] Compatibility of AttributeTailoring ⌈
Two AttributeTailorings are considered to be incompatible if

- multiplicityRestriction is incompatible [TPS_STDT_00210] **OR**
- variationRestriction is incompatible [TPS_STDT_00201]

⌋*(RS_STDT_00118)*

[TPS_STDT_00131] Compatibility of AggregationTailoring ⌈
Two AggregationTailorings are incompatible

- if they are incompatible according to [TPS_STDT_00136] **OR**
- if the elements in typeTailoring are incompatible.

If no incompatibilities were identified then it is considered as compatible.⌋*(RS_STDT_-00118)*

[TPS_STDT_00133] Compatibility of ReferenceTailoring ⌈
Two ReferenceTailorings are incompatible

- if they are incompatible according to [TPS_STDT_00136] **OR**
- if the elements in typeTailoring are incompatible **OR**
- if unresolvedReferenceRestriction is incompatible

If no incompatibilities were identified then it is considered as compatible.⌋*(RS_STDT_-00118)*

[TPS_STDT_00134] Compatibility of PrimitiveAttributeTailoring ⌈
Two PrimitiveAttributeTailorings are incompatible

- if they are incompatible according to [TPS_STDT_00136] **OR**
- if defaultValueHandling is not the same **OR**
- if valueRestriction [TPS_STDT_00205] is incompatible

⌋*(RS_STDT_00118)*

[TPS_STDT_00209] Compatibility of SdgTailorings ⌈
Two SdgTailorings are considered to be compatible if their sdgClass reference points to the same SdgClass.⌋*(RS_STDT_00118)*

Note: This definition of compatibility does not cover the case, where two SdgClass definitions exist at different locations, but boil down to the equivalent SdgClasses. This is accepted for simplicity of validation.

[TPS_STDT_00208] Compatibility of ConstraintTailorings ⌈
An interoperability risk exists if the severity of a constraint in the producer's profile is less than the severity in the consumer's profile. Another interoperability risk exists, if a custom constraint is referenced and the textual description is not identical in the producer's and consumer's profile. In both cases, expert discussion is needed.⌋

(RS_STDT_00118)

[TPS_STDT_00210] Compatibility of MultiplicityRestrictionWithSeverity ⌈
MultiplicityRestrictionWithSeveritys are compatible if the range that is specified by lowerMultiplicity and upperMultiplicity/upperMultiplicityInfinite of the producer is fully covered by the range that is specified by the consumer. See also table 8.46.⌋*(RS_STDT_00118)*

	MultiplicityRestrictionWithSeverity of Consumer x = compatible, - = incompatible				
MultiplicityRestrictionWithSeverity of Producer	**0..0**	**0..1**	**0..***	**1..1**	**1..***
0..0	x	x	x	-	-
0..1	-	x	x	-	-
0..*	-	-	x	-	-
1..1	-	x	x	x	x
1..*	-	-	x	-	x

Table 8.46: Compatibility of Multiplicity Restrictions

[TPS_STDT_00201] Compatibility of VariationRestrictionWithSeverity.variation ⌈
When evaluating the compatibility of VariationRestrictionWithSeveritys the variation attribute at the producer's and the consumer's side are compared in a first step. Table 8.47 illustrates the outcome of this evaluation w.r.t. compatibility.⌋*(RS_STDT_00118)*

	Value of VariationRestrictionWithSeverity.variation of the Consumer	
Value of VariationRestrictionWithSeverity.variation of the Producer	**false**	**true**
false	compatible	compatible
true	incompatible	further evaluation of the VariationRestrictionWithSeverity.validBindingTime attribute is required. See [TPS_STDT_00202]

Table 8.47: Compatibility of Variation Restrictions

[TPS_STDT_00202] Compatibility of VariationRestrictionWithSeverity.validBindingTime ⌈
In case the value of VariationRestrictionWithSeverity.variation is true at both the producer's and the consumer's side, further evaluation of the VariationRestrictionWithSeverity.validBindingTime attribute is required. The validBindingTime attributes at the producer's and the consumer's side are considered compatible if the the set of valid binding times of the producer is a subset of the set of valid binding times of the consumer. Otherwise the validBindingTime attributes

at the producer's and the consumer's side are considered incompatible.⌋*(RS_STDT_-00118)*

[TPS_STDT_00205] Compatibility of ValueRestrictionWithSeverity ⌈
The compatibility of ValueRestrictionWithSeveritys is calculated using the following algorithm:

1. **if** min of the producer >= min of the consumer **then**
   min is compatible. Continue with next attribute. **else**
   ValueRestrictionWithSeverity is incompatible. Stop comparison.
2. **if** max of the producer <= max of the consumer **then**
   max is compatible. Continue with next attribute. **else**
   ValueRestrictionWithSeverity is incompatible. Stop comparison.
3. **if** minLength of the producer >= minLength of the consumer **then**
   minLength is compatible. Continue with next attribute. **else**
   ValueRestrictionWithSeverity is incompatible. Stop comparison.
4. **if** maxLength of the producer <= maxLength of the consumer **then**
   maxLength is compatible. Continue with next attribute. **else**
   ValueRestrictionWithSeverity is incompatible. Stop comparison.
5. **if** any match to the regular expression defined in the pattern attribute at the producer's side also yields a match to the regular expression defined in the pattern attribute at the consumer's side[3] **then**
   pattern is compatible. ValueRestrictionWithSeverity is compatible
   else
   ValueRestrictionWithSeverity is incompatible. Stop comparison.

⌋*(RS_STDT_00118)*

[TPS_STDT_00206] Compatibility of UnresolvedReferenceRestrictionWithSeverity ⌈
For an existing reference attribute, the UnresolvedReferenceRestrictionWithSeverity defines the severity, if the given reference path cannot be resolved.
An interoperability problem exists if the UnresolvedReferenceRestrictionWithSeverity.severity on the producer side is lower than the UnresolvedReferenceRestrictionWithSeverity.severity on the consumer side.⌋*(RS_STDT_00118)*

Note: Unresolved references may happen by mistake or intentionally. For example, unresolved references may be tolerated by the consumer, if the data is not needed for the intended methodology step.

[3]Note that this basically boils down to computing the intersection to the two languages describe by the two regular expressions and checking whether this intersection in equal to the language described by the regular expression at the producer's side. Since this is a rather complex check it is permissible that a validating tool simply performs a string comparison of the two pattern attributes and treats them as incompatible if the two strings are not equal

A Example Profiles of Data Exchange Points

A.1 Referencing Specification Elements

Example A.1 shows examples of references to standardized and custom specification elements.

Listing A.1: Referencing Specification Elements

```
<DATA-EXCHANGE-POINT>
  <SHORT-NAME>ExampleDataExchangePointWithCustomExtensions</SHORT-NAME>
  <REFERENCED-BASELINE>


    <STANDARD-REVISIONS>
      <STANDARD-REVISION>CP R4.2.2</STANDARD-REVISION>
    </STANDARD-REVISIONS>
      <CUSTOM-SPECIFICATION-REFS>
      <CUSTOM-SPECIFICATION-REF DEST="DOCUMENTATION">
         CustomSpecificationOfOS</CUSTOM-SPECIFICATION-REF>
      <CUSTOM-SPECIFICATION-REF DEST="DOCUMENTATION">
         CustomDataFormatExtensions</CUSTOM-SPECIFICATION-REF>
    </CUSTOM-SPECIFICATION-REFS>
      <CUSTOM-SDG-DEF-REFS>
      <CUSTOM-SDG-DEF-REF DEST="SDG-DEF">SafetyExtensionSdgDef</CUSTOM-SDG-
         DEF-REF>
    </CUSTOM-SDG-DEF-REFS>
  </REFERENCED-BASELINE>
  <SPECIFICATION-SCOPE>
    <SPECIFICATION-DOCUMENT-SCOPES>
      <SPECIFICATION-DOCUMENT-SCOPE>
        <SHORT-NAME>Methodology</SHORT-NAME>
        <IN-SCOPE>true</IN-SCOPE>
        <DOCUMENT-ELEMENT-SCOPES>
          <DOCUMENT-ELEMENT-SCOPE>
            <SHORT-NAME>Topology</SHORT-NAME>
            <DESC>
              <L-2 L="EN">Reference to STANDARDIZED element via shortName</
                 L-2>
            </DESC>
            <IN-SCOPE>true</IN-SCOPE>
          </DOCUMENT-ELEMENT-SCOPE>
          <DOCUMENT-ELEMENT-SCOPE>
            <SHORT-NAME>ECU_System_Description</SHORT-NAME>
            <DESC>
              <L-2 L="EN">Reference to STANDARDIZED deliverable via
                 alternativeName. The name of the deliverable contains
                 spaces and thus it is required to use the alternativeName<
                 /L-2>
            </DESC>
            <ALTERNATIVE-NAME>ECU System Description</ALTERNATIVE-NAME>
            <IN-SCOPE>true</IN-SCOPE>
          </DOCUMENT-ELEMENT-SCOPE>
        </DOCUMENT-ELEMENT-SCOPES>
      </SPECIFICATION-DOCUMENT-SCOPE>
      <SPECIFICATION-DOCUMENT-SCOPE>
        <SHORT-NAME>Specification_of_Operating_System</SHORT-NAME>
        <DESC>
          <L-2 L="EN">Reference to STANDARDIZED Specification via
              alternative name that represents the title of the
              specification</L-2>
        </DESC>
        <ALTERNATIVE-NAME>Specification of Operating System</ALTERNATIVE-
           NAME>
        <IN-SCOPE>true</IN-SCOPE>
        <DOCUMENT-ELEMENT-SCOPES>
          <DOCUMENT-ELEMENT-SCOPE>
            <SHORT-NAME>SRS_Os_11005</SHORT-NAME>
            <DESC>
              <L-2 L="EN">Reference to STANDARDIZED requirement via
                 shortName</L-2>
            </DESC>
            <IN-SCOPE>true</IN-SCOPE>
          </DOCUMENT-ELEMENT-SCOPE>
        </DOCUMENT-ELEMENT-SCOPES>
      </SPECIFICATION-DOCUMENT-SCOPE>
      <SPECIFICATION-DOCUMENT-SCOPE>
        <SHORT-NAME>CustomSpecificationOfOsScope</SHORT-NAME>
        <IN-SCOPE>true</IN-SCOPE>
        <CUSTOM-DOCUMENTATION-REF DEST="DOCUMENTATION">
           CustomSpecificationOfOS</CUSTOM-DOCUMENTATION-REF>
        <DOCUMENT-ELEMENT-SCOPES>
          <DOCUMENT-ELEMENT-SCOPE>
            <SHORT-NAME>Custom_SRS_Os_00001_Scope</SHORT-NAME>
            <DESC>
              <L-2 L="EN">Reference to CUSTOM requirement via shortName
                 path</L-2>
            </DESC>
            <IN-SCOPE>true</IN-SCOPE>
            <CUSTOM-DOCUMENT-ELEMENT-REF DEST="STRUCTURED-REQ">
                CustomSpecificationOfOS/FunctionalExtensions/
                Custom_SRS_Os_00001</CUSTOM-DOCUMENT-ELEMENT-REF>
          </DOCUMENT-ELEMENT-SCOPE>
        </DOCUMENT-ELEMENT-SCOPES>
      </SPECIFICATION-DOCUMENT-SCOPE>
      <SPECIFICATION-DOCUMENT-SCOPE>
        <SHORT-NAME>Software_Component_Template</SHORT-NAME>
        <ALTERNATIVE-NAME>Software Component Template</ALTERNATIVE-NAME>
        <DOCUMENT-ELEMENT-SCOPES>
          <DOCUMENT-ELEMENT-SCOPE>
            <SHORT-NAME>TPS_SWCT_01251</SHORT-NAME>
            <DESC>
              <L-2 L="EN">Reference to STANDARDIZED specItem via shortName<
                 /L-2>
            </DESC>
            <IN-SCOPE>true</IN-SCOPE>
          </DOCUMENT-ELEMENT-SCOPE>
        </DOCUMENT-ELEMENT-SCOPES>
      </SPECIFICATION-DOCUMENT-SCOPE>
    </SPECIFICATION-DOCUMENT-SCOPES>
  </SPECIFICATION-SCOPE>
```

```
    <DATA-FORMAT-TAILORING>
      <CLASS-TAILORINGS>
        <CONCRETE-CLASS-TAILORING>
          <SHORT-NAME>StructuredReq</SHORT-NAME>
          <IN-SCOPE>true</IN-SCOPE>
          <CLASS-CONTENTS>
            <CLASS-CONTENT-CONDITIONAL>
              <SHORT-NAME>Invariant</SHORT-NAME>
              <SDG-TAILORINGS>
                <SDG-TAILORING>
                  <SHORT-NAME>SafetyExtension</SHORT-NAME>
                  <SEVERITY>WARNING</SEVERITY>
                  <SDG-CLASS-REF DEST="SDG-CLASS">SafetyExtensionSdgDef/
                      SafetyRequirement</SDG-CLASS-REF>
                </SDG-TAILORING>
              </SDG-TAILORINGS>
            </CLASS-CONTENT-CONDITIONAL>
          </CLASS-CONTENTS>
        </CONCRETE-CLASS-TAILORING>
      </CLASS-TAILORINGS>
      <CONSTRAINT-TAILORINGS>
        <CONSTRAINT-TAILORING>
          <SHORT-NAME>constr_2508</SHORT-NAME>
          <DESC>
            <L-2 L="EN">Reference to STANDARDIZED constraint via shortName</L
                -2>
          </DESC>
          <IN-SCOPE>true</IN-SCOPE>
        </CONSTRAINT-TAILORING>
        <CONSTRAINT-TAILORING>
          <SHORT-NAME>CUSTOM_constr_0001Tailoring</SHORT-NAME>
          <DESC>
            <L-2 L="EN">Reference to CUSTOM constraint via shortName path</L
                -2>
          </DESC>
          <IN-SCOPE>true</IN-SCOPE>
          <SEVERITY>ERROR</SEVERITY>
          <CONSTRAINT-REF DEST="TRACEABLE-TEXT">CustomDataFormatExtensions/
              CustomConstraints/CUSTOM_constr_0001</CONSTRAINT-REF>
        </CONSTRAINT-TAILORING>
      </CONSTRAINT-TAILORINGS>
    </DATA-FORMAT-TAILORING>
  </DATA-EXCHANGE-POINT>
  <DOCUMENTATION>
    <SHORT-NAME>CustomSpecificationOfOS</SHORT-NAME>
    <DOCUMENTATION-CONTENT>
      <CHAPTER>
        <SHORT-NAME>FunctionalExtensions</SHORT-NAME>
        <STRUCTURED-REQ>
          <SHORT-NAME>Custom_SRS_Os_00001</SHORT-NAME>
          <DESCRIPTION>
            <P>
              <L-1 L="EN">The description of the custom requirement</L-1>
            </P>
          </DESCRIPTION>
          <RATIONALE>
            <P>
              <L-1 L="EN">The rationale of the custom requirement</L-1>
            </P>
          </RATIONALE>
        </STRUCTURED-REQ>
      </CHAPTER>
    </DOCUMENTATION-CONTENT>
  </DOCUMENTATION>
  <DOCUMENTATION>
    <SHORT-NAME>CustomDataFormatExtensions</SHORT-NAME>
    <DOCUMENTATION-CONTENT>
      <CHAPTER>
        <SHORT-NAME>CustomConstraints</SHORT-NAME>
        <TRACE>
          <SHORT-NAME>CUSTOM_constr_0001</SHORT-NAME>
          <CATEGORY>CONSTRAINT_ITEM</CATEGORY>
          <P>
            <L-1 L="EN">Description of the custom constraint</L-1>
          </P>
        </TRACE>
      </CHAPTER>
    </DOCUMENTATION-CONTENT>
  </DOCUMENTATION>
  <SDG-DEF>
    <SHORT-NAME>SafetyExtensionSdgDef</SHORT-NAME>
    <DESC>
      <L-2 L="EN">Sdgs used for safety extensions</L-2>
    </DESC>
    <SDG-CLASSES>
      <SDG-CLASS>
        <SHORT-NAME>SafetyRequirement</SHORT-NAME>
        <DESC>
          <L-2 L="EN">[TPS_SAFEX_00104] Status attribute</L-2>
        </DESC>
        <GID>SAFEX</GID>
        <EXTENDS-META-CLASS>StructuredReq</EXTENDS-META-CLASS>
        <ATTRIBUTES>
          <SDG-PRIMITIVE-ATTRIBUTE>
            <SHORT-NAME>asil</SHORT-NAME>
            <DESC>
              <L-2 L="EN">[TPS_SAFEX_00201] ASIL attribute of safety
                  requirements</L-2>
            </DESC>
            <GID>ASIL</GID>
            <PATTERN>QM|A|B|C|D|QM(A)|QM(B)|QM(C)|QM(D)|A(B)|A\(C\)|A(D)|B(B)
                |B(C)|B(D)|C(C)|C(D)|D(D)</PATTERN>
          </SDG-PRIMITIVE-ATTRIBUTE>
          <SDG-PRIMITIVE-ATTRIBUTE>
            <SHORT-NAME>status</SHORT-NAME>
            <DESC>
              <L-2 L="EN">[TPS_SAFEX_00104] Status attribute</L-2>
            </DESC>
            <GID>STATUS</GID>
          </SDG-PRIMITIVE-ATTRIBUTE>
        </ATTRIBUTES>
      </SDG-CLASS>
```

A.2 Class Tailoring With MultiplicityRestrictions and ValueRestrictions

Example A.2 specifies a ClassTailoring that

- Requires exactly one instance of the meta class System in the complete model.
- This instance of System is a validationRoot element for determining the set of reachable elements.
- The value of the attribute category of that System shall be "ECU_SYSTEM_DESCRIPTION"
- The number of elements that are referenced by System.fibexElement is restricted to 100.
- Exactly one instance of EcuInstance shall be referenced in the role System.fibexElement
- Exactly one instance of CanCluster shall be referenced in the role System.fibexElement

Listing A.2: Example of Class Tailoring With MultiplicityRestrictions and ValueRestriction

```
<CONCRETE-CLASS-TAILORING>
  <SHORT-NAME>System</SHORT-NAME>
  <DESC>
    <L-2 L="EN">The model shall contain exactly one instance.</L-2>
  </DESC>
  <IN-SCOPE>true</IN-SCOPE>
  <MULTIPLICITY-RESTRICTION>
    <SEVERITY>ERROR</SEVERITY>
    <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
    <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
  </MULTIPLICITY-RESTRICTION>
  <CLASS-CONTENTS>
    <CLASS-CONTENT-CONDITIONAL>
      <SHORT-NAME>Invariant</SHORT-NAME>
      <DESC>
        <L-2 L="EN">The category shall be set to ECU_SYSTEM_DESCRIPTION</L
           -2>
      </DESC>
      <ATTRIBUTE-TAILORINGS>
        <PRIMITIVE-ATTRIBUTE-TAILORING>
          <SHORT-NAME>category</SHORT-NAME>
          <IN-SCOPE>true</IN-SCOPE>
          <MULTIPLICITY-RESTRICTION>
            <SEVERITY>ERROR</SEVERITY>
            <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          </MULTIPLICITY-RESTRICTION>
          <VALUE-RESTRICTION>
            <SEVERITY>ERROR</SEVERITY>
            <PATTERN>ECU_SYSTEM_DESCRIPTION</PATTERN>
          </VALUE-RESTRICTION>
        </PRIMITIVE-ATTRIBUTE-TAILORING>
```

```
          <REFERENCE-TAILORING>
            <SHORT-NAME>fibexElement</SHORT-NAME>
            <DESC>
              <L-2 L="EN">
              This reference is a collection of all elements that
              belong to the System. We expect at least one element
              and at most 100 elements.
              </L-2>
            </DESC>
            <IN-SCOPE>true</IN-SCOPE>
            <MULTIPLICITY-RESTRICTION>
              <SEVERITY>ERROR</SEVERITY>
              <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
              <UPPER-MULTIPLICITY>100</UPPER-MULTIPLICITY>
            </MULTIPLICITY-RESTRICTION>
            <TYPE-TAILORINGS>
              <CONCRETE-CLASS-TAILORING>
                <SHORT-NAME>ECUInstance</SHORT-NAME>
                <DESC>
                  <L-2 L="EN">exactly one ECU instance is required</L-2>
                </DESC>
                <IN-SCOPE>true</IN-SCOPE>
                <MULTIPLICITY-RESTRICTION>
                  <SEVERITY>ERROR</SEVERITY>
                  <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
                  <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
                </MULTIPLICITY-RESTRICTION>
              </CONCRETE-CLASS-TAILORING>
              <CONCRETE-CLASS-TAILORING>
                <SHORT-NAME>CanCluster</SHORT-NAME>
                <DESC>
                  <L-2 L="EN">exactly one CanCluster is required</L-2>
                </DESC>
                <IN-SCOPE>true</IN-SCOPE>
                <MULTIPLICITY-RESTRICTION>
                  <SEVERITY>ERROR</SEVERITY>
                  <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
                  <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
                </MULTIPLICITY-RESTRICTION>
              </CONCRETE-CLASS-TAILORING>
            </TYPE-TAILORINGS>
            <UNRESOLVED-REFERENCE-RESTRICTION>
              <SEVERITY>ERROR</SEVERITY>
            </UNRESOLVED-REFERENCE-RESTRICTION>
          </REFERENCE-TAILORING>
        </ATTRIBUTE-TAILORINGS>
      </CLASS-CONTENT-CONDITIONAL>
    </CLASS-CONTENTS>
    <VALIDATION-ROOT>true</VALIDATION-ROOT>
  </CONCRETE-CLASS-TAILORING>
```

A.3 Class Tailoring With Global and Local MultiplicityRestrictions

Example A.3 specifies ClassTailorings that express the following semantics:

- PPortPrototypes and RPortPrototypes are in scope an may be used without restrictions with respect to the multiplicity. Any exception from this rule has to be defined explicitly by adding further multiplicity restrictions.
- PRPortPrototypes are not allowed.
- In the context of an ParameterSwComponentType an additional restriction applies which disallows the usage of RPortPrototypes in the role port.

Listing A.3: Example of Class Tailoring With Global and Local MultiplicityRestrictions

```
<DATA-EXCHANGE-POINT>
  <SHORT-NAME>MyExchangePointPorts</SHORT-NAME>
  <!-- -->
  <DATA-FORMAT-TAILORING>
    <CLASS-TAILORINGS>
      <!-- -->
      <CONCRETE-CLASS-TAILORING>
        <SHORT-NAME>PPortPrototype</SHORT-NAME>
        <DESC>
          <L-2 L="EN">No restriction with respect to multiplicity of
              PPortPrototypes</L-2>
        </DESC>
        <IN-SCOPE>true</IN-SCOPE>
        <MULTIPLICITY-RESTRICTION>
          <SEVERITY>INFO</SEVERITY>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY-INFINITE>true</UPPER-MULTIPLICITY-INFINITE>
        </MULTIPLICITY-RESTRICTION>
        <VALIDATION-ROOT>false</VALIDATION-ROOT>
      </CONCRETE-CLASS-TAILORING>
      <CONCRETE-CLASS-TAILORING>
        <SHORT-NAME>PRPortPrototype</SHORT-NAME>
        <DESC>
          <L-2 L="EN">No PRPortPrototypes are allowed in the set of
              reachable elements</L-2>
        </DESC>
        <IN-SCOPE>true</IN-SCOPE>
        <MULTIPLICITY-RESTRICTION>
          <SEVERITY>ERROR</SEVERITY>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>0</UPPER-MULTIPLICITY>
        </MULTIPLICITY-RESTRICTION>
        <VALIDATION-ROOT>false</VALIDATION-ROOT>
      </CONCRETE-CLASS-TAILORING>
      <CONCRETE-CLASS-TAILORING>
        <SHORT-NAME>RPortPrototype</SHORT-NAME>
        <DESC>
          <L-2 L="EN">No restriction with respect to multiplicity of
              RPortPrototypes</L-2>
        </DESC>
        <IN-SCOPE>true</IN-SCOPE>
        <MULTIPLICITY-RESTRICTION>
          <SEVERITY>INFO</SEVERITY>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY-INFINITE>true</UPPER-MULTIPLICITY-INFINITE>
        </MULTIPLICITY-RESTRICTION>
        <VALIDATION-ROOT>false</VALIDATION-ROOT>
      </CONCRETE-CLASS-TAILORING>
      <CONCRETE-CLASS-TAILORING>
        <SHORT-NAME>ParameterSwComponentType</SHORT-NAME>
        <DESC>
          <L-2 L="EN">
      No restriction with respect to the number of used
      ParameterSwComponentTypes.
      In the context of the ParameterSwComponentType
      only PPortPrototypes are allowed. (constr_1092)</L-2>
        </DESC>
        <IN-SCOPE>true</IN-SCOPE>
        <MULTIPLICITY-RESTRICTION>
          <SEVERITY>INFO</SEVERITY>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY-INFINITE>true</UPPER-MULTIPLICITY-INFINITE>
        </MULTIPLICITY-RESTRICTION>
        <CLASS-CONTENTS>
          <CLASS-CONTENT-CONDITIONAL>
            <SHORT-NAME>invariant</SHORT-NAME>
            <ATTRIBUTE-TAILORINGS>
              <AGGREGATION-TAILORING>
                <SHORT-NAME>port</SHORT-NAME>
                <DESC>
                  <L-2 L="EN">No restriction with respect to the number of
                     PortPrototypes</L-2>
                </DESC>
                <MULTIPLICITY-RESTRICTION>
                  <SEVERITY>INFO</SEVERITY>
                  <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
                  <UPPER-MULTIPLICITY-INFINITE>true</UPPER-MULTIPLICITY-
                     INFINITE>
                </MULTIPLICITY-RESTRICTION>
                <TYPE-TAILORINGS>
                  <!--
                    No additional restriction for PRPortPrototypes:
                    Globally defined ClassTailoring applies which
                    does not allow the use of  PRPortPrototypes
                    in the context of all references and aggregations
                  -->
                  <!--
                    No additional restriction for PPortPrototypes:
                    Globally defined ClassTailoring applies which
                    allows for unrestricted number of PPortPrototypes
                    in the context of all references and aggregations
                  -->
                  <CONCRETE-CLASS-TAILORING>
                    <SHORT-NAME>RPortPrototype</SHORT-NAME>
                    <DESC>
                      <L-2 L="EN">No RPortPrototypes are allowed at
                         ParameterSwComponentTypes</L-2>
```

```
                    </DESC>
                    <MULTIPLICITY-RESTRICTION>
                      <SEVERITY>ERROR</SEVERITY>
                      <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
                      <UPPER-MULTIPLICITY>0</UPPER-MULTIPLICITY>
                    </MULTIPLICITY-RESTRICTION>
                  </CONCRETE-CLASS-TAILORING>
                </TYPE-TAILORINGS>
              </AGGREGATION-TAILORING>
            </ATTRIBUTE-TAILORINGS>
          </CLASS-CONTENT-CONDITIONAL>
        </CLASS-CONTENTS>
        <VALIDATION-ROOT>true</VALIDATION-ROOT>
      </CONCRETE-CLASS-TAILORING>
    </CLASS-TAILORINGS>
  </DATA-FORMAT-TAILORING>
</DATA-EXCHANGE-POINT>
```

A.4 Class Tailoring That Depends On the Using Role

Example A.4 specifies:

- The initValue of a VariableDataPrototype is optional if the VariableDataPrototype is used in the roles implicitInterRunnableVariable or implicitInterRunnableVariable of a SwcInternalBehavior.
- The initValue of a VariableDataPrototype shall not exist if the VariableDataPrototype is used in the role dataElement of a SenderReceiverInterface.

Listing A.4: Example of Class Tailoring That Depends On the Using Role

```
<CONCRETE-CLASS-TAILORING>
  <SHORT-NAME>SwcInternalBehavior</SHORT-NAME>
  <IN-SCOPE>true</IN-SCOPE>
  <CLASS-CONTENTS>
    <CLASS-CONTENT-CONDITIONAL>
      <SHORT-NAME>Invariant</SHORT-NAME>
      <ATTRIBUTE-TAILORINGS>
        <AGGREGATION-TAILORING>
          <SHORT-NAME>explicitInterRunnableVariable</SHORT-NAME>
          <IN-SCOPE>true</IN-SCOPE>
          <TYPE-TAILORINGS>
            <CONCRETE-CLASS-TAILORING>
              <SHORT-NAME>VariableDataPrototype</SHORT-NAME>
              <IN-SCOPE>true</IN-SCOPE>
              <CLASS-CONTENTS>
                <CLASS-CONTENT-CONDITIONAL>
                  <SHORT-NAME>Invariant</SHORT-NAME>
                  <ATTRIBUTE-TAILORINGS>
                    <AGGREGATION-TAILORING>
                      <SHORT-NAME>initValue</SHORT-NAME>
                      <DESC>
```

```
                        <L-2 L="EN">[TPS_SWCT_01268] Definition of
                           initValue for a VariableDataPrototype or a
                           ParameterDataPrototype</L-2>
                      </DESC>
                      <IN-SCOPE>true</IN-SCOPE>
                      <MULTIPLICITY-RESTRICTION>
                        <SEVERITY>ERROR</SEVERITY>
                        <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
                        <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
                      </MULTIPLICITY-RESTRICTION>
                    </AGGREGATION-TAILORING>
                  </ATTRIBUTE-TAILORINGS>
                </CLASS-CONTENT-CONDITIONAL>
              </CLASS-CONTENTS>
              <VALIDATION-ROOT>false</VALIDATION-ROOT>
            </CONCRETE-CLASS-TAILORING>
          </TYPE-TAILORINGS>
        </AGGREGATION-TAILORING>
        <AGGREGATION-TAILORING>
          <SHORT-NAME>implicitInterRunnableVariable</SHORT-NAME>
          <IN-SCOPE>true</IN-SCOPE>
          <TYPE-TAILORINGS>
            <CONCRETE-CLASS-TAILORING>
              <SHORT-NAME>VariableDataPrototype</SHORT-NAME>
              <IN-SCOPE>true</IN-SCOPE>
              <CLASS-CONTENTS>
                <CLASS-CONTENT-CONDITIONAL>
                  <SHORT-NAME>Invariant</SHORT-NAME>
                  <ATTRIBUTE-TAILORINGS>
                    <AGGREGATION-TAILORING>
                      <SHORT-NAME>initValue</SHORT-NAME>
                      <DESC>
                        <L-2 L="EN">[TPS_SWCT_01268] Definition of
                           initValue for a VariableDataPrototype or a
                           ParameterDataPrototype</L-2>
                      </DESC>
                      <IN-SCOPE>true</IN-SCOPE>
                      <MULTIPLICITY-RESTRICTION>
                        <SEVERITY>ERROR</SEVERITY>
                        <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
                        <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
                      </MULTIPLICITY-RESTRICTION>
                    </AGGREGATION-TAILORING>
                  </ATTRIBUTE-TAILORINGS>
                </CLASS-CONTENT-CONDITIONAL>
              </CLASS-CONTENTS>
              <VALIDATION-ROOT>false</VALIDATION-ROOT>
            </CONCRETE-CLASS-TAILORING>
          </TYPE-TAILORINGS>
        </AGGREGATION-TAILORING>
      </ATTRIBUTE-TAILORINGS>
    </CLASS-CONTENT-CONDITIONAL>
  </CLASS-CONTENTS>
</CONCRETE-CLASS-TAILORING>
<CONCRETE-CLASS-TAILORING>
  <SHORT-NAME>SenderReceiverInterface</SHORT-NAME>
  <IN-SCOPE>true</IN-SCOPE>
  <CLASS-CONTENTS>
    <CLASS-CONTENT-CONDITIONAL>
      <SHORT-NAME>Invariant</SHORT-NAME>
      <ATTRIBUTE-TAILORINGS>
        <AGGREGATION-TAILORING>
          <SHORT-NAME>dataElement</SHORT-NAME>
          <IN-SCOPE>true</IN-SCOPE>
          <TYPE-TAILORINGS>
            <CONCRETE-CLASS-TAILORING>
              <SHORT-NAME>VariableDataPrototype</SHORT-NAME>
              <IN-SCOPE>true</IN-SCOPE>
              <CLASS-CONTENTS>
                <CLASS-CONTENT-CONDITIONAL>
                  <SHORT-NAME>Invariant</SHORT-NAME>
                  <ATTRIBUTE-TAILORINGS>
                    <AGGREGATION-TAILORING>
                      <SHORT-NAME>initValue</SHORT-NAME>
                      <DESC>
                        <L-2 L="EN">[TPS_SWCT_01269] In PortInterfaces,
                            initial values defined for DataPrototypes are
                            ignored</L-2>
                      </DESC>
                      <IN-SCOPE>true</IN-SCOPE>
                      <MULTIPLICITY-RESTRICTION>
                        <UPPER-MULTIPLICITY>0</UPPER-MULTIPLICITY>
                      </MULTIPLICITY-RESTRICTION>
                    </AGGREGATION-TAILORING>
                  </ATTRIBUTE-TAILORINGS>
                </CLASS-CONTENT-CONDITIONAL>
              </CLASS-CONTENTS>
            </CONCRETE-CLASS-TAILORING>
          </TYPE-TAILORINGS>
        </AGGREGATION-TAILORING>
      </ATTRIBUTE-TAILORINGS>
    </CLASS-CONTENT-CONDITIONAL>
  </CLASS-CONTENTS>
</CONCRETE-CLASS-TAILORING>
```

A.5 Class Tailoring That Depends On the Value of an Attribute

Example A.5 specifies a ClassTailoring that specifies the content model of an instance of SwDataDefProps if it attached to an ImplementationDataType with category VALUE or DATA_REFERENCE as described in table "Allowed Attributes vs. category for ImplementationDataType" in the [1].

Listing A.5: Example of Class Tailoring That Depends On the Value of an Attribute

```
<CONCRETE-CLASS-TAILORING>
  <SHORT-NAME>ImplementationDataType</SHORT-NAME>
  <DESC>
```

```
    <L-2 L="EN">Example that demonstates how to express complex
        constraints as defined in [constr_1009] SwDataDefProps applicable
        to ImplementationDataTypes.</L-2>
  </DESC>
  <IN-SCOPE>true</IN-SCOPE>
  <CLASS-CONTENTS>
    <CLASS-CONTENT-CONDITIONAL>
      <SHORT-NAME>Invariant</SHORT-NAME>
      <ATTRIBUTE-TAILORINGS>
        <PRIMITIVE-ATTRIBUTE-TAILORING>
          <SHORT-NAME>category</SHORT-NAME>
          <DESC>
            <L-2 L="EN">[TPS_SWCT_01251] Limited set of values for
                category are applicable for ImplementationDataType.</L-2>
          </DESC>
          <IN-SCOPE>true</IN-SCOPE>
          <MULTIPLICITY-RESTRICTION>
            <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          </MULTIPLICITY-RESTRICTION>
          <DEFAULT-VALUE-HANDLING>NO-DEFAULT</DEFAULT-VALUE-HANDLING>
          <VALUE-RESTRICTION>
            <PATTERN>VALUE|DATA_REFERENCE|FUNCTION_REFERENCE|
                TYPE_REFERENCE|STRUCTURE|UNION|ARRAY</PATTERN>
          </VALUE-RESTRICTION>
        </PRIMITIVE-ATTRIBUTE-TAILORING>
      </ATTRIBUTE-TAILORINGS>
    </CLASS-CONTENT-CONDITIONAL>
    <CLASS-CONTENT-CONDITIONAL>
      <SHORT-NAME>VALUE</SHORT-NAME>
      <CONDITION>
        <PRIMITIVE-ATTRIBUTE-CONDITION>
          <PATTERN>VALUE</PATTERN>
          <ATTRIBUTE-REF DEST="PRIMITIVE-ATTRIBUTE-TAILORING">
              ExampleClassTailoring/ImplementationDataType/Invariant/
              category</ATTRIBUTE-REF>
        </PRIMITIVE-ATTRIBUTE-CONDITION>
      </CONDITION>
      <ATTRIBUTE-TAILORINGS>
        <AGGREGATION-TAILORING>
          <SHORT-NAME>swDataDefProps</SHORT-NAME>
          <IN-SCOPE>true</IN-SCOPE>
          <MULTIPLICITY-RESTRICTION>
            <SEVERITY>ERROR</SEVERITY>
            <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          </MULTIPLICITY-RESTRICTION>
          <TYPE-TAILORINGS>
            <CONCRETE-CLASS-TAILORING>
              <SHORT-NAME>SwDataDefProps</SHORT-NAME>
              <IN-SCOPE>true</IN-SCOPE>
              <CLASS-CONTENTS>
                <CLASS-CONTENT-CONDITIONAL>
                  <SHORT-NAME>Invariant</SHORT-NAME>
                  <ATTRIBUTE-TAILORINGS>
                    <PRIMITIVE-ATTRIBUTE-TAILORING>
                      <SHORT-NAME>additionalNativeTypeQualifier</SHORT-
                          NAME>
                      <IN-SCOPE>true</IN-SCOPE>
                    </PRIMITIVE-ATTRIBUTE-TAILORING>
                    <AGGREGATION-TAILORING>
                      <SHORT-NAME>annotation</SHORT-NAME>
                      <IN-SCOPE>true</IN-SCOPE>
                    </AGGREGATION-TAILORING>
                    <REFERENCE-TAILORING>
                      <SHORT-NAME>baseType</SHORT-NAME>
                      <IN-SCOPE>true</IN-SCOPE>
                      <MULTIPLICITY-RESTRICTION>
                        <SEVERITY>ERROR</SEVERITY>
                        <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
                      </MULTIPLICITY-RESTRICTION>
                    </REFERENCE-TAILORING>
                    <REFERENCE-TAILORING>
                      <SHORT-NAME>compuMethod</SHORT-NAME>
                      <IN-SCOPE>true</IN-SCOPE>
                    </REFERENCE-TAILORING>
                    <REFERENCE-TAILORING>
                      <SHORT-NAME>dataConstr</SHORT-NAME>
                      <IN-SCOPE>true</IN-SCOPE>
                    </REFERENCE-TAILORING>
                    <PRIMITIVE-ATTRIBUTE-TAILORING>
                      <SHORT-NAME>displayFormat</SHORT-NAME>
                      <IN-SCOPE>true</IN-SCOPE>
                    </PRIMITIVE-ATTRIBUTE-TAILORING>
                    <REFERENCE-TAILORING>
                      <SHORT-NAME>implementationDataType</SHORT-NAME>
                      <IN-SCOPE>false</IN-SCOPE>
                    </REFERENCE-TAILORING>
                    <AGGREGATION-TAILORING>
                      <SHORT-NAME>invalidValue</SHORT-NAME>
                      <IN-SCOPE>true</IN-SCOPE>
                    </AGGREGATION-TAILORING>
                    <!-- ... -->
                  </ATTRIBUTE-TAILORINGS>
                </CLASS-CONTENT-CONDITIONAL>
              </CLASS-CONTENTS>
            </CONCRETE-CLASS-TAILORING>
          </TYPE-TAILORINGS>
        </AGGREGATION-TAILORING>
      </ATTRIBUTE-TAILORINGS>
    </CLASS-CONTENT-CONDITIONAL>
    <CLASS-CONTENT-CONDITIONAL>
      <SHORT-NAME>DATA_REFERENCE</SHORT-NAME>
      <CONDITION>
        <PRIMITIVE-ATTRIBUTE-CONDITION>
          <PATTERN>DATA_REFERENCE</PATTERN>
          <ATTRIBUTE-REF DEST="PRIMITIVE-ATTRIBUTE-TAILORING">
              ExampleClassTailoring/ImplementationDataType/Invariant/
              category</ATTRIBUTE-REF>
        </PRIMITIVE-ATTRIBUTE-CONDITION>
      </CONDITION>
      <ATTRIBUTE-TAILORINGS>
        <AGGREGATION-TAILORING>
          <SHORT-NAME>swDataDefProps</SHORT-NAME>
```

```
<IN-SCOPE>true</IN-SCOPE>
<MULTIPLICITY-RESTRICTION>
  <SEVERITY>ERROR</SEVERITY>
  <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
</MULTIPLICITY-RESTRICTION>
<TYPE-TAILORINGS>
  <CONCRETE-CLASS-TAILORING>
    <SHORT-NAME>SwDataDefProps</SHORT-NAME>
    <IN-SCOPE>true</IN-SCOPE>
    <CLASS-CONTENTS>
      <CLASS-CONTENT-CONDITIONAL>
        <SHORT-NAME>Invariant</SHORT-NAME>
        <ATTRIBUTE-TAILORINGS>
          <PRIMITIVE-ATTRIBUTE-TAILORING>
            <SHORT-NAME>additionalNativeTypeQualifier</SHORT-
               NAME>
            <IN-SCOPE>true</IN-SCOPE>
          </PRIMITIVE-ATTRIBUTE-TAILORING>
          <AGGREGATION-TAILORING>
            <SHORT-NAME>annotation</SHORT-NAME>
            <IN-SCOPE>true</IN-SCOPE>
          </AGGREGATION-TAILORING>
          <REFERENCE-TAILORING>
            <SHORT-NAME>baseType</SHORT-NAME>
            <IN-SCOPE>false</IN-SCOPE>
            <MULTIPLICITY-RESTRICTION>
              <SEVERITY>ERROR</SEVERITY>
              <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
            </MULTIPLICITY-RESTRICTION>
          </REFERENCE-TAILORING>
          <REFERENCE-TAILORING>
            <SHORT-NAME>compuMethod</SHORT-NAME>
            <IN-SCOPE>false</IN-SCOPE>
          </REFERENCE-TAILORING>
          <REFERENCE-TAILORING>
            <SHORT-NAME>dataConstr</SHORT-NAME>
            <IN-SCOPE>false</IN-SCOPE>
          </REFERENCE-TAILORING>
          <PRIMITIVE-ATTRIBUTE-TAILORING>
            <SHORT-NAME>displayFormat</SHORT-NAME>
            <IN-SCOPE>false</IN-SCOPE>
          </PRIMITIVE-ATTRIBUTE-TAILORING>
          <REFERENCE-TAILORING>
            <SHORT-NAME>implementationDataType</SHORT-NAME>
            <IN-SCOPE>false</IN-SCOPE>
          </REFERENCE-TAILORING>
          <AGGREGATION-TAILORING>
            <SHORT-NAME>invalidValue</SHORT-NAME>
            <IN-SCOPE>false</IN-SCOPE>
          </AGGREGATION-TAILORING>
          <!-- ... -->
        </ATTRIBUTE-TAILORINGS>
      </CLASS-CONTENT-CONDITIONAL>
    </CLASS-CONTENTS>
  </CONCRETE-CLASS-TAILORING>
</TYPE-TAILORINGS>
```

```
          </AGGREGATION-TAILORING>
        </ATTRIBUTE-TAILORINGS>
      </CLASS-CONTENT-CONDITIONAL>
    </CLASS-CONTENTS>
  </CONCRETE-CLASS-TAILORING>
```

A.6 Class Tailoring That Depends on Existence of Attribute

Example A.6 specifies a `ClassTailoring` that specifies the content model of a class that depends on the existence of an attribute: If there exists an `NvBlockDescriptor.romBlock` then `NvBlockDescriptor.nvBlockNeeds.nRomBlocks` is mandatory and its value shall be bigger than 1.

Listing A.6: Example of Class Tailoring That Depends on Existence of Attribute

```
<CONCRETE-CLASS-TAILORING>
  <SHORT-NAME>NvBlockDescriptor</SHORT-NAME>
  <DESC>
    <L-2 L="EN">The input may contain a 'NvBlockDescriptor'.</L-2>
  </DESC>
  <IN-SCOPE>true</IN-SCOPE>
  <MULTIPLICITY-RESTRICTION>
    <SEVERITY>INFO</SEVERITY>
    <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
    <UPPER-MULTIPLICITY-INFINITE>true</UPPER-MULTIPLICITY-INFINITE>
  </MULTIPLICITY-RESTRICTION>
  <CLASS-CONTENTS>
    <CLASS-CONTENT-CONDITIONAL>
      <SHORT-NAME>Invariant</SHORT-NAME>
      <ATTRIBUTE-TAILORINGS>
        <AGGREGATION-TAILORING>
          <SHORT-NAME>romBlock</SHORT-NAME>
          <DESC>
            <L-2 L="EN">The input may contain a 'romBlock'.</L-2>
          </DESC>
          <IN-SCOPE>true</IN-SCOPE>
          <MULTIPLICITY-RESTRICTION>
            <SEVERITY>ERROR</SEVERITY>
            <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
            <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          </MULTIPLICITY-RESTRICTION>
        </AGGREGATION-TAILORING>
      </ATTRIBUTE-TAILORINGS>
    </CLASS-CONTENT-CONDITIONAL>
    <CLASS-CONTENT-CONDITIONAL>
      <SHORT-NAME>UsingRomBlock</SHORT-NAME>
      <DESC>
        <L-2 L="EN">Content that is required if romBlock is defined</L-2>
      </DESC>
      <CONDITION>
        <AGGREGATION-CONDITION>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <AGGREGATION-REF DEST="AGGREGATION-TAILORING">
             ExampleClassTailoring/NvBlockDescriptor/Invariant/romBlock</
             AGGREGATION-REF>
        </AGGREGATION-CONDITION>
      </CONDITION>
      <ATTRIBUTE-TAILORINGS>
        <AGGREGATION-TAILORING>
          <SHORT-NAME>nvBlockNeeds</SHORT-NAME>
          <DESC>
            <L-2 L="EN">The input may contain 'nvBlockNeeds'.</L-2>
          </DESC>
          <IN-SCOPE>true</IN-SCOPE>
          <MULTIPLICITY-RESTRICTION>
            <SEVERITY>ERROR</SEVERITY>
            <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
            <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          </MULTIPLICITY-RESTRICTION>
          <TYPE-TAILORINGS>
            <CONCRETE-CLASS-TAILORING>
              <SHORT-NAME>NvBlockNeeds</SHORT-NAME>
              <IN-SCOPE>true</IN-SCOPE>
              <MULTIPLICITY-RESTRICTION>
                <SEVERITY>ERROR</SEVERITY>
                <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
                <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
              </MULTIPLICITY-RESTRICTION>
              <CLASS-CONTENTS>
                <CLASS-CONTENT-CONDITIONAL>
                  <SHORT-NAME>Invariant</SHORT-NAME>
                  <ATTRIBUTE-TAILORINGS>
                    <PRIMITIVE-ATTRIBUTE-TAILORING>
                      <SHORT-NAME>nRomBlocks</SHORT-NAME>
                      <DESC>
                        <L-2 L="EN">'nRomBlocks' shall be present and
                            greater than 0 in case the enclosing
                            NvBlockDescriptor has a romBlock.</L-2>
                      </DESC>
                      <IN-SCOPE>true</IN-SCOPE>
                      <MULTIPLICITY-RESTRICTION>
                        <SEVERITY>ERROR</SEVERITY>
                        <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
                        <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
                      </MULTIPLICITY-RESTRICTION>
                      <VALUE-RESTRICTION>
                        <MIN INTERVAL-TYPE="CLOSED">1</MIN>
                      </VALUE-RESTRICTION>
                    </PRIMITIVE-ATTRIBUTE-TAILORING>
                  </ATTRIBUTE-TAILORINGS>
                </CLASS-CONTENT-CONDITIONAL>
              </CLASS-CONTENTS>
              <VALIDATION-ROOT>false</VALIDATION-ROOT>
            </CONCRETE-CLASS-TAILORING>
          </TYPE-TAILORINGS>
        </AGGREGATION-TAILORING>
      </ATTRIBUTE-TAILORINGS>
    </CLASS-CONTENT-CONDITIONAL>
```

```
    </CLASS-CONTENTS>
  </CONCRETE-CLASS-TAILORING>
```

B Glossary

Artifact This is a Work Product Definition that provides a description and definition for tangible work product types. Artifacts may be composed of other artifacts ([23]).

At a high level, an artifact is represented as a single conceptual file.

AUTOSAR Tool This is a software tool which supports one or more tasks defined as AUTOSAR tasks in the methodology. Depending on the supported tasks, an AUTOSAR tool can act as an authoring tool, a converter tool, a processor tool or as a combination of those (see separate definitions).

AUTOSAR Authoring Tool An AUTOSAR Tool used to create and modify AUTOSAR XML Descriptions. Example: System Description Editor.

AUTOSAR Converter Tool An AUTOSAR Tool used to create AUTOSAR XML files by converting information from other AUTOSAR XML files. Example: ECU Flattener

AUTOSAR Definition This is the definition of parameters which can have values. One could say that the parameter values are Instances of the definitions. But in the meta model hierarchy of AUTOSAR, definitions are also instances of the meta model and therefore considered as a description. Examples for AUTOSAR definitions are: `EcucParameterDef`, `PostBuildVariantCriterion`, `SwSystemconst`.

AUTOSAR XML Description In AUTOSAR this means "filled Template". In fact an AUTOSAR XML description is the XML representation of an AUTOSAR model.

The AUTOSAR XML description can consist of several files. Each individual file represents an AUTOSAR partial model and shall validate successfully against the AUTOSAR XML schema.

AUTOSAR Meta-Model This is an UML2.0 model that defines the language for describing AUTOSAR systems. The AUTOSAR meta-model is an UML representation of the AUTOSAR templates. UML2.0 class diagrams are used to describe the attributes and their interrelationships. Stereotypes, UML tags and OCL expressions (object constraint language) are used for defining specific semantics and constraints.

AUTOSAR Meta-Model Tool The AUTOSAR Meta-Model Tool is the tool that generates different views (class tables, list of constraints, diagrams, XML Schema etc.) on the AUTOSAR meta-model.

AUTOSAR Model This is a representation of an AUTOSAR product. The AUTOSAR model represents aspects suitable to the intended use according to the AUTOSAR methodology.

Strictly speaking, this is an instance of the AUTOSAR meta-model. The information contained in the AUTOSAR model can be anything that is representable according to the AUTOSAR meta-model.

AUTOSAR Partial Model In AUTOSAR, the possible partitioning of models is marked in the meta-model by ≪atpSplitable≫. One partial model is represented in an AUTOSAR XML description by one file. The partial model does not need to fulfill all semantic constraints applicable to an AUTOSAR model.

AUTOSAR Processor Tool An AUTOSAR Tool used to create non-AUTOSAR files by processing information from AUTOSAR XML files. Example: RTE Generator

AUTOSAR Specification Element An AUTOSAR Specification Element is a named element that is part of an AUTOSAR specification. Examples: requirement, constraint, specification item, class or attribute in the meta model, methodology, deliverable, methodology activity, model element, bsw module etc.

AUTOSAR Template The term "Template" is used in AUTOSAR to describe the format different kinds of descriptions. The term template comes from the idea, that AUTOSAR defines a kind of form which shall be filled out in order to describe a model. The filled form is then called the description.

In fact the AUTOSAR templates are now defined as a meta-model.

AUTOSAR Validation Tool A specialized `AUTOSAR Tool` which is able to check an AUTOSAR model against the rules defined by a profile.

AUTOSAR XML Schema This is a W3C XML schema that defines the language for exchanging AUTOSAR models. This Schema is derived from the AUTOSAR meta-model. The AUTOSAR XML Schema defines the AUTOSAR data exchange format.

Blueprint This is a model from which other models can be derived by copy and refinement. Note that in contrast to meta model resp. types, this process is *not* an instantiation.

Instance Generally this is a particular exemplar of a model or of a type.

Life Cycle Life Cycle is the course of development/evolutionary stages of a model element during its life time.

Meta-Model This defines the building blocks of a model. In that sense, a Meta-Model represents the language for building models.

Meta-Data This includes pertinent information about data, including information about the authorship, versioning, access-rights, timestamps etc.

Model A Model is an simplified representation of reality. The model represents the aspects suitable for an intended purpose.

Partial Model This is a part of a model which is intended to be persisted in one particular artifact.

Pattern in GST This is an approach to simplify the definition of the meta model by applying a model transformation. This transformation creates an enhanced model out of an annotated model.

Profile Authoring Support Data Data that is used for efficient authoring of a profile. E.g. list of referable constraints, meta-classes, meta-attributes or other reusable model assets (blueprints)

Profile Authoring Tool A specialized `AUTOSAR Tool` which focuses on the authoring of profiles for data exchange points. It e.g. provides support for the creation of profiles from scratch, modification of existing profiles or composition of existing profiles.

Profile Compatibility Checker Tool A specialized `AUTOSAR Tool` which focuses on checking the compatibility of profiles for data exchange. Note that this compatibility check includes manual compatibility checks by engineers and automated assistance using more formal algorithms.

Profile Consistency Checker Tool A specialized `AUTOSAR Tool` which focuses on checking the consistency of profiles.

Property A property is a structural feature of an object. As an example a “connector” has the properties “receive port” and “send port”

Properties are made variant by the ≪`atpVariation`≫.

Prototype This is the implementation of a role of a type within the definition of another type. In other words a type may contain Prototypes that in turn are typed by "Types". Each one of these prototypes becomes an instance when this type is instantiated.

Type A type provides features that can appear in various roles of this type.

Value This is a particular value assigned to a “Definition”.

Variability Variability of a system is its quality to describe a set of variants. These variants are characterized by variant specific property settings and / or selections. As an example, such a system property selection manifests itself in a particular “receive port” for a connection.

This is implemented using the ≪`atpVariation`≫.

Variant A system variant is a concrete realization of a system, so that all its properties have been set respectively selected. The software system has no variability anymore with respect to the binding time.

This is implemented using `EvaluatedVariantSet`.

Variation Binding A variant is the result of a variation binding process that resolves the variability of the system by assigning particular values/selections to all the system’s properties.

This is implemented by `VariationPoint`.

Variation Binding Time The variation binding time determines the step in the methodology at which the variability given by a set of variable properties is resolved.

This is implemented by `vh.LatestBindingtime` at the related properties.

Variation Definition Time The variation definition time determines the step in the methodology at which the variation points are defined.

Variation Point A variation point indicates that a property is subject to variation. Furthermore, it is associated with a condition and a binding time which define the system context for the selection / setting of a concrete variant.

This is implemented by `VariationPoint`.

C Change History

C.1 Change History R4.0.3

C.1.1 Added Constraints

Number	Heading
[constr_2500]	PortInterfaces shall be of same kind
[constr_2526]	PortInterfaces need to be compatible to the blueprints
[constr_2527]	Blueprints shall live in package of a proper category
[constr_2528]	PortPrototypes shall not refer to blueprints of a PortInterface
[constr_2529]	PortPrototypeBlueprints and derived PortPrototypes shall reference proper PortInterfaces
[constr_2540]	Tagged text category
[constr_2542]	Compatibility of introduction of blueprint and blueprinted element
[constr_2543]	Specify a name pattern in blueprints
[constr_2546]	References from Blueprint to Blueprint need to be replaced in derived objects
[constr_2553]	shortName shall follow the pattern defined in the Blueprint
[constr_2554]	Derived objects shall match the blueprints
[constr_2555]	Derived objects may have more attributes than the blueprints
[constr_2556]	No Blueprint Motivated VariationPoints in AUTOSAR Descriptions
[constr_2563]	BswModuleDescription blueprints should not have a BswModuleBehavior
[constr_2564]	VariationPoint in Blueprints of PackageableElements
[constr_2565]	Trace shall not be nested
[constr_2566]	Blueprintmapping shall map appropriate elements
[constr_2568]	SwComponentTypes shall be of same kind
[constr_2569]	Purely Bluprint Motivated VariationPoints
[constr_2570]	No Blueprints in system descriptions
[constr_2571]	Outgoing references from Blueprints

Table C.1: Added Constraints in 4.0.3

C.1.2 Added Specification Items

Number	Heading
[TPS_STDT_00037]	Port Direction
[TPS_STDT_00038]	Life Cycle Support
[TPS_STDT_00040]	Influence of ECUC
[TPS_STDT_00041]	Constraints may be Violated in Blueprints
[TPS_STDT_00042]	namePattern for short names of TraceableText in Template Documents
[TPS_STDT_00043]	Blueprinting LifeCycleDefinitionGroups
[TPS_STDT_00044]	Transferring VariationPoint
[TPS_STDT_00045]	Transferring Objects in General
[TPS_STDT_00046]	Configuration dependent properties
[TPS_STDT_00047]	Ignore Blueprint Attributes
[TPS_STDT_00048]	Express Decisions when Deriving Objects
[TPS_STDT_00049]	Blueprinting Enumerators
[TPS_STDT_00050]	namePattern for AUTOSAR delivered Files
[TPS_STDT_00051]	Handling references when deriving objects from blueprints

[TPS_STDT_00052]	Characteristics of TraceableText
[TPS_STDT_00053]	Expression of obligation
[TPS_STDT_00054]	Organisation of TraceableText
[TPS_STDT_00055]	General Syntax for Name Patterns

Table C.2: Added Specification Items in 4.0.3

C.2 Change History R4.1.1

C.2.1 Added Constraints

Number	Heading

Table C.3: Added Constraints in 4.1.1

C.2.2 Added Specification Items

Number	Heading
[TPS_STDT_00056]	Identifying not applicable requirements
[TPS_STDT_00057]	Identifying generally fulfilled requirements
[TPS_STDT_00058]	Identifying requirements which need more specialization
[TPS_STDT_00059]	TraceableText
[TPS_STDT_00060]	StructuredReq
[TPS_STDT_00062]	Blueprinting Elements of AccessControl
[TPS_STDT_00063]	Blueprinting BuildActionManifest
[TPS_STDT_00064]	Applied Life Cycle Information Sets on AUTOSAR provided Models (M1)
[TPS_STDT_00065]	Nested Blueprint Can be Used as Blueprint of its own
[TPS_STDT_00066]	Blueprinting PortInterface
[TPS_STDT_00067]	Standardized Path for Standardized Elements
[TPS_STDT_00068]	Expressing "stem"-Relation of Keywords
[TPS_STDT_00069]	Attributes of Keyword
[TPS_STDT_00070]	Classification of Keywords
[TPS_STDT_00071]	Blueprinting ConsistencyNeeds
[TPS_STDT_00072]	Same Meta Class For Blueprints and Derived Objects
[TPS_STDT_00073]	Early definition of ConsistencyNeeds
[TPS_STDT_00074]	Categorization of Blueprints of ConsistencyNeeds
[TPS_STDT_00075]	Categories for DataPrototypeGroup in a Blueprint of ConsistencyNeeds
[TPS_STDT_00076]	Categories for RunnableEntityGroup in a Blueprint of ConsistencyNeeds
[TPS_STDT_00077]	Blueprinting KeywordSet
[TPS_STDT_00078]	Representation of requirements in AUTOSAR documents

Table C.4: Added Specification Items in 4.1.1

C.3 Change History R4.1.2

C.3.1 Added Constraints

Number	Heading

Table C.5: Added Constraints in 4.1.2

C.3.2 Added Specification Items

Number	Heading
[TPS_STDT_00006]	Applying expressionPattern
[TPS_STDT_00010]	General Syntax for Expression Patterns
[TPS_STDT_00021]	Specialization of BlueprintFormula
[TPS_STDT_00079]	Blueprinting VfbTiming
[TPS_STDT_00080]	Representation of specification items in AUTOSAR documents
[TPS_STDT_00081]	Representation of constraint items in AUTOSAR documents

Table C.6: Added Specification Items in 4.1.2

C.4 Change History R4.1.3

C.4.1 Added Constraints in 4.1.3

Number	Heading
[constr_2589]	In VFB Timing Blueprint TDEventVfbPort shall reference PortPrototypeBlueprint

Table C.7: Added Constraints in 4.1.3

C.4.2 Changed Constraints in 4.1.3

none

C.4.3 Deleted Constraints in 4.1.3

none

C.4.4 Added Traceables in 4.1.3

Id	Heading
[TPS_STDT_00026]	Blueprinting SwAddrMethod

Table C.8: Added Traceables in 4.1.3

C.4.5 Changed Traceables in 4.1.3

Id	Heading
[TPS_STDT_00055]	General Syntax for Name Patterns
[TPS_STDT_00057]	Identifying generally fulfilled requirements

Table C.9: Changed Traceables in 4.1.3

C.4.6 Deleted Traceables in 4.1.3

none

C.5 Change History R4.2.1

C.5.1 Added Constraints in 4.2.1

Id	Heading
[constr_2590]	One BlueprintPolicy is allowed
[constr_2591]	BlueprintPolicyNotModifiable
[constr_2592]	No BlueprintPolicy
[constr_2593]	Expression for identifying the attribute a BlueprintPolicy relates to

Table C.10: Added Constraints in 4.2.1

C.5.2 Changed Constraints in 4.2.1

Id	Heading
[constr_2540]	Tagged text category

Table C.11: Changed Constraints in 4.2.1

C.5.3 Deleted Constraints in 4.2.1

none

C.5.4 Added Traceables in 4.2.1

Id	Heading
[TPS_STDT_00029]	Representation of test items in AUTOSAR documents
[TPS_STDT_00032]	BlueprintPolicy
[TPS_STDT_00039]	Xpath Expressions for BlueprintPolicy
[TPS_STDT_00061]	PortPrototypeBlueprint can own both RPortComSpecs and PPortComSpecs
[TPS_STDT_00082]	Multiple existence of initValue in the context of a PortPrototypeBlueprint

Table C.12: Added Traceables in 4.2.1

C.5.5 Changed Traceables in 4.2.1

Id	Heading
[TPS_STDT_00004]	Abbreviated Name
[TPS_STDT_00012]	Defining Keywords
[TPS_STDT_00021]	Specialization of BlueprintFormula
[TPS_STDT_00041]	Constraints may be violated in Blueprints
[TPS_STDT_00067]	Standardized Path for Standardized Elements
[TPS_STDT_00068]	Expressing "stem"-Relation of Keywords
[TPS_STDT_00069]	Attributes of Keyword
[TPS_STDT_00070]	Classification of Keywords

Table C.13: Changed Traceables in 4.2.1

C.5.6 Deleted Traceables in 4.2.1

none

C.6 Change History R4.2.2

C.6.1 Added Constraints in 4.2.2

none

C.6.2 Changed Constraints in 4.2.2

Id	Heading
[constr_2592]	No BlueprintPolicy

Table C.14: Changed Constraints in 4.2.2

C.6.3 Deleted Constraints in 4.2.2

none

C.6.4 Added Traceables in 4.2.2

none

C.6.5 Changed Traceables in 4.2.2

Id	Heading

[TPS_STDT_00039]	Xpath Expressions for BlueprintPolicy
[TPS_STDT_00077]	Blueprinting KeywordSet
[TPS_STDT_00080]	Representation of specification items in AUTOSAR documents
[TPS_STDT_00081]	Representation of constraint items in AUTOSAR documents

Table C.15: Changed Traceables in 4.2.2

C.6.6 Deleted Traceables in 4.2.2

none

C.7 Change History R4.3.0

C.7.1 Added Constraints in 4.3.0

Id	Heading
[constr_2597]	ClientServerOperationBlueprintMapping constraints number of arguments
[constr_2598]	ClientServerOperationBlueprintMapping constraints the types of arguments
[constr_2603]	Use of "applies to" in context of the specification level
[constr_2604]	Allowed uptraces in context of "applies to" values
[constr_2608]	Custom extensions shall be part of the Documentation that is referenced by the Baseline
[constr_2609]	Single revision per AUTOSAR standard
[constr_2610]	No alternativeName if matching via shortName
[constr_2611]	Referenced AUTOSAR Specification Elements shall be part of the AUTOSAR Specification Baseline
[constr_2612]	shortName of ConcreteClassTailoring shall match the name of an AUTOSAR specified concrete meta-class
[constr_2613]	shortName of AbstractClassTailoring shall match the name of an AUTOSAR specified abstract meta-class
[constr_2614]	PrimitiveAttributeCondition.attribute shall reference invariant owned PrimitiveAttributeTailoring, only
[constr_2615]	AggregationCondition.aggregation shall reference invariant owned AggregationTailoring, only
[constr_2616]	ReferenceCondition.reference shall reference invariant owned ReferenceTailoring, only
[constr_2617]	ClassTailoring.variationRestriction only applicable for «atpVariation» classes
[constr_2618]	ShortName of AttributeTailoring shall match owned or inherited attributes
[constr_2619]	No AttributeTailoring for Derived or Abstract Attributes
[constr_2620]	shortName of PrimitiveAttributeTailoring shall be a primitive attribute in the referenced Baseline
[constr_2621]	The shortName of AggregationTailoring shall match the name of an AUTOSAR specified aggregation of the meta-class
[constr_2622]	The shortName of ReferenceTailoring shall match the name of an AUTOSAR specified reference of the meta-class
[constr_2623]	Referenced SdgClass shall be part of a SdgDef that is referenced by the Baseline

[constr_2624]	AttributeTailoring.variationRestriction only applicable for «atpVariation» attributes

Table C.16: Added Constraints in 4.3.0

C.7.2 Changed Constraints in 4.3.0

Id	Heading
[constr_2546]	References in derived model elements
[constr_2553]	shortName shall follow the pattern defined in the Blueprint

Table C.17: Changed Constraints in 4.3.0

C.7.3 Deleted Constraints in 4.3.0

Id	Heading
[constr_2542]	Compatibility of longName, desc and introduction of blueprint and blueprinted element
[constr_2543]	Specify a name pattern in blueprints
[constr_2555]	Derived objects may have more attributes than the blueprints

Table C.18: Deleted Constraints in 4.3.0

C.7.4 Added Traceables in 4.3.0

Id	Heading
[TPS_STDT_00083]	Blueprinting ClientServerInterfaceToBswModuleEntryBlueprintMapping
[TPS_STDT_00084]	ClientServerOperationBlueprintMapping predetermines the implementation of an ClientServerOperation
[TPS_STDT_00085]	Compatibility of longName, desc and introduction of blueprint and blueprinted element
[TPS_STDT_00086]	Specify a name pattern in blueprints
[TPS_STDT_00087]	Derived objects may have more attributes than the blueprints
[TPS_STDT_00088]	Representation of constraint items in AUTOSAR non template documents
[TPS_STDT_00089]	Identifying specification items which are constraints in AUTOSAR non template documents
[TPS_STDT_00090]	Blueprinting BswEntryRelationshipSet
[TPS_STDT_00091]	Blueprinting BswEntryRelationshipSet
[TPS_STDT_00100]	Motivation of Description of Data Exchange Points
[TPS_STDT_00101]	Compatibility of ConcreteClassTailorings
[TPS_STDT_00102]	Referencing AUTOSAR Specification Elements via shortName
[TPS_STDT_00103]	Referencing AUTOSAR Specification Elements via alternativeName
[TPS_STDT_00104]	Referencing Custom Specification Elements
[TPS_STDT_00105]	Serialized Profile
[TPS_STDT_00106]	Effective Profile
[TPS_STDT_00107]	Validation Semantics of global ConcreteClassTailoring.multiplicityRestriction with validationRoot==true
[TPS_STDT_00108]	Validation Semantics of global ConcreteClassTailoring.multiplicityRestriction with validationRoot==false

[TPS_STDT_00109]	AUTOSAR Standardized Concrete Meta-Classes
[TPS_STDT_00110]	Identification of Potential Interoperability Issues
[TPS_STDT_00111]	AUTOSAR Standardized Constraints
[TPS_STDT_00112]	Validation Semantics of ClassTailoring.multiplicityRestriction in the context of AggregationTailoring.typeTailoring
[TPS_STDT_00113]	Validation Semantics of AbstractClassTailoring.multiplicityRestriction
[TPS_STDT_00114]	MultiplicityRestrictionWithSeverity in the context of ClassTailoring vs. AggregationTailoring/ReferenceTailoring
[TPS_STDT_00115]	Analysis of Tool Compatibility
[TPS_STDT_00116]	Limitation of Analysis of Profile of Data Exchange Points
[TPS_STDT_00117]	Agreed Profile of Data Exchange Point
[TPS_STDT_00118]	Compliance with Profile of Data Exchange Point
[TPS_STDT_00119]	Validation Semantics of ClassTailoring.multiplicityRestriction in the context of ReferenceTailoring.typeTailoring
[TPS_STDT_00120]	Purpose of DataExchangePoint
[TPS_STDT_00121]	High-level Overview Description of DataExchangePoint
[TPS_STDT_00122]	Purpose of Baseline
[TPS_STDT_00123]	Guidance on how to specify SpecificationDocumentScope and DocumentElementScope
[TPS_STDT_00124]	Purpose of SpecElementScope
[TPS_STDT_00125]	Trigger for Evaluation of Constraints
[TPS_STDT_00126]	Definition: Data Format Elements
[TPS_STDT_00127]	Validation Environment
[TPS_STDT_00128]	Compatibility of SpecificationDocumentScopes
[TPS_STDT_00129]	Semantics of DataFormatElementScope with inScope==true
[TPS_STDT_00130]	Navigation strategy for validation
[TPS_STDT_00131]	Compatibility of AggregationTailoring
[TPS_STDT_00132]	Purpose of SdgTailoring
[TPS_STDT_00133]	Compatibility of ReferenceTailoring
[TPS_STDT_00134]	Compatibility of PrimitiveAttributeTailoring
[TPS_STDT_00135]	Compatibility of ClassContentConditional
[TPS_STDT_00136]	Compatibility of AttributeTailoring
[TPS_STDT_00138]	Purpose of ReferenceTailoring
[TPS_STDT_00139]	AUTOSAR Standardized References of Meta-Class
[TPS_STDT_00140]	Purpose of AggregationTailoring
[TPS_STDT_00141]	AUTOSAR Standardized Aggregations of Meta-Class
[TPS_STDT_00142]	Purpose of PrimitiveAttributeTailoring
[TPS_STDT_00143]	AUTOSAR Standardized Primitive Attributes of Meta-Class
[TPS_STDT_00144]	Purpose of AttributeTailoring
[TPS_STDT_00145]	Purpose of ClassTailoring
[TPS_STDT_00146]	AUTOSAR Standardized Abstract Meta-Classes
[TPS_STDT_00147]	Purpose of ConstraintTailoring
[TPS_STDT_00156]	Purpose of SpecificationScope
[TPS_STDT_00157]	Purpose of DataFormatTailoring
[TPS_STDT_00159]	Semantics of Attribute that is in Scope
[TPS_STDT_00160]	Compatibility of DocumentElementScopes
[TPS_STDT_00163]	Validation Semantics of ConcreteClassTailoring
[TPS_STDT_00164]	Semantics of a Constraint that is out of Scope
[TPS_STDT_00165]	Semantics of Constraint that is in Scope
[TPS_STDT_00167]	Semantics of SdgTailoring that is in scope
[TPS_STDT_00168]	Share documentation of Rationale
[TPS_STDT_00169]	Handling of unresolved references

[TPS_STDT_00170]	Local documentation of Rationale
[TPS_STDT_00172]	Purpose of RestrictionWithSeverity
[TPS_STDT_00173]	Purpose of ValueRestrictionWithSeverity
[TPS_STDT_00174]	Purpose of MultiplicityRestrictionWithSeverity
[TPS_STDT_00175]	Purpose of VariationRestrictionWithSeverity
[TPS_STDT_00176]	Context specific Tailoring
[TPS_STDT_00177]	Global ClassTailoring
[TPS_STDT_00178]	Role Specific ClassTailoring
[TPS_STDT_00179]	Conditional ClassTailoring
[TPS_STDT_00180]	Invariant Content Model
[TPS_STDT_00181]	Conditional Content Model
[TPS_STDT_00182]	Validation Semantics of AbstractClassTailoring
[TPS_STDT_00183]	Compatibility of Baselines
[TPS_STDT_00186]	Scope and Restrictions of Data Format Elements
[TPS_STDT_00187]	Purpose of DocumentElementScope
[TPS_STDT_00188]	Purpose of SpecificationDocumentScope
[TPS_STDT_00190]	Default Scope of concrete Meta Classes
[TPS_STDT_00191]	Purpose of Baseline Profile of Data Exchange Point
[TPS_STDT_00192]	Default Scope of AUTOSAR Specifications
[TPS_STDT_00193]	Default Scope of AUTOSAR Specification Elements
[TPS_STDT_00195]	Default Scope of Meta Attributes
[TPS_STDT_00196]	Default Validation Root of concrete Meta Classes
[TPS_STDT_00197]	Default multiplicityRestriction of Meta Classes
[TPS_STDT_00198]	Default multiplicityRestriction of Meta Attributes
[TPS_STDT_00199]	Default variationRestriction of Meta Attributes
[TPS_STDT_00200]	Default variationRestriction of Meta Classes
[TPS_STDT_00201]	Compatibility of VariationRestrictionWithSeverity.variation
[TPS_STDT_00202]	Compatibility of VariationRestrictionWithSeverity.validBind-ingTime
[TPS_STDT_00203]	Default PrimitiveAttributeTailoring.valueRestriction
[TPS_STDT_00204]	Default PrimitiveAttributeTailoring.defaultValueHandling
[TPS_STDT_00205]	Compatibility of ValueRestrictionWithSeverity
[TPS_STDT_00206]	Compatibility of UnresolvedReferenceRestrictionWithSeverity
[TPS_STDT_00207]	Default ReferenceTailoring.unresolvedReferenceRestriction
[TPS_STDT_00208]	Compatibility of ConstraintTailorings
[TPS_STDT_00209]	Compatibility of SdgTailorings
[TPS_STDT_00210]	Compatibility of MultiplicityRestrictionWithSeverity

Table C.19: Added Traceables in 4.3.0

C.7.5 Changed Traceables in 4.3.0

Id	Heading
[TPS_STDT_00005]	Compliance with Blueprints
[TPS_STDT_00029]	Representation of test items in AUTOSAR documents
[TPS_STDT_00042]	namePattern for shortNames of TraceableText in Standardization Docu-ments
[TPS_STDT_00044]	Transferring VariationPoint
[TPS_STDT_00077]	Blueprinting KeywordSet
[TPS_STDT_00078]	Representation of requirements in AUTOSAR documents
[TPS_STDT_00080]	Representation of specification items in AUTOSAR documents
[TPS_STDT_00081]	Representation of constraint items in AUTOSAR documents

Table C.20: Changed Traceables in 4.3.0

C.7.6 Deleted Traceables in 4.3.0

none

C.8 Change History R4.3.1

C.8.1 Added Constraints in 4.3.1

none

C.8.2 Changed Constraints in 4.3.1

none

C.8.3 Deleted Constraints in 4.3.1

none

C.8.4 Added Traceables in 4.3.1

none

C.8.5 Changed Traceables in 4.3.1

none

C.8.6 Deleted Traceables in 4.3.1

none

C.9 Change History R4.4.0

C.9.1 Added Constraints in 4.4.0

Number	Heading
[constr_2625]	Allowed uptraces wrt. life cycles

Table C.21: Added Constraints in 4.4.0

C.9.2 Changed Constraints in 4.4.0

Number	Heading
[constr_2553]	shortName shall follow the pattern defined in the Blueprint
[constr_2554]	Derived objects shall match the blueprints
[constr_2569]	Purely Blueprint Motivated VariationPoints

Table C.22: Changed Constraints in 4.4.0

C.9.3 Deleted Constraints in 4.4.0

none

C.9.4 Added Traceables in 4.4.0

Number	Heading
[TPS_STDT_00092]	Return values of the BlueprintFormula.ecuc query
[TPS_STDT_00211]	Specification of the AUTOSAR Standards that are part of the Baseline

Table C.23: Added Traceables in 4.4.0

C.9.5 Changed Traceables in 4.4.0

Number	Heading
[TPS_STDT_00006]	Applying Expression Pattern
[TPS_STDT_00021]	Specialization of BlueprintFormula
[TPS_STDT_00045]	Transferring Objects in General
[TPS_STDT_00047]	Ignore Blueprint Attributes in Non Blueprints
[TPS_STDT_00086]	Specify a name pattern or a blueprint value in blueprints

Table C.24: Changed Traceables in 4.4.0

C.9.6 Deleted Traceables in 4.4.0

none

C.10 Change History R19-11

C.10.1 Added Constraints in 19-11

none

C.10.2 Changed Constraints in 19-11

Number	Heading
[constr_2556]	No Blueprint Motivated VariationPoints in AUTOSAR Descriptions
[constr_2569]	Purely Blueprint Motivated VariationPoints

Table C.25: Changed Constraints in 19-11

C.10.3 Deleted Constraints in 19-11

none

C.10.4 Added Traceables in 19-11

none

C.10.5 Changed Traceables in 19-11

Number	Heading
[TPS_STDT_00006]	Applying Expression Pattern
[TPS_STDT_00021]	Specialization of BlueprintFormula
[TPS_STDT_00028]	Resolving VariationPoint in Blueprints
[TPS_STDT_00030]	Blueprint of VariationPoint
[TPS_STDT_00044]	Transferring VariationPoint
[TPS_STDT_00046]	Configuration dependent properties
[TPS_STDT_00048]	Express Decisions when Deriving Objects

Table C.26: Changed Traceables in 19-11

C.10.6 Deleted Traceables in 19-11

none

C.11 Change History R20-11

C.11.1 Added Constraints in R20-11

none

C.11.2 Changed Constraints in R20-11

Number	Heading
[constr_2540]	Tagged text category

Table C.27: Changed Constraints in R20-11

C.11.3 Deleted Constraints in R20-11

none

C.11.4 Added Traceables in R20-11

none

C.11.5 Changed Traceables in R20-11

none

C.11.6 Deleted Traceables in R20-11

none

D Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

Class	***ARElement*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, ApplicationPartition, *AutosarDataType*, *BaseType*, BlueprintMappingSet, BswEntryRelationshipSet, BswModuleDescription, BswModuleEntry, BuildActionManifest, CalibrationParameterValueSet, ClientIdDefinitionSet, ClientServerInterfaceToBswModuleEntryBlueprintMapping, Collection, CompuMethod, ConsistencyNeedsBlueprintSet, ConstantSpecification, ConstantSpecificationMappingSet, CpSoftwareCluster, CpSoftwareClusterBinaryManifestDescriptor, CpSoftwareClusterMappingSet, CpSoftwareClusterResourcePool, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, CryptoServiceQueue, DataConstr, DataExchangePoint, DataTransformationSet, DataTypeMappingSet, *DiagnosticCommonElement*, DiagnosticConnection, DiagnosticContributionSet, Documentation, E2EProfileCompatibilityProps, EcucDefinitionCollection, EcucDestinationUriDefSet, EcucModuleConfigurationValues, EcucModuleDef, EcucValueCollection, EndToEndProtectionSet, EthIpProps, EthTcpIpIcmpProps, EthTcpIpProps, EvaluatedVariantSet, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FlatMap, GeneralPurposeConnection, HwCategory, HwElement, HwType, IPSecConfigProps, IPv6ExtHeaderFilterSet, *IdsCommonElement*, IdsDesign, *Implementation*, InterpolationRoutineMappingSet, J1939ControllerApplication, KeywordSet, LifeCycleInfoSet, LifeCycleStateDefinitionGroup, McFunction, McGroup, ModeDeclarationGroup, ModeDeclarationMappingSet, PhysicalDimension, PhysicalDimensionMappingSet, *PortInterface*, PortInterfaceMappingSet, PortPrototypeBlueprint, PostBuildVariantCriterion, PostBuildVariantCriterionValueSet, PredefinedVariant, RapidPrototypingScenario, SdgDef, SignalServiceTranslationPropsSet, SomeipSdClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroupTimingConfig, SomeipSdServerServiceInstanceConfig, SwAddrMethod, SwAxisType, *SwComponentType*, SwRecordLayout, SwSystemconst, SwSystemconstantValueSet, SwcBswMapping, System, SystemSignal, SystemSignalGroup, TDCpSoftwareClusterMappingSet, TcpOptionFilterSet, *TimingExtension*, TlvDataIdDefinitionSet, TransformationPropsSet, Unit, UnitGroup, ViewMapSet			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.1: ARElement

Class	**ARPackage**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	AUTOSAR package, allowing to create top level packages to structure the contained ARElements. ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package. This is an extended version of MSR's SW-SYSTEM.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	ARPackage			

arPackage	ARPackage	*	aggr	This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy.
				
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=arPackage.shortName, arPackage.variation Point.shortLabel
				vh.latestBindingTime=blueprintDerivationTime
				xml.sequenceOffset=30

element	PackageableElement	*	aggr	Elements that are part of this package
				
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=element.shortName, element.variation Point.shortLabel
				vh.latestBindingTime=systemDesignTime
				xml.sequenceOffset=20

referenceBase	ReferenceBase	*	aggr	This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=referenceBase.shortLabel xml.sequenceOffset=10

Table D.2: ARPackage

Class	**AclObjectSet**			
Package	M2::AUTOSARTemplates::GenericStructure::RolesAndRights			
Note	This meta class represents the ability to denote a set of objects for which roles and rights (access control lists) shall be defined. It basically can define the objects based on • the nature of objects • the involved blueprints • the artifact in which the objects are serialized • the definition of the object (in a definition - value pattern) • individual reference objects **Tags:**atp.recommendedPackage=AclObjectSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
aclObjectClass	ReferrableSubtypes Enum	*	attr	This specifies that the considered objects as instances of the denoted meta class.
aclScope	AclScopeEnum	1	attr	this indicates the scope of the referenced objects.
collection	Collection	0..1	ref	This indicates that the relevant objects are specified via a collection.
derivedFrom Blueprint	AtpBlueprint	*	ref	This association indicates that the considered objects are the ones being derived from the associated blueprint. **Stereotypes:** atpUriDef

▽

△

Class	**AclObjectSet**			
engineering Object	AutosarEngineering Object	*	aggr	This indicates an engineering object. The AclPermission relates to all objects in this partial model. This also implies that the other objects in this set shall be placed in the specified engineering object. Note that semantic constraints apply with respect to <<atpSplitable>>
object	Referrable	*	ref	This association applies a particular (usually small) set of objects (e.g. a singular package). Main usage is, if one does not want to create a collection specifically for access control.
objectDefinition	AtpDefinition	*	ref	This denotes an object by its definition. For example the right to manipulate the value of a particular ecuc parameter is denoted by reference to the definition of the parameter. Note that this can also be a reference to a Standard Module Definition. Therefore it is stereotyped by atpUri Def. **Stereotypes:** atpUriDef

Table D.3: AclObjectSet

Class	**AclOperation**			
Package	M2::AUTOSARTemplates::GenericStructure::RolesAndRights			
Note	This meta class represents the ability to denote a particular operation which may be performed on objects in an AUTOSAR model. **Tags:**atp.recommendedPackage=AclOperations			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
implied Operation	AclOperation	*	ref	This indicates that the related operations are also implied. Therefore the permission is also granted for this operation.

Table D.4: AclOperation

Class	**AclPermission**			
Package	M2::AUTOSARTemplates::GenericStructure::RolesAndRights			
Note	This meta class represents the ability to represent permissions granted on objects in an AUTOSAR model. **Tags:**atp.recommendedPackage=AclPermissions			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
aclContext	NameToken	*	attr	This attribute is intended to specify the context under which the AclPemission is applicable. The values are subject to mutual agreement between the involved stakeholders. For examples the values can be the names of binding times.

▽

△

Class	**AclPermission**			
aclObject	AclObjectSet	*	ref	This denotes an object to which the AclPermission applies.
aclOperation	AclOperation	*	ref	This denotes an operation which is granted by the given AclPermission.
aclRole	AclRole	*	ref	This denotes the role (individual or even organization) for which the AclPermission. is granted.
aclScope	AclScopeEnum	1	attr	This indicates the scope of applied permissions: explicit, descendant, dependent;

Table D.5: AclPermission

Class	**AclRole**			
Package	M2::AUTOSARTemplates::GenericStructure::RolesAndRights			
Note	This meta class represents the ability to specify a particular role which is used to grant access rights to AUTOSAR model. The purpose of this meta-class is to support the mutual agreements between the involved parties. **Tags:**atp.recommendedPackage=AclRoles			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
ldapUrl	UriString	0..1	attr	This is an URL which allows to represent users or organizations taking the particular role.

Table D.6: AclRole

Class	**AliasNameSet**			
Package	M2::AUTOSARTemplates::CommonStructure::FlatMap			
Note	This meta-class represents a set of AliasNames. The AliasNameSet can for example be an input to the A2L-Generator. **Tags:**atp.recommendedPackage=AliasNameSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
aliasName	AliasNameAssignment	1..*	aggr	AliasNames contained in the AliasNameSet. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=aliasName.shortLabel, aliasName.variation Point.shortLabel vh.latestBindingTime=preCompileTime

Table D.7: AliasNameSet

Class	***ApplicationDataType*** (abstract)
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes

▽

△

Class	***ApplicationDataType*** (abstract)			
Note	ApplicationDataType defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake. An ApplicationDataType represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianess, etc. It should be possible to model the application level aspects of a VFB system by using ApplicationData Types only.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*ApplicationCompositeDataType*, ApplicationPrimitiveDataType			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.8: ApplicationDataType

Class	**ArgumentDataPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
direction	ArgumentDirection Enum	0..1	attr	This attribute specifies the direction of the argument prototype.
serverArgument ImplPolicy	ServerArgumentImpl PolicyEnum	0..1	attr	This defines how the argument type of the servers RunnableEntity is implemented. If the attribute is not defined this has the same semantics as if the attribute is set to the value useArgumentType for primitive arguments and structures.

Table D.9: ArgumentDataPrototype

Class	***AtomicSwComponentType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	An atomic software component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Subclasses	ApplicationSwComponentType, ComplexDeviceDriverSwComponentType, EcuAbstractionSwComponent Type, NvBlockSwComponentType, SensorActuatorSwComponentType, ServiceProxySwComponent Type, ServiceSwComponentType			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
internalBehavior	SwcInternalBehavior	0..1	aggr	The SwcInternalBehaviors owned by an AtomicSw ComponentType can be located in a different physical file. Therefore the aggregation is <<atpSplitable>>. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalBehavior.shortName, internal Behavior.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	***AtomicSwComponentType*** (abstract)			
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the AtomicSw ComponentType. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=symbolProps.shortName

Table D.10: AtomicSwComponentType

Enumeration	**BindingTimeEnum**
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling
Note	This enumerator specifies the applicable binding times for the pre build variation points.
Literal	***Description***
codeGeneration Time	• Coding by hand, based on requirements document. • Tool based code generation, e.g. from a model. • The model may contain variants. • Only code for the selected variant(s) is actually generated. **Tags:**atp.EnumerationLiteralIndex=0
linkTime	Configure what is included in object code, and what is omitted Based on which variant(s) are selected E.g. for modules that are delivered as object code (as opposed to those that are delivered as source code) **Tags:**atp.EnumerationLiteralIndex=1
preCompileTime	This is typically the C-Preprocessor. Exclude parts of the code from the compilation process, e.g., because they are not required for the selected variant, because they are incompatible with the selected variant, because they require resources that are not present in the selected variant. Object code is only generated for the selected variant(s). The code that is excluded at this stage code will not be available at later stages. **Tags:**atp.EnumerationLiteralIndex=2
systemDesignTime	• Designing the VFB. • Software Component types (PortInterfaces). • SWC Prototypes and the Connections between SWCprototypes. • Designing the Topology • ECUs and interconnecting Networks • Designing the Communication Matrix and Data Mapping **Tags:**atp.EnumerationLiteralIndex=3

Table D.11: BindingTimeEnum

Class	<<atpMixedString>> **BlueprintFormula**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::BlueprintFormula			
Note	This class express the extension of the Formula Language to provide formalized blueprint-Value resp. blueprintCondition.			
Base	*ARObject*, *FormulaExpression*, *SwSystemconstDependentFormula*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ecuc	EcucDefinitionElement	1	ref	The EcucDefinitionElement serves as a argument for the formular.

▽

△

Class	<<atpMixedString>> **BlueprintFormula**			
verbatim	MultiLanguageVerbatim	1	aggr	This represents an informal term in the expression as verbatim text. Note that the result of this is same as formula keyword "undefined".

Table D.12: BlueprintFormula

Class	**BlueprintMapping**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::BlueprintDedicated::Generic Blueprint			
Note	This meta-class represents the ability to map two an object and its blueprint.			
Base	*ARObject*, *AtpBlueprintMapping*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blueprint	AtpBlueprint	1	ref	This represents the mapped blueprint.
derivedObject	AtpBlueprintable	1	ref	This represents the object which was derived from the blueprint.

Table D.13: BlueprintMapping

Class	**BswInternalBehavior**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	Specifies the behavior of a BSW module or a BSW cluster w.r.t. the code entities visible by the BSW Scheduler. It is possible to have several different BswInternalBehaviors referring to the same BswModule Description.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *InternalBehavior*, *Multilanguage Referrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arTypedPer Instance Memory	VariableDataPrototype	*	aggr	Defines an AUTOSAR typed memory-block that needs to be available for each instance of the Basic Software Module. The aggregation of arTypedPerInstanceMemory is subject to variability with the purpose to support variability in the Basic Software Module's implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arTypedPerInstanceMemory.shortName, ar TypedPerInstanceMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
bswPerInstance MemoryPolicy	BswPerInstance MemoryPolicy	*	aggr	Policy for a arTypedPerInstanceMemory The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswPerInstanceMemoryPolicy, bswPer InstanceMemoryPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**BswInternalBehavior**			
clientPolicy	BswClientPolicy	*	aggr	Policy for a requiredClientServerEntry. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=clientPolicy, clientPolicy.variationPoint.short Label vh.latestBindingTime=preCompileTime

distinguished Partition	BswDistinguished Partition	*	aggr	Indicates an abstract partition context in which the enclosing BswModuleEntity can be executed.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=distinguishedPartition.shortName, distinguishedPartition.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=60

entity	BswModuleEntity	*	aggr	A code entity for which the behavior is described
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=entity.shortName, entity.variationPoint.short Label
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=5

event	BswEvent	*	aggr	An event required by this module behavior.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=event.shortName, event.variationPoint.short Label
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=10

exclusiveArea Policy	BswExclusiveArea Policy	*	aggr	Policy for an ExclusiveArea in this BswInternalBehavior. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveAreaPolicy, exclusiveArea Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
includedData TypeSet	IncludedDataTypeSet	*	aggr	The includedDataTypeSet is used by a basic software module for its implementation. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=includedDataTypeSet

internal TriggeringPoint	BswInternalTriggering Point	*	aggr	An internal triggering point.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=internalTriggeringPoint.shortName, internal TriggeringPoint.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=2

▽

△

Class	BswInternalBehavior			
internal TriggeringPoint Policy	BswInternalTriggering PointPolicy	*	aggr	Policy for an internalTriggeringPoint in this BswInternal Behavior.. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalTriggeringPointPolicy, internal TriggeringPointPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

modeReceiver Policy	BswModeReceiver Policy	*	aggr	Implementation policy for the reception of mode switches.
				
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=modeReceiverPolicy, modeReceiver Policy.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=25

modeSender Policy	BswModeSenderPolicy	*	aggr	Implementation policy for providing a mode group.
				
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=modeSenderPolicy, modeSender Policy.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=20

parameterPolicy	BswParameterPolicy	*	aggr	Policy for a perInstanceParameter in this BswInternal Behavior. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=parameterPolicy, parameterPolicy.variation Point.shortLabel vh.latestBindingTime=preCompileTime
perInstance Parameter	ParameterData Prototype	*	aggr	Describes a read only memory object containing characteristic value(s) needed by this BswInternal Behavior. The role name perInstanceParameter is chosen in analogy to the similar role in the context of SwcInternal Behavior. In contrast to constantMemory, this object is not allocated locally by the module's code, but by the BSW Scheduler and it is accessed from the BSW module via the BSW Scheduler API. The main use case is the support of software emulation of calibration data. The aggregation is subject to variability with the purpose to support implementation variants. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=perInstanceParameter.shortName, per InstanceParameter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45

receptionPolicy	BswDataReception Policy	*	aggr	Data reception policy for inter-partition and/or inter-core communication.
				
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=receptionPolicy, receptionPolicy.variation Point.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=55

▽

△

Class	**BswInternalBehavior**			
releasedTrigger Policy	BswReleasedTrigger Policy	*	aggr	Policy for a releasedTrigger. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=releasedTriggerPolicy, releasedTrigger Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

schedulerName Prefix	BswSchedulerName Prefix	*	aggr	Optional definition of one or more prefixes to be used for the BswScheduler.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=schedulerNamePrefix.shortName, scheduler NamePrefix.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=50

sendPolicy	BswDataSendPolicy	*	aggr	Policy for a providedData. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=sendPolicy, sendPolicy.variationPoint.short Label vh.latestBindingTime=preCompileTime
service Dependency	BswService Dependency	*	aggr	Defines the requirements on AUTOSAR Services for a particular item. The aggregation is subject to variability with the purpose to support the conditional existence of ServiceNeeds. The aggregation is splitable in order to support that ServiceNeeds might be provided in later development steps. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceDependency.ident.shortName, serviceDependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

triggerDirect Implementation	BswTriggerDirect Implementation	*	aggr	Specifies a trigger to be directly implemented via OS calls.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=triggerDirectImplementation, triggerDirect Implementation.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=15

variationPoint Proxy	VariationPointProxy	*	aggr	Proxy of a variation points in the C/C++ implementation. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=variationPointProxy.shortName

Table D.14: BswInternalBehavior

Class	**BswModuleDescription**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswOverview			
Note	Root element for the description of a single BSW module or BSW cluster. In case it describes a BSW module, the short name of this element equals the name of the BSW module. **Tags:**atp.recommendedPackage=BswModuleDescriptions			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

bswModule Dependency	BswModuleDependency	*	aggr	Describes the dependency to another BSW module.
				
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=bswModuleDependency.shortName, bsw ModuleDependency.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=20

bswModule Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the BSW module.
				
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=bswModuleDocumentation, bswModule Documentation.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=6

expectedEntry	BswModuleEntry	*	ref	Indicates an entry which is required by this module. Replacement of outgoingCallback / requiredEntry. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=expectedEntry.bswModuleEntry, expected Entry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
implemented Entry	BswModuleEntry	*	ref	Specifies an entry provided by this module which can be called by other modules. This includes "main" functions, interrupt routines, and callbacks. Replacement of providedEntry / expectedCallback. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=implementedEntry.bswModuleEntry, implementedEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
internalBehavior	BswInternalBehavior	*	aggr	The various BswInternalBehaviors associated with a Bsw ModuleDescription can be distributed over several physical files. Therefore the aggregation is <<atp Splitable>>. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=internalBehavior.shortName xml.sequenceOffset=65
moduleId	PositiveInteger	0..1	attr	Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen. **Tags:**xml.sequenceOffset=5

▽

△

Class	**BswModuleDescription**			

providedClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module provides a client server entry which can be called from another parition or core.This entry is declared locally to this context and will be connected to the requiredClientServerEntry of another or the same module via the configuration of the BSW Scheduler.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=providedClientServerEntry.shortName, providedClientServerEntry.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=45

providedData	VariableDataPrototype	*	aggr	Specifies a data prototype provided by this module in order to be read from another partition or core.The providedData is declared locally to this context and will be connected to the requiredData of another or the same module via the configuration of the BSW Scheduler.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=providedData.shortName, provided Data.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=55

providedMode Group	ModeDeclarationGroup Prototype	*	aggr	A set of modes which is owned and provided by this module or cluster. It can be connected to the required ModeGroups of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with modes provided via ports by an associated ServiceSwComponentType, EcuAbstraction SwComponentType or ComplexDeviceDriverSw ComponentType.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=providedModeGroup.shortName, provided ModeGroup.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=25

releasedTrigger	Trigger	*	aggr	A Trigger released by this module or cluster. It can be connected to the requiredTriggers of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with Triggers provided via ports by an associated ServiceSwComponentType, Ecu AbstractionSwComponentType or ComplexDeviceDriver SwComponentType.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=releasedTrigger.shortName, released Trigger.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=35

requiredClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module requires a client server entry which can be implemented on another parition or core.This entry is declared locally to this context and will be connected to the providedClientServerEntry of another or the same module via the configuration of the BSW Scheduler. ▽

▽

△

Class	BswModuleDescription			

				△
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=requiredClientServerEntry.shortName, requiredClientServerEntry.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=50

requiredData	VariableDataPrototype	*	aggr	Specifies a data prototype required by this module in oder to be provided from another partition or core.The required Data is declared locally to this context and will be connected to the providedData of another or the same module via the configuration of the BswScheduler.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=requiredData.shortName, required Data.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=60

requiredMode Group	ModeDeclarationGroup Prototype	*	aggr	Specifies that this module or cluster depends on a certain mode group. The requiredModeGroup is local to this context and will be connected to the providedModeGroup of another module or cluster via the configuration of the BswScheduler.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=requiredModeGroup.shortName, required ModeGroup.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=30

requiredTrigger	Trigger	*	aggr	Specifies that this module or cluster reacts upon an external trigger.This requiredTrigger is declared locally to this context and will be connected to the providedTrigger of another module or cluster via the configuration of the BswScheduler.
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=requiredTrigger.shortName, required Trigger.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=40

Table D.15: BswModuleDescription

Class	**BswModuleEntry**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces			
Note	This class represents a single API entry (C-function prototype) into the BSW module or cluster. The name of the C-function is equal to the short name of this element with one exception: In case of multiple instances of a module on the same CPU, special rules for "infixes" apply, see description of class BswImplementation. **Tags:**atp.recommendedPackage=BswModuleEntrys			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**BswModuleEntry**			
argument (ordered)	SwServiceArg	*	aggr	An argument belonging to this BswModuleEntry. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=45
bswEntryKind	BswEntryKindEnum	0..1	attr	This describes whether the entry is concrete or abstract. If the attribute is missing the entry is considered as concrete. **Tags:**xml.sequenceOffset=40
callType	BswCallType	1	attr	The type of call associated with this service. **Tags:**xml.sequenceOffset=25
execution Context	BswExecutionContext	1	attr	Specifies the execution context which is required (in case of entries into this module) or guaranteed (in case of entries called from this module) for this service. **Tags:**xml.sequenceOffset=30
function Prototype Emitter	NameToken	0..1	attr	This attribute is used to control the generation of function prototypes. If set to "RTE", the RTE generates the function prototypes in the Module Interlink Header File.
isReentrant	Boolean	1	attr	Reentrancy from the viewpoint of function callers: • True: Enables the service to be invoked again, before the service has finished. • False: It is prohibited to invoke the service again before is has finished. **Tags:**xml.sequenceOffset=15
isSynchronous	Boolean	1	attr	Synchronicity from the viewpoint of function callers: • True: This calls a synchronous service, i.e. the service is completed when the call returns. • False: The service (on semantical level) may not be complete when the call returns. **Tags:**xml.sequenceOffset=20
returnType	SwServiceArg	0..1	aggr	The return type belonging to this bswModuleEntry. **Tags:**xml.sequenceOffset=40
role	Identifier	0..1	attr	Specifies the role of the entry in the given context. It shall be equal to the standardized name of the service call, especially in cases where no ServiceIdentifier is specified, e.g. for callbacks. Note that the ShortName is not always sufficient because it maybe vendor specific (e.g. for callbacks which can have more than one instance). **Tags:**xml.sequenceOffset=10
serviceId	PositiveInteger	0..1	attr	Refers to the service identifier of the Standardized Interfaces of AUTOSAR basic software. For non-standardized interfaces, it can optionally be used for proprietary identification. **Tags:**xml.sequenceOffset=5
swServiceImpl Policy	SwServiceImplPolicy Enum	1	attr	Denotes the implementation policy as a standard function call, inline function or macro. This has to be specified on interface level because it determines the signature of the call. **Tags:**xml.sequenceOffset=35

Table D.16: BswModuleEntry

Class	**BuildAction**			
Package	M2::AUTOSARTemplates::GenericStructure::BuildActionManifest			
Note	This meta-class represents the ability to specify a build action.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *BuildActionEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
createdData	BuildActionIoElement	*	aggr	This represents the artifacts which are cated by the processor.
followUpAction	BuildAction	*	ref	This association specifies a set of follow up actions. **Tags:**xml.sequenceOffset=-80
inputData	BuildActionIoElement	*	aggr	This represents the artifacts which are read by the processor.
modifiedData	BuildActionIoElement	*	aggr	This denotes the data which are modifed by the action.
predecessor Action	BuildAction	*	ref	This association specifies a set of predecessors. These actions shall be finished before but necessarily immediately after the given action.. These actions need to be performed in the specified order. **Tags:**xml.sequenceOffset=-90
required Environment	BuildActionEnvironment	1	ref	This represents the environment which is required to use the specified Processor.

Table D.17: BuildAction

Class	**BuildActionEnvironment**			
Package	M2::AUTOSARTemplates::GenericStructure::BuildActionManifest			
Note	This meta-class represents the ability to specify a build action environment.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
sdg	Sdg	*	aggr	This represents a general data structure intended to denote parameters for the BuildActionEnvironment.

Table D.18: BuildActionEnvironment

Class	**BuildActionManifest**			
Package	M2::AUTOSARTemplates::GenericStructure::BuildActionManifest			

Note	This meta-class represents the ability to specify a manifest for processing artifacts. An example use case is the processing of ECUC parameter values.
	Tags:
	atp.recommendedPackage=BuildActionManifests
	xml.globalElement=false			

Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
buildAction	BuildAction	*	aggr	This represents a particular action in the build chain. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime

▽

△

Class	**BuildActionManifest**			
buildAction Environment	BuildActionEnvironment	*	aggr	This represents a build action environment. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime
dynamicAction	BuildAction	*	ref	This denots an Action which is to be executed as part of the dynamic action set.
startAction	BuildAction	*	ref	This specifies the list of actions to be performed at the beginning of the process. **Tags:**xml.sequenceOffset=-90
tearDownAction	BuildAction	*	ref	This specifies the set of action which shall be performed after all other actions in the manifest were performed. **Tags:**xml.sequenceOffset=-80

Table D.19: BuildActionManifest

Class	**ClientServerInterfaceToBswModuleEntryBlueprintMapping**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::ClientServerInterfaceToBsw ModuleEntryMapping			
Note	This represents a mapping between one ClientServerInterface blueprint and BswModuleEntry blueprint in order to express the intended implementation of ClientServerOperations by specific BswModuleEntries under consideration of PortDefinedArguments. Such a mapping enables the formal check whether the number of arguments and the data types of arguments of the operation + additional PortDefined Arguments matches the signature of the BswModuleEntry. **Tags:**atp.recommendedPackage=BlueprintMappingSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
clientServer Interface	ClientServerInterface	1	ref	The referenced ClientServerInterface represents the client server interface the mapping is dedicated to.
operation Mapping	ClientServerOperation BlueprintMapping	1..*	aggr	This specifies the operations used in the mapping between the ClientServerInterface and the BswModule Entry. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
portDefined Argument Blueprint (ordered)	PortDefinedArgument Blueprint	*	aggr	This specifies the PortDefinedArguments used in the mapping between the ClientServerInterface and the Bsw ModuleEntry. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table D.20: ClientServerInterfaceToBswModuleEntryBlueprintMapping

Class	**ClientServerOperation**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An operation declared within the scope of a client/server interface.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	**ClientServerOperation**			
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this ClientServerOperation **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime
diagArgIntegrity	Boolean	0..1	attr	This attribute shall only be used in the implementation of diagnostic routines to support the case where input and output arguments are allocated in a shared buffer and might unintentionally overwrite input arguments by tentative write operations to output arguments. This situation can happen during sliced execution or while output parameters are arrays (call by reference). The value true means that the ClientServerOperation is aware of the usage of a shared buffer and takes precautions to avoid unintentional overwrite of input arguments. If the attribute does not exist or is set to false the Client ServerOperation does not have to consider the usage of a shared buffer.
possibleError	ApplicationError	*	ref	Possible errors that may by raised by the referring operation.

Table D.21: ClientServerOperation

Class	**Collection**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ElementCollection			
Note	This meta-class specifies a collection of elements. A collection can be utilized to express additional aspects for a set of elements. Note that Collection is an ARElement. Therefore it is applicable e.g. for EvaluatedVariant, even if this is not obvious. Usually the category of a Collection is "SET". On the other hand, a Collection can also express an arbitrary relationship between elements. This is denoted by the category "RELATION" (see also [TPS_GST_00347]). In this case the collection represents an association from "sourceElement" to "targetElement" in the role "role". **Tags:**atp.recommendedPackage=Collections			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
autoCollect	AutoCollectEnum	0..1	attr	This attribute reflects how far the referenced objects are part of the collection. **Tags:**xml.sequenceOffset=20
collected Instance	AtpFeature	*	iref	This instance ref supports the use case that a particular instance is part of the collection. **Tags:**xml.sequenceOffset=60 **InstanceRef implemented by:**AnyInstanceRef
element	Identifiable	*	ref	This is an element in the collection. Note that Collection itself is collectable. Therefore collections can be nested. In case of category="RELATION" this represents the target end of the relation. **Tags:**xml.sequenceOffset=40

▽

△

Class	**Collection**			
elementRole	Identifier	0..1	attr	This attribute allows to denote a particular role of the collection. Note that the applicable semantics shall be mutually agreed between the two parties. In particular it denotes the role of element in the context of sourceElement. **Tags:**xml.sequenceOffset=30
sourceElement	Identifiable	*	ref	Only if Category = "RELATION". This represents the source of a relation. **Tags:**xml.sequenceOffset=50
sourceInstance	AtpFeature	*	iref	Only if Category = "RELATION". This represents the source instance of a relation. **Tags:**xml.sequenceOffset=70 **InstanceRef implemented by:**AnyInstanceRef

Table D.22: Collection

Class	**CompuMethod**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:**atp.recommendedPackage=CompuMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
compuInternal ToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Tags:**xml.sequenceOffset=80
compuPhysTo Internal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Tags:**xml.sequenceOffset=90
displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:**xml.sequenceOffset=20
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:**xml.sequenceOffset=30

Table D.23: CompuMethod

Class	**CompuScale**
Package	M2::MSR::AsamHdo::ComputationMethod
Note	This meta-class represents the ability to specify one segment of a segmented computation method.
Base	*ARObject*

▽

△

Class	**CompuScale**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compuInverse Value	CompuConst	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se. **Tags:**xml.sequenceOffset=60

compuScale Contents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale.
				
				Tags:
				xml.roleElement=false
				xml.roleWrapperElement=false
				xml.sequenceOffset=70
				xml.typeElement=false
				xml.typeWrapperElement=false

desc	MultiLanguageOverview Paragraph	0..1	aggr	<desc> represents a general but brief description of the object in question. **Tags:**xml.sequenceOffset=30
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
mask	PositiveInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the COMPU-SCALE elements. **Tags:**xml.sequenceOffset=35
shortLabel	Identifier	0..1	attr	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. **Tags:**xml.sequenceOffset=20
symbol	CIdentifier	0..1	attr	The symbol, if provided, is used by code generators to get a C identifier for the CompuScale. The name will be used as is for the code generation, therefore it needs to be unique within the generation context. **Tags:**xml.sequenceOffset=25
upperLimit	Limit	0..1	attr	This specifies the upper limit of a of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

Table D.24: CompuScale

Class	ConsistencyNeeds			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
Note	This meta-class represents the ability to define requirements on the implicit communication behavior.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dpgDoesNotRequireCoherency	DataPrototypeGroup	*	aggr	This group of VariableDataPrototypes does not require coherency with respect to the implicit communication behavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dpgDoesNotRequireCoherency.shortName, dpgDoesNotRequireCoherency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dpgRequiresCoherency	DataPrototypeGroup	*	aggr	This group of VariableDataPrototypes requires coherency with respect to the implicit communication behavior, i.e. all read and write access to VariableDataPrototypes in the DataPrototypeGroup by the RunnableEntitys of the RunnableEntityGroup need to be handled in a coherent manner. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dpgRequiresCoherency.shortName, dpgRequiresCoherency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
regDoesNotRequireStability	RunnableEntityGroup	*	aggr	This group of RunnableEntities does not require stability with respect to the implicit communication behavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=regDoesNotRequireStability.shortName, regDoesNotRequireStability.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
regRequiresStability	RunnableEntityGroup	*	aggr	This group of RunnableEntities requires stability with respect to the implicit communication behavior, i.e. all read and write access to VariableDataPrototypes in the DataPrototypeGroup by the RunnableEntitys of the RunnableEntityGroup need to be handled in a stable manner. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=regRequiresStability.shortName, regRequiresStability.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table D.25: ConsistencyNeeds

Class	ConsistencyNeedsBlueprintSet			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::BlueprintDedicated::ConsistencyNeedsBlueprintSet			
Note	This meta class represents the ability to specify a set of blueprint for ConsistencyNeeds. **Tags:**atp.recommendedPackage=ConsistencyNeedsBlueprintSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**ConsistencyNeedsBlueprintSet**			
consistency Needs	ConsistencyNeeds	*	aggr	This represents a particular blueprint of consistency Needs. Note that it is **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table D.26: ConsistencyNeedsBlueprintSet

Class	**DataConstr**			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to specify constraints on data. **Tags:**atp.recommendedPackage=DataConstrs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

dataConstrRule	DataConstrRule	*	aggr	This is one particular rule within the data constraints.
				Tags:
				xml.roleElement=true
				xml.roleWrapperElement=true
				xml.sequenceOffset=30
				xml.typeElement=false
				xml.typeWrapperElement=false

Table D.27: DataConstr

Class	**DataFormatTailoring**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	This class collects all rules that tailor the AUTOSAR templates for a specific data exchange point.			
Base	*ARObject*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
classTailoring	ClassTailoring	*	aggr	Specification of tailorings of Meta Classes **Tags:**xml.sequenceOffset=10
constraint Tailoring	ConstraintTailoring	*	aggr	Specification of tailorings of Constraints that are not explicitly owned by any Meta-Class **Tags:**xml.sequenceOffset=20

Table D.28: DataFormatTailoring

Class	**DataPrototypeGroup**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
Note	This meta-class represents the ability to define a collection of DataPrototypes that are subject to the formal definition of implicit communication behavior. The definition of the collection can be nested.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	**DataPrototypeGroup**			
dataPrototype Group	DataPrototypeGroup	*	iref	This represents the ability to define nested groups of VariableDataPrototypes. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime **InstanceRef implemented by:**InnerDataPrototypeGroup InCompositionInstanceRef
implicitData Access	VariableDataPrototype	*	iref	This represents a collection of VariableDataPrototypes that belong to the enclosing DataPrototypeGroup **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime **InstanceRef implemented by:**VariableDataPrototypeIn CompositionInstanceRef

Table D.29: DataPrototypeGroup

Class	**DataTypeMappingSet**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	This class represents a list of mappings between ApplicationDataTypes and ImplementationDataTypes. In addition, it can contain mappings between ImplementationDataTypes and ModeDeclarationGroups. **Tags:**atp.recommendedPackage=DataTypeMappingSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataTypeMap	DataTypeMap	*	aggr	This is one particular association between an Application DataType and its AbstractImplementationDataType.
modeRequest TypeMap	ModeRequestTypeMap	*	aggr	This is one particular association between an Mode DeclarationGroup and its AbstractImplementationData Type.

Table D.30: DataTypeMappingSet

Class	**Documentation**			
Package	M2::AUTOSARTemplates::GenericStructure::DocumentationOnM1			
Note	This meta-class represents the ability to handle a so called standalone documentation. Standalone means, that such a documentation is not embedded in another ARElement or identifiable object. The standalone documentation is an entity of its own which denotes its context by reference to other objects and instances. **Tags:**atp.recommendedPackage=Documentations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
context	DocumentationContext	*	aggr	This is the context of the particular documentation.
documentation Content	PredefinedChapter	0..1	aggr	This is the content of the documentation related to the specified contexts. **Tags:**xml.sequenceOffset=200

Table D.31: Documentation

Class	<<atpMixed>> **DocumentationBlock**			
Package	M2::MSR::Documentation::BlockElements			
Note	This class represents a documentation block. It is made of basic text structure elements which can be displayed in a table cell.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defList	DefList	0..1	aggr	This represents a definition list in the documentation block. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=40
figure	MlFigure	0..1	aggr	This represents a figure in the documentation block. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=70
formula	MlFormula	0..1	aggr	This is a formula in the definition block. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=60
labeledList	LabeledList	0..1	aggr	This represents a labeled list. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=50
list	List	0..1	aggr	This represents numbered or unnumbered list. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=30
msrQueryP2	MsrQueryP2	0..1	aggr	This represents automatically contributed contents provided by an msrquery in the context of Documentation Block.
note	Note	0..1	aggr	This represents a note in the text flow. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=80
p	MultiLanguage Paragraph	0..1	aggr	This is one particular paragraph. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=10
structuredReq	StructuredReq	0..1	aggr	This aggregation supports structured requirements embedded in a documentation block. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=100

▽

△

Class	<<atpMixed>> **DocumentationBlock**			
trace	TraceableText	0..1	aggr	This represents traceable text in the documentation block. This allows to specify requirements/constraints in any documentation block. The kind of the trace is specified in the category. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=90
verbatim	MultiLanguageVerbatim	0..1	aggr	This represents one particular verbatim text. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=20

Table D.32: DocumentationBlock

Class	<<atpVariation>> ***EcucAbstractStringParamDef*** (abstract)			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Abstract class that is used to collect the common properties for StringParamDefs, LinkerSymbolDef, FunctionNameDef and MultilineStringParamDefs. atpVariation: [RS_ECUC_00083] **Tags:**vh.latestBindingTime=codeGenerationTime			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	EcucFunctionNameDef, EcucLinkerSymbolDef, EcucMultilineStringParamDef, EcucStringParamDef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultValue	VerbatimString	0..1	attr	Default value of the string configuration parameter.
maxLength	PositiveInteger	0..1	attr	Max length allowed for this string.
minLength	PositiveInteger	0..1	attr	Min length allowed for this string.
regular Expression	RegularExpression	0..1	attr	This represents the regular expression which shall be used to validate the string parameter value.

Table D.33: EcucAbstractStringParamDef

Class	**EcucBooleanParamDef**			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Configuration parameter type for Boolean. Allowed values are true and false. **Tags:**xml.sequenceOffset=0			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultValue	Boolean	0..1	attr	Default value of the boolean configuration parameter. atpVariation: [RS_ECUC_00083] **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=codeGenerationTime

Table D.34: EcucBooleanParamDef

Class	**EcucChoiceReferenceDef**			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Specify alternative references where in the ECU Configuration description only one of the specified references will actually be used.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractInternalReferenceDef*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
destination	EcucContainerDef	*	ref	All the possible parameter containers for the reference are specified. **Stereotypes:** atpUriDef

Table D.35: EcucChoiceReferenceDef

Class	***EcucContainerDef*** (abstract)			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Base class used to gather common attributes of configuration container definitions.			
Base	*ARObject*, *AtpDefinition*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	EcucChoiceContainerDef, EcucParamConfContainerDef			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
destinationUri	EcucDestinationUriDef	*	ref	Several destinationUris can be defined for an EcucContainerDef. With such destinationUris an EcucContainerDef is applicable for several EcucUriReferenceDefs. **Stereotypes:** atpUriDef
multiplicityConfigClass	EcucMultiplicityConfigurationClass	*	aggr	Specifies which MultiplicityConfigurationClass this container is available for which ConfigurationVariant. This aggregation is optional if the surrounding EcucModuleDef has the Category STANDARDIZED_MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_MODULE_DEFINITION and if the upperMultiplicity is greater than the lowerMultiplicity then this aggregation is mandatory. **Tags:**xml.namePlural=MULTIPLICITY-CONFIG-CLASSES
postBuildVariantMultiplicity	Boolean	0..1	attr	Indicates if a container may have different number of instances in different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.
requiresIndex	Boolean	0..1	attr	Used to define whether the value element for this definition shall be provided with an index.

Table D.36: EcucContainerDef

Class	**EcucContainerValue**			
Package	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
Note	Represents a Container definition in the ECU Configuration Description.			
Base	*ARObject*, *EcucIndexableValue*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	**EcucContainerValue**			
definition	EcucContainerDef	0..1	ref	Reference to the definition of this Container in the ECU Configuration Parameter Definition. **Stereotypes:** atpIdentityContributor **Tags:**xml.sequenceOffset=-10
parameterValue	EcucParameterValue	*	aggr	Aggregates all ECU Configuration Values within this Container. atpVariation: [RS_ECUC_00079] **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=parameterValue.definition, parameter Value.variationPoint.shortLabel vh.latestBindingTime=postBuild
referenceValue	EcucAbstractReference Value	*	aggr	Aggregates all References with this container. atpVariation: [RS_ECUC_00079] **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=referenceValue.definition, reference Value.variationPoint.shortLabel vh.latestBindingTime=postBuild
subContainer	EcucContainerValue	*	aggr	Aggregates all sub-containers within this container. atpVariation: [RS_ECUC_00078] **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=subContainer.definition, subContainer.short Name, subContainer.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table D.37: EcucContainerValue

Class	**EcucDefinitionCollection**			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	This represents the anchor point of an ECU Configuration Parameter Definition within the AUTOSAR templates structure. **Tags:**atp.recommendedPackage=EcucDefinitionCollections			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
module	EcucModuleDef	*	ref	References to the module definitions of individual software modules.

Table D.38: EcucDefinitionCollection

Class	**EcucEnumerationParamDef**
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate
Note	Configuration parameter type for Enumeration. **Tags:**xml.sequenceOffset=0
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**EcucEnumerationParamDef**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultValue	Identifier	0..1	attr	Default value of the enumeration configuration parameter. This string needs to be one of the literals specified for this enumeration.
literal	EcucEnumerationLiteralDef	*	aggr	Aggregation on the literals used to define this enumeration parameter. This aggregation is optional if the surrounding EcucModuleDef has the category STANDARDIZED_MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_MODULE_DEFINITION then this aggregation is mandatory. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=literal.shortName

Table D.39: EcucEnumerationParamDef

Class	**EcucFloatParamDef**			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Configuration parameter type for Float. **Tags:**xml.sequenceOffset=0			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultValue	Float	0..1	attr	Default value of the float configuration parameter. atpVariation: [RS_ECUC_00083] **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=codeGenerationTime
max	Limit	0..1	attr	Max value allowed for the parameter defined. atpVariation: [RS_ECUC_00084] **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=codeGenerationTime
min	Limit	0..1	attr	Min value allowed for the parameter defined. atpVariation: [RS_ECUC_00084] **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=codeGenerationTime

Table D.40: EcucFloatParamDef

Class	**EcucIntegerParamDef**			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Configuration parameter type for Integer. **Tags:**xml.sequenceOffset=0			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**EcucIntegerParamDef**			
defaultValue	UnlimitedInteger	0..1	attr	Default value of the integer configuration parameter. atpVariation: [RS_ECUC_00083] **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=codeGenerationTime
max	UnlimitedInteger	0..1	attr	Max value allowed for the parameter defined. atpVariation: [RS_ECUC_00084] **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=codeGenerationTime
min	UnlimitedInteger	0..1	attr	Min value allowed for the parameter defined. atpVariation: [RS_ECUC_00084] **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=codeGenerationTime

Table D.41: EcucIntegerParamDef

Class	**EcucModuleDef**			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Used as the top-level element for configuration definition for Software Modules, including BSW and RTE as well as ECU Infrastructure. **Tags:**atp.recommendedPackage=EcucModuleDefs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpDefinition*, *CollectableElement*, *Ecuc DefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
apiServicePrefix	CIdentifier	0..1	attr	For CDD modules this attribute holds the apiService Prefix. The shortName of the module definition of a Complex Driver is always "Cdd". Therefore for CDD modules the module apiServicePrefix is described with this attribute.
container	EcucContainerDef	*	aggr	Aggregates the top-level container definitions of this specific module definition. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=container.shortName xml.sequenceOffset=11
postBuildVariant Support	Boolean	0..1	attr	Indicates if a module supports different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.
refinedModule Def	EcucModuleDef	0..1	ref	Optional reference from the Vendor Specific Module Definition to the Standardized Module Definition it refines. In case this EcucModuleDef has the category STANDARDIZED_MODULE_DEFINITION this reference shall not be provided. In case this EcucModuleDef has the category VENDOR_SPECIFIC_MODULE_ DEFINITION this reference is mandatory. **Stereotypes:** atpUriDef

▽

△

Class	**EcucModuleDef**			
supported ConfigVariant	EcucConfiguration VariantEnum	*	attr	Specifies which ConfigurationVariants are supported by this software module. This attribute is optional if the Ecuc ModuleDef has the category STANDARDIZED_ MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_ MODULE_DEFINITION then this attribute is mandatory.

Table D.42: EcucModuleDef

Class	**EcucNumericalParamValue**			
Package	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
Note	Holding the value which is subject to variant handling.			
Base	*ARObject*, *EcucIndexableValue*, *EcucParameterValue*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
value	Numerical	0..1	attr	Value which is subject to variant handling. atpVariation: [RS_ECUC_00080] **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table D.43: EcucNumericalParamValue

Class	***EcucParameterDef*** (abstract)			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Abstract class used to define the similarities of all ECU Configuration Parameter types defined as subclasses.			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Subclasses	*EcucAbstractStringParamDef*, EcucAddInfoParamDef, EcucBooleanParamDef, EcucEnumerationParam Def, EcucFloatParamDef, EcucIntegerParamDef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
derivation	EcucDerivation Specification	0..1	aggr	A derivation of a Configuration Parameter value can be specified by an informal Calculation Formula or by a formal language that can be used to specify the computational rules.
symbolicName Value	Boolean	0..1	attr	Specifies that this parameter's value is used, together with the aggregating container, to derive a symbolic name definition. See chapter "Representation of Symbolic Names" in Ecuc specification for more details.
withAuto	Boolean	0..1	attr	Specifies whether it shall be allowed on the value side to specify this parameter value as "AUTO". If withAuto is "true" it shall be possible to set the "isAuto Value" attribute of the respective parameter to "true". This means that the actual value will not be considered during ECU Configuration but will be (re-)calculated by the code generator and stored in the value attribute afterwards. These implicit updated values might require a ▽

▽

△

Class	*EcucParameterDef* (abstract)			
				△ re-generation of other modules which reference these values. If withAuto is "false" it shall not be possible to set the "isAutoValue" attribute of the respective parameter to "true". If withAuto is not present the default is "false".

Table D.44: EcucParameterDef

Class	*EcucParameterValue* (abstract)			
Package	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
Note	Common class to all types of configuration values.			
Base	*ARObject*, *EcucIndexableValue*			
Subclasses	EcucAddInfoParamValue, EcucNumericalParamValue, EcucTextualParamValue			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining the ECU Configuration Parameter Values. These are not intended as documentation but are mere design notes. **Tags:**xml.sequenceOffset=10
definition	EcucParameterDef	0..1	ref	Reference to the definition of this EcucParameterValue subclasses in the ECU Configuration Parameter Definition. **Stereotypes:** atpIdentityContributor **Tags:**xml.sequenceOffset=-10
isAutoValue	Boolean	0..1	attr	If withAuto is set to "true" for this parameter definition the isAutoValue can be set to "true". If isAutoValue is set to "true" the actual value will not be considered during ECU Configuration but will be (re-)calculated by the code generator and stored in the value attribute afterwards. These implicit updated values might require a re-generation of other modules which reference these values. If isAutoValue is not present the default is "false". **Tags:**xml.sequenceOffset=20

Table D.45: EcucParameterValue

Class	**EcucReferenceDef**			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Specify references within the ECU Configuration Description between parameter containers.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractInternalReferenceDef*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
destination	EcucContainerDef	0..1	ref	Exactly one reference to a parameter container is allowed as destination. **Stereotypes:** atpUriDef

Table D.46: EcucReferenceDef

Class	**EcucTextualParamValue**			
Package	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
Note	Holding a value which is not subject to variation.			
Base	*ARObject*, *EcucIndexableValue*, *EcucParameterValue*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
value	VerbatimString	0..1	attr	Value of the parameter, not subject to variant handling.

Table D.47: EcucTextualParamValue

Class	**EcucUriReferenceDef**			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Definition of reference with a destination that is specified via a destinationUri. With such a reference it is possible to define a reference to a EcucContainerDef in a different module independent from the concrete definition of the target container.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractInternalReferenceDef*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
destinationUri	EcucDestinationUriDef	0..1	ref	Any EcucContainerDef with a destinationUri that is identical to the destinationUri that is referenced here defines a valid target. **Stereotypes:** atpUriDef

Table D.48: EcucUriReferenceDef

Class	**FlatMap**			
Package	M2::AUTOSARTemplates::CommonStructure::FlatMap			
Note	Contains a flat list of references to software objects. This list is used to identify instances and to resolve name conflicts. The scope is given by the RootSwCompositionPrototype for which it is used, i.e. it can be applied to a system, system extract or ECU-extract. An instance of FlatMap may also be used in a preliminary context, e.g. in the scope of a software component before integration into a system. In this case it is not referred by a RootSwCompositionPrototype. **Tags:**atp.recommendedPackage=FlatMaps			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
instance	FlatInstanceDescriptor	1..*	aggr	A descriptor instance aggregated in the flat map. The variation point accounts for the fact, that the system in scope can be subject to variability, and thus the existence of some instances is variable. The aggregation has been made splitable because the content might be contributed by different stakeholders at different times in the workflow. Plus, the overall size might be so big that eventually it becomes more manageable if it is distributed over several files. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=instance.shortName, instance.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table D.49: FlatMap

Class	***Identifiable*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.			
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstanceFilter*, *AbstractServiceInstance*, ApplicationEndpoint, ApplicationError, ApplicationPartitionToEcuPartitionMapping, AsynchronousServerCallResultPoint, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BinaryManifestAddressableObject*, BinaryManifestItemDefinition, *BinaryManifestResource*, BinaryManifestResourceDefinition, BlockState, BswInternalTriggeringPoint, BswModuleDependency, *BuildActionEntity*, BuildActionEnvironment, CanTpAddress, CanTpChannel, CanTpNode, Chapter, ClassContentConditional, ClientIdDefinition, ClientServerOperation, Code, *CollectableElement*, ComManagementMapping, *CommConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, *CouplingPortStructuralElement*, *CpSoftwareClusterResource*, CpSoftwareClusterResourceToApplicationPartitionMapping, CpSoftwareClusterToEcuInstanceMapping, CpSoftwareClusterToResourceMapping, *CryptoServiceMapping*, DataPrototypeGroup, DataTransformation, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticFunctionInhibitSource, *DiagnosticRoutineSubfunction*, DltArgument, DltLogChannel, DltMessage, DoIpInterface, DoIpLogicAddress, DoIpRoutingActivation, ECUMapping, *EOCExecutableEntityRefAbstract*, EcuPartition, EcucContainerValue, *EcucDefinitionElement*, EcucDestinationUriDef, EcucEnumerationLiteralDef, EcucQuery, EcucValidationCondition, EndToEndProtection, EthernetWakeupSleepOnDatalineConfig, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FlatInstanceDescriptor, FlexrayArTpNode, FlexrayTpConnectionControl, FlexrayTpNode, FlexrayTpPduPool, *FrameTriggering*, GeneralParameter, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HeapUsage*, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignalTriggering, *IdentCaption*, InternalTriggeringPoint, J1939SharedAddressCluster, J1939TpNode, Keyword, LifeCycleState, LinScheduleTable, LinTpNode, Linker, MacMulticastGroup, McDataInstance, MemorySection, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, NetworkEndpoint, *NmCluster*, NmEcu, *NmNode*, NvBlockDescriptor, *PackageableElement*, ParameterAccess, PduToFrameMapping, PduTriggering, PerInstanceMemory, *PhysicalChannel*, PortElementToCommunicationResourceMapping, PortGroup, *PortInterfaceMapping*, PossibleErrorReaction, ResourceConsumption, RootSwCompositionPrototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RunnableEntityGroup, *SdgAttribute*, SdgClass, SecureCommunicationAuthenticationProps, SecureCommunicationFreshnessProps, SecurityEventContextProps, *ServerCallPoint*, *ServiceNeeds*, SignalServiceTranslationElementProps, SignalServiceTranslationEventProps, SignalServiceTranslationProps, SocketAddress, SomeipTpChannel, *SpecElementReference*, *StackUsage*, StaticSocketConnection, StructuredReq, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SwcToApplicationPartitionMapping, SwcToEcuMapping, SwcToImplMapping, SystemMapping, TDCpSoftwareClusterMapping, TDCpSoftwareClusterResourceMapping, TcpOptionFilterList, TimingCondition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, TimingModelInstance, TlsCryptoCipherSuite, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, *TransformationProps*, TransformationTechnology, Trigger, VariableAccess, VariationPointProxy, ViewMap, VlanConfig, WaitPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Tags:**xml.sequenceOffset=-40
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:**xml.sequenceOffset=-25

▽

△

Class	***Identifiable*** (abstract)			
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:**xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:**xml.sequenceOffset=-60
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:**xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:**xml.attribute=true

Table D.50: Identifiable

Class	**ImplementationDataType**			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:**atp.recommendedPackage=ImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.

▽

△

Class	**ImplementationDataType**			
isStructWith Optional Element	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE. If set to True, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=symbolProps.shortName
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table D.51: ImplementationDataType

Class	**LifeCycleInfo**			
Package	M2::AUTOSARTemplates::GenericStructure::LifeCycles			
Note	LifeCycleInfo describes the life cycle state of an element together with additional information like what to use instead			
Base	*ARObject*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
lcObject	Referrable	1	ref	Element(s) have the life cycle as described in lcState.
lcState	LifeCycleState	0..1	ref	This denotes the particular state assigned to the object. If no lcState is given then the default life cycle state of Life CycleInfoSet is assumed.
periodBegin	LifeCyclePeriod	0..1	aggr	Starting point of period in which the element has the denoted life cycle state lcState. If no periodBegin is given then the default period begin of LifeCycleInfoSet is assumed.
periodEnd	LifeCyclePeriod	0..1	aggr	Expiry date, i.e. end point of period the element does not have the denoted life cycle state lcState any more. If no periodEnd is given then the default period begin of Life CycleInfoSet is assumed.
remark	DocumentationBlock	0..1	aggr	Remark describing for example • why the element was given the specified life cycle • the semantics of useInstead
useInstead	Referrable	*	ref	Element(s) that should be used instead of the one denoted in referrable. Only relevant in case of life cycle states lcState unlike "valid". In case there are multiple references the exact semantics shall be individually described in the remark.

Table D.52: LifeCycleInfo

Class	**LifeCycleInfoSet**			
Package	M2::AUTOSARTemplates::GenericStructure::LifeCycles			
Note	This meta class represents the ability to attach a life cycle information to a particular set of elements. The information can be defined for a particular period. This supports the definition of transition plans. If no period is specified, the life cycle state applies forever. **Tags:**atp.recommendedPackage=LifeCycleInfoSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
defaultLcState	LifeCycleState	1	ref	This denotes the default life cycle state. To be used in all LifeCycleInfo elements within the LifeCycleInfoSet if no life cycle state is stated there explicitly. I.e. the defaultLcState can be overwritten in LifeCycleInfo elements.
defaultPeriodBegin	LifeCyclePeriod	0..1	aggr	Default starting point of period in which all the specified lifeCycleInfo apply. Note that the default period can be overridden for each lifeCycleInfo individually.
defaultPeriodEnd	LifeCyclePeriod	0..1	aggr	Default expiry date, i.e. default end point of period for which all specified lifeCycleInfo apply. Note that the default period can be overridden for each lifeCycleInfo individually.
lifeCycleInfo	LifeCycleInfo	*	aggr	This represents one particular life cycle information.
usedLifeCycleStateDefinitionGroup	LifeCycleStateDefinitionGroup	1	ref	This denotes the life cycle states applicable to the current life cycle info set.

Table D.53: LifeCycleInfoSet

Class	**LifeCycleState**			
Package	M2::AUTOSARTemplates::GenericStructure::LifeCycles			
Note	This meta class represents one particular state in the LifeCycle.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table D.54: LifeCycleState

Class	**LifeCycleStateDefinitionGroup**			
Package	M2::AUTOSARTemplates::GenericStructure::LifeCycles			
Note	This meta class represents the ability to define the states and properties of one particular life cycle. **Tags:**atp.recommendedPackage=LifeCycleStateDefintionGroups			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
lcState	LifeCycleState	*	aggr	Describes a single life cycle state of this life cycle state definition group.

Table D.55: LifeCycleStateDefinitionGroup

Primitive	**Limit**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			

Note	This class represents the ability to express a numerical limit. Note that this is in fact a NumericalVariation Point but has the additional attribute intervalType.
	
	Tags:
	xml.xsd.customType=LIMIT-VALUE
	xml.xsd.pattern=(0[xX][0-9a-fA-F]+)|(0[0-7]+)|(0[bB][0-1]+)|((([+\-]?[1-9][0-9]+(\.[0-9]+)?|[+\-]?[0-9](\.[0-9]+)?)([eE]([+\-]?)[0-9]+)?)|\.0|INF|-INF|NaN
	xml.xsd.type=string			

Attribute	***Type***	***Mult.***	***Kind***	***Note***
intervalType	IntervalTypeEnum	0..1	attr	This specifies the type of the interval. If the attribute is missing the interval shall be considered as "CLOSED". **Tags:**xml.attribute=true

Table D.56: Limit

Class	**ModeDeclarationGroup**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:**atp.recommendedPackage=ModeDeclarationGroups			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialMode	ModeDeclaration	0..1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred.
mode Declaration	ModeDeclaration	*	aggr	The ModeDeclarations collected in this ModeDeclaration Group. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime
modeManager ErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the ability to define the error behavior expected by the mode manager in case of errors on the mode user side (e.g. terminated mode user).
modeTransition	ModeTransition	*	aggr	This represents the avaliable ModeTransitions of the ModeDeclarationGroup
modeUserError Behavior	ModeErrorBehavior	0..1	aggr	This represents the definition of the error behavior expected by the mode user in case of errors on the mode manager side (e.g. terminated mode manager).
onTransition Value	PositiveInteger	0..1	attr	The value of this attribute shall be taken into account by the RTE generator for programmatically representing a value used for the transition between two statuses.

Table D.57: ModeDeclarationGroup

Class	***MultilanguageReferrable*** (abstract)
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders). They also may have a longName. But they are not considered to contribute substantially to the overall structure of an AUTOSAR description. In particular it does not contain other Referrables.
Base	*ARObject*, *Referrable*
Subclasses	Caption, DefItem, DocumentationContext, *Identifiable*, SdgCaption, *TraceReferrable*, *Traceable*

▽

△

Class	***MultilanguageReferrable*** (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
longName	MultilanguageLong Name	0..1	aggr	This specifies the long name of the object. Long name is targeted to human readers and acts like a headline.

Table D.58: MultilanguageReferrable

Class	**NonqueuedReceiverComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to non-queued receiving.			
Base	*ARObject*, *RPortComSpec*, *ReceiverComSpec*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
aliveTimeout	TimeValue	0..1	attr	Specify the amount of time (in seconds) after which the software component (via the RTE) needs to be notified if the corresponding data item have not been received according to the specified timing description. If the aliveTimeout attribute is 0 no timeout monitoring shall be performed.
enableUpdate	Boolean	0..1	attr	This attribute controls whether application code is entitled to check whether the value of the corresponding Variable DataPrototype has been updated.
filter	DataFilter	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement.
handleData Status	Boolean	0..1	attr	If this attribute is set to true than the Rte_IStatus API shall exist. If the attribute does not exist or is set to false then the Rte_IStatus API may still exist in response to the existence of further conditions.
handleNever Received	Boolean	0..1	attr	This attribute specifies whether for the corresponding VariableDataPrototype the "never received" flag is available. If yes, the RTE is supposed to assume that initially the VariableDataPrototype has not been received before. After the first reception of the corresponding VariableDataPrototype the flag is cleared. • If the value of this attribute is set to "true" the flag is required. • If set to "false", the RTE shall not support the "never received" functionality for the corresponding VariableDataPrototype.
handleTimeout Type	HandleTimeoutEnum	0..1	attr	This attribute controls the behavior with respect to the handling of timeouts.
initValue	ValueSpecification	0..1	aggr	Initial value to be used in case the sending component is not yet initialized. If the sender also specifies an initial value the receiver's value will be used.
timeout Substitution Value	ValueSpecification	0..1	aggr	This attribute represents the substitution value applicable in the case of a timeout.

Table D.59: NonqueuedReceiverComSpec

Class	**NonqueuedSenderComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes for non-queued sender/receiver communication (sender side)			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataFilter	DataFilter	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement.
initValue	ValueSpecification	0..1	aggr	Initial value to be sent if sender component is not yet fully initialized, but receiver needs data already.

Table D.60: NonqueuedSenderComSpec

Class	***PPortComSpec*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of a provided PortPrototype. This class will contain attributes that are valid for all kinds of provide ports, independent of client-server or sender-receiver communication patterns.			
Base	*ARObject*			
Subclasses	ModeSwitchSenderComSpec, NvProvideComSpec, ParameterProvideComSpec, *SenderComSpec*, ServerComSpec			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.61: PPortComSpec

Class	**PPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port providing a certain port interface.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
provided Interface	PortInterface	0..1	tref	The interface that this port provides. **Stereotypes:** isOfType

Table D.62: PPortPrototype

Class	**PRPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This kind of PortPrototype can take the role of both a required and a provided PortPrototype.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *Atp Feature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
provided Required Interface	PortInterface	0..1	tref	This represents the PortInterface used to type the PRPort Prototype **Stereotypes:** isOfType

Table D.63: PRPortPrototype

Class	***PackageableElement*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	This meta-class specifies the ability to be a member of an AUTOSAR package.			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ARElement*, EnumerationMappingTable, *FibexElement*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.64: PackageableElement

Class	**PortDefinedArgumentValue**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PortAPIOptions			
Note	A PortDefinedArgumentValue is passed to a RunnableEntity dealing with the ClientServerOperations provided by a given PortPrototype. Note that this is restricted to PPortPrototypes of a ClientServer Interface.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
value	ValueSpecification	0..1	aggr	Specifies the actual value.
valueType	ImplementationData Type	0..1	tref	The implementation type of this argument value. It should not be composite type or a pointer. **Stereotypes:** isOfType

Table D.65: PortDefinedArgumentValue

Class	***PortInterface*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	ClientServerInterface, *DataInterface*, ModeSwitchInterface, TriggerInterface			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
isService	Boolean	0..1	attr	This flag is set if the PortInterface is to be used for communication between an • ApplicationSwComponentType or • ServiceProxySwComponentType or • SensorActuatorSwComponentType or • ComplexDeviceDriverSwComponentType • ServiceSwComponentType • EcuAbstractionSwComponentType and a ServiceSwComponentType (namely an AUTOSAR Service) located on the same ECU. Otherwise the flag is not set.
serviceKind	ServiceProviderEnum	0..1	attr	This attribute provides further details about the nature of the applied service.

Table D.66: PortInterface

Class	***PortInterfaceMapping*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Specifies one PortInterfaceMapping to support the connection of Ports typed by two different Port Interfaces with PortInterface elements having unequal names and/or unequal semantic (resolution or range).			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ClientServerInterfaceMapping, ModeInterfaceMapping, TriggerInterfaceMapping, VariableAndParameter InterfaceMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.67: PortInterfaceMapping

Class	**PortInterfaceMappingSet**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Specifies a set of (one or more) PortInterfaceMappings. **Tags:**atp.recommendedPackage=PortInterfaceMappingSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
portInterface Mapping	PortInterfaceMapping	*	aggr	Specifies one PortInterfaceMapping to support the connection of Ports typed by two different PortInterfaces with PortInterface elements having unequal names and/or unequal semantic (resolution or range). **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime

Table D.68: PortInterfaceMappingSet

Class	***PortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/ server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.

▽

△

Class	***PortPrototype*** (abstract)			
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table D.69: PortPrototype

Class	***RPortComSpec*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of a required PortPrototype. This class will contain attributes that are valid for all kinds of require-ports, independent of client-server or sender-receiver communication patterns.			
Base	*ARObject*			
Subclasses	ClientComSpec, ModeSwitchReceiverComSpec, NvRequireComSpec, ParameterRequireComSpec, *ReceiverComSpec*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.70: RPortComSpec

Class	**RPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType

Table D.71: RPortPrototype

Class	***Referrable*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	*ARObject*			
Subclasses	*AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, Bsw VariableAccess, CouplingPortTrafficClassAssignment, DiagnosticDebounceAlgorithmProps, *Diagnostic EnvModeElement*, EthernetPriorityRegeneration, EventHandler, ExclusiveAreaNestingOrder, *Hw DescriptionEntity*, *ImplementationProps*, LinSlaveConfigIdent, ModeTransition, *MultilanguageReferrable*, PduActivationRoutingGroup, PncMappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, Socket ConnectionBundle, TimeSyncServerConfiguration, TpConnectionIdent			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100

▽

△

Class	***Referrable*** (abstract)			
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:**xml.sequenceOffset=-90

Table D.72: Referrable

Class	**RunnableEntity**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
Note	A RunnableEntity represents the smallest code-fragment that is provided by an AtomicSwComponent Type and are executed under control of the RTE. RunnableEntities are for instance set up to respond to data reception or operation invocation on a server.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *ExecutableEntity*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument (ordered)	RunnableEntity Argument	*	aggr	This represents the formal definition of a an argument to a RunnableEntity.
asynchronous ServerCall ResultPoint	AsynchronousServer CallResultPoint	*	aggr	The server call result point admits a runnable to fetch the result of an asynchronous server call. The aggregation of AsynchronousServerCallResultPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes and the variant existence of server call result points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=asynchronousServerCallResultPoint.short Name, asynchronousServerCallResultPoint.variation Point.shortLabel vh.latestBindingTime=preCompileTime
canBeInvoked Concurrently	Boolean	0..1	attr	If the value of this attribute is set to "true" the enclosing RunnableEntity can be invoked concurrently (even for one instance of the corresponding AtomicSwComponent Type). This implies that it is the responsibility of the implementation of the RunnableEntity to take care of this form of concurrency. Note that the default value of this attribute is set to "false".
dataRead Access	VariableAccess	*	aggr	RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataReadAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataReadAccess in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReadAccess.shortName, dataRead Access.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataReceive PointBy Argument	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of an argument in the function signature. ▽

▽

△

Class	**RunnableEntity**			
				△ The aggregation of dataReceivePointByArgument is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data receive points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReceivePointByArgument.shortName, dataReceivePointByArgument.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataReceive PointByValue	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of the return value. The aggregation of dataReceivePointBy Value is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of data receive points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReceivePointByValue.shortName, data ReceivePointByValue.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataSendPoint	VariableAccess	*	aggr	RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataSendPoint is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data send points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataSendPoint.shortName, dataSend Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataWrite Access	VariableAccess	*	aggr	RunnableEntity has implicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataWriteAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataWriteAccess in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataWriteAccess.shortName, dataWrite Access.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**RunnableEntity**			
external TriggeringPoint	ExternalTriggeringPoint	*	aggr	The aggregation of ExternalTriggeringPoint is subject to variability with the purpose to support the conditional existence of trigger ports or the variant existence of external triggering points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=externalTriggeringPoint.ident.shortName, externalTriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
internal TriggeringPoint	InternalTriggeringPoint	*	aggr	The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalTriggeringPoint.shortName, internal TriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
modeAccess Point	ModeAccessPoint	*	aggr	The runnable has a mode access point. The aggregation of ModeAccessPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode access points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeAccessPoint.ident.shortName, mode AccessPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
modeSwitch Point	ModeSwitchPoint	*	aggr	The runnable has a mode switch point. The aggregation of ModeSwitchPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode switch points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeSwitchPoint.shortName, modeSwitch Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
parameter Access	ParameterAccess	*	aggr	The presence of a ParameterAccess implies that a RunnableEntity needs read only access to a Parameter DataPrototype which may either be local or within a Port Prototype. The aggregation of ParameterAccess is subject to variability with the purpose to support the conditional existence of parameter ports and component local parameters as well as the variant existence of Parameter Access (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=parameterAccess.shortName, parameter Access.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**RunnableEntity**			
readLocal Variable	VariableAccess	*	aggr	The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of read LocalVariable (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=readLocalVariable.shortName, readLocal Variable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
serverCallPoint	ServerCallPoint	*	aggr	The RunnableEntity has a ServerCallPoint. The aggregation of ServerCallPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes or the variant existence of server call points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serverCallPoint.shortName, serverCall Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbol	CIdentifier	0..1	attr	The symbol describing this RunnableEntity's entry point. This is considered the API of the RunnableEntity and is required during the RTE contract phase.
waitPoint	WaitPoint	*	aggr	The WaitPoint associated with the RunnableEntity.
writtenLocal Variable	VariableAccess	*	aggr	The presence of a writtenLocalVariable implies that a RunnableEntity needs write access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of writtenLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of written LocalVariable (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=writtenLocalVariable.shortName, written LocalVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table D.73: RunnableEntity

Class	**RunnableEntityGroup**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
Note	This meta-class represents the ability to define a collection of RunnableEntities. The collection can be nested.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	**RunnableEntityGroup**			
runnableEntity	RunnableEntity	*	iref	This represents a collection of RunnableEntitys that belong to the enclosing RunnableEntityGroup. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime **InstanceRef implemented by:**RunnableEntityInCompositionInstanceRef
runnableEntity Group	RunnableEntityGroup	*	iref	This represents the ability to define nested groups of RunnableEntitys. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime **InstanceRef implemented by:**InnerRunnableEntityGroupInCompositionInstanceRef

Table D.74: RunnableEntityGroup

Class	**SdgClass**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	An SdgClass specifies the name and structure of the SDG that may be used to store proprietary data in an AUTOSAR model. The SdgClass is similar to an UML stereotype.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgElementWithGid*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
attribute (ordered)	SdgAttribute	*	aggr	Defintion of the structure of the Sdg **Tags:**xml.sequenceOffset=30
caption	Boolean	0..1	attr	Specifies if a caption is required. Note: only Sdgs that have a caption can be referenced **Tags:**xml.sequenceOffset=20
extendsMeta Class	MetaClassName	0..1	attr	The AUTOSAR Meta-Class that may be extended by this SdgClass. **Tags:**xml.sequenceOffset=10
sdgConstraint	TraceableText	*	ref	Semantic constraints that restrict the structure of the special data group. **Tags:**xml.sequenceOffset=40

Table D.75: SdgClass

Class	**SdgDef**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	A SdgDef groups several SdgClasses which belong to the same extension. The concept of an SdgDef is similiar to an UML Profile. **Tags:**atp.recommendedPackage=SdgDefs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**SdgDef**			
sdgClass	SdgClass	*	aggr	The owned sdgClasses which define the structure of the Sdgs **Tags:**xml.namePlural=SDG-CLASSES

Table D.76: SdgDef

Primitive	**SectionInitializationPolicyType**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	SectionInitializationPolicyType describes the intended initialization of MemorySections. The following values are standardized in AUTOSAR Methodology: • **NO-INIT**: No initialization and no clearing is performed. Such data elements shall not be read before one has written a value into it. • **INIT**: To be used for data that are initialized by every reset to the specified value (initValue). • **POWER-ON-INIT**: To be used for data that are initialized by "Power On" to the specified value (initValue). Note: there might be several resets between power on resets. • **CLEARED**: To be used for data that are initialized by every reset to zero. • **POWER-ON-CLEARED**: To be used for data that are initialized by "Power On" to zero. Note: there might be several resets between power on resets. Please note that the values are defined similar to the representation of enumeration types in the XML schema to ensure backward compatibility. **Tags:** xml.xsd.customType=SECTION-INITIALIZATION-POLICY-TYPE xml.xsd.type=NMTOKEN

Table D.77: SectionInitializationPolicyType

Class	**SenderReceiverInterface**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A sender/receiver interface declares a number of data elements to be sent and received. **Tags:**atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DataInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	VariableDataPrototype	*	aggr	The data elements of this SenderReceiverInterface.
invalidation Policy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement
metaDataItem Set	MetaDataItemSet	*	aggr	This aggregation defines fixed sets of meta-data items associated with dataElements of the enclosing Sender ReceiverInterface

Table D.78: SenderReceiverInterface

Class	**SwAddrMethod**
Package	M2::MSR::DataDictionary::AuxillaryObjects
Note	Used to assign a common addressing method, e.g. common memory section, to data or code objects. These objects could actually live in different modules or components. **Tags:**atp.recommendedPackage=SwAddrMethods

▽

△

Class	**SwAddrMethod**			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
memoryAllocationKeywordPolicy	MemoryAllocationKeywordPolicyType	0..1	attr	Enumeration to specify the name pattern of the Memory Allocation Keyword.
option	Identifier	*	attr	This attribute introduces the ability to specify further intended properties of the MemorySection in with the related objects shall be placed. These properties are handled as to be selected. The intended options are mentioned in the list. In the Memory Mapping configuration, this option list is used to determine an appropriate MemMapAddressingModeSet.
sectionInitializationPolicy	SectionInitializationPolicyType	0..1	attr	Specifies the expected initialization of the variables (inclusive those which are implementing VariableDataPrototypes). Therefore this is an implementation constraint for initialization code of BSW modules (especially RTE) as well as the start-up code which initializes the memory segment to which the AutosarDataPrototypes referring to the SwAddrMethod's are later on mapped. If the attribute is not defined it has the identical semantic as the attribute value "INIT"
sectionType	MemorySectionType	0..1	attr	Defines the type of memory sections which can be associated with this addresssing method.

Table D.79: SwAddrMethod

Class	**SwBaseType**			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This meta-class represents a base type used within ECU software. **Tags:**atp.recommendedPackage=BaseTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *BaseType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table D.80: SwBaseType

Class	**SwComponentPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	Role of a software component within a composition.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
type	SwComponentType	0..1	tref	Type of the instance. **Stereotypes:** isOfType

Table D.81: SwComponentPrototype

Class	***SwComponentType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for AUTOSAR software components.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AtomicSwComponentType*, CompositionSwComponentType, ParameterSwComponentType			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
consistency Needs	ConsistencyNeeds	*	aggr	This represents the collection of ConsistencyNeeds owned by the enclosing SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=consistencyNeeds.shortName, consistency Needs.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
port	PortPrototype	*	aggr	The PortPrototypes through which this SwComponent Type can communicate. The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=port.shortName, port.variationPoint.short Label vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

swComponent Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the SwComponentType.
				
				Stereotypes: atpSplitable; atpVariation
				Tags:
				atp.Splitkey=swComponentDocumentation, sw ComponentDocumentation.variationPoint.shortLabel
				vh.latestBindingTime=preCompileTime
				xml.sequenceOffset=-10

unitGroup	UnitGroup	*	ref	This allows for the specification of which UnitGroups are relevant in the context of referencing SwComponentType.

Table D.82: SwComponentType

Class	**SwServiceArg**			
Package	M2::MSR::DataDictionary::ServiceProcessTask			
Note	Specifies the properties of a data object exchanged during the call of an SwService, e.g. an argument or a return value. The SwServiceArg can also be used in the argument list of a C-macro. For this purpose the category shall be set to "MACRO". A reference to implementationDataType can optional be added if the actual argument has an implementationDataType.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**SwServiceArg**			
direction	ArgumentDirection Enum	0..1	attr	Specifies the direction of the data transfer. The direction shall indicate the direction of the actual information that is being consumed by the caller and/or the callee, not the direction of formal arguments in C. The attribute is optional for backwards compatibility reasons. For example, if a pointer is used to pass a memory address for the expected result, the direction shall be "out". If a pointer is used to pass a memory address with content to be read by the callee, its direction shall be "in". **Tags:**xml.sequenceOffset=10
swArraysize	ValueList	0..1	aggr	This turns the argument of the service to an array. **Tags:**xml.sequenceOffset=20
swDataDef Props	SwDataDefProps	0..1	aggr	Data properties of this SwServiceArg. **Tags:**xml.sequenceOffset=30

Table D.83: SwServiceArg

Class	**SwcBswMapping**			
Package	M2::AUTOSARTemplates::CommonStructure::SwcBswMapping			
Note	Maps an SwcInternalBehavior to an BswInternalBehavior. This is required to coordinate the API generation and the scheduling for AUTOSAR Service Components, ECU Abstraction Components and Complex Driver Components by the RTE and the BSW scheduling mechanisms. **Tags:**atp.recommendedPackage=SwcBswMappings			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
bswBehavior	BswInternalBehavior	0..1	ref	The mapped BswInternalBehavior
runnable Mapping	SwcBswRunnable Mapping	*	aggr	A mapping between a pair of SWC and BSW runnables. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
swcBehavior	SwcInternalBehavior	0..1	ref	The mapped SwcInternalBehavior.
synchronized ModeGroup	SwcBswSynchronized ModeGroupPrototype	*	aggr	A pair of SWC and BSW mode group prototypes to be synchronized by the scheduler. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
synchronized Trigger	SwcBswSynchronized Trigger	*	aggr	A pair of SWC and BSW Triggers to be synchronized by the scheduler. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table D.84: SwcBswMapping

Class	**SwcInternalBehavior**
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior
Note	The SwcInternalBehavior of an AtomicSwComponentType describes the relevant aspects of the software-component with respect to the RTE, i.e. the RunnableEntities and the RTEEvents they respond to.

▽

△

Class	SwcInternalBehavior			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *InternalBehavior*, *Multilanguage Referrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
arTypedPer Instance Memory	VariableDataPrototype	*	aggr	Defines an AUTOSAR typed memory-block that needs to be available for each instance of the SW-component. This is typically only useful if supportsMultipleInstantiation is set to "true" or if the component defines NVRAM access via permanent blocks. The aggregation of arTypedPerInstanceMemory is subject to variability with the purpose to support variability in the software component's implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arTypedPerInstanceMemory.shortName, ar TypedPerInstanceMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
event	RTEEvent	*	aggr	This is a RTEEvent specified for the particular Swc InternalBehavior. The aggregation of RTEEvent is subject to variability with the purpose to support the conditional existence of RTE events. Note: the number of RTE events might vary due to the conditional existence of PortPrototypes using Data ReceivedEvents or due to different scheduling needs of algorithms. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=event.shortName, event.variationPoint.short Label vh.latestBindingTime=preCompileTime
exclusiveArea Policy	SwcExclusiveArea Policy	*	aggr	Options how to generate the ExclusiveArea related APIs. When no SwcExclusiveAreaPolicy is specified for an ExclusiveArea the default values apply. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveAreaPolicy, exclusiveArea Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
explicitInter Runnable Variable	VariableDataPrototype	*	aggr	Implement state message semantics for establishing communication among runnables of the same component. The aggregation of explicitInterRunnable Variable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=explicitInterRunnableVariable.shortName, explicitInterRunnableVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
handle TerminationAnd Restart	HandleTerminationAnd RestartEnum	0..1	attr	This attribute controls the behavior with respect to stopping and restarting. The corresponding AtomicSw ComponentType may either not support stop and restart, or support only stop, or support both stop and restart.

▽

△

Class	**SwcInternalBehavior**			
implicitInter Runnable Variable	VariableDataPrototype	*	aggr	Implement state message semantics for establishing communication among runnables of the same component. The aggregation of implicitInterRunnable Variable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=implicitInterRunnableVariable.shortName, implicitInterRunnableVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
includedData TypeSet	IncludedDataTypeSet	*	aggr	The includedDataTypeSet is used by a software component for its implementation. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=includedDataTypeSet
includedMode Declaration GroupSet	IncludedMode DeclarationGroupSet	*	aggr	This aggregation represents the included Mode DeclarationGroups **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=includedModeDeclarationGroupSet
instantiation DataDefProps	InstantiationDataDef Props	*	aggr	The purpose of this is that within the context of a given SwComponentType some data def properties of individual instantiations can be modified. The aggregation of InstantiationDataDefProps is subject to variability with the purpose to support the conditional existence of Port Prototypes and component local memories like "per InstanceParameter" or "arTypedPerInstanceMemory". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=instantiationDataDefProps, instantiationData DefProps.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
perInstance Memory	PerInstanceMemory	*	aggr	Defines a per-instance memory object needed by this software component. The aggregation of PerInstance Memory is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=perInstanceMemory.shortName, perInstance Memory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
perInstance Parameter	ParameterData Prototype	*	aggr	Defines parameter(s) or characteristic value(s) that needs to be available for each instance of the software-component. This is typically only useful if supportsMultipleInstantiation is set to "true". The aggregation of perInstanceParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. ▽

▽

△

Class	**SwcInternalBehavior**			
				△ **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=perInstanceParameter.shortName, per InstanceParameter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
portAPIOption	PortAPIOption	*	aggr	Options for generating the signature of port-related calls from a runnable to the RTE and vice versa. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=portAPIOption, portAPIOption.variation Point.shortLabel vh.latestBindingTime=preCompileTime
runnable	RunnableEntity	*	aggr	This is a RunnableEntity specified for the particular Swc InternalBehavior. The aggregation of RunnableEntity is subject to variability with the purpose to support the conditional existence of RunnableEntities. Note: the number of RunnableEntities might vary due to the conditional existence of Port Prototypes using DataReceivedEvents or due to different scheduling needs of algorithms. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=runnable.shortName, runnable.variation Point.shortLabel vh.latestBindingTime=preCompileTime
service Dependency	SwcService Dependency	*	aggr	Defines the requirements on AUTOSAR Services for a particular item. The aggregation of SwcServiceDependency is subject to variability with the purpose to support the conditional existence of ports as well as the conditional existence of ServiceNeeds. The SwcServiceDependency owned by an SwcInternal Behavior can be located in a different physical file in order to support that SwcServiceDependency might be provided in later development steps or even by different expert domain (e.g OBD expert for Obd related Service Needs) tools. Therefore the aggregation is <<atp Splitable>>. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceDependency.shortName, service Dependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
shared Parameter	ParameterData Prototype	*	aggr	Defines parameter(s) or characteristic value(s) shared between SwComponentPrototypes of the same Sw ComponentType The aggregation of sharedParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. ▽

▽

△

Class	SwcInternalBehavior			
				△ **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=sharedParameter.shortName, shared Parameter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
supports Multiple Instantiation	Boolean	0..1	attr	Indicate whether the corresponding software-component can be multiply instantiated on one ECU. In this case the attribute will result in an appropriate component API on programming language level (with or without instance handle).
variationPoint Proxy	VariationPointProxy	*	aggr	Proxy of a variation points in the C/C++ implementation. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=variationPointProxy.shortName

Table D.85: SwcInternalBehavior

Class	***TDEventVfbPort*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	This is the abstract parent class to describe specific timing event types at Virtual Functional Bus (VFB) level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TimingDescription*, *Timing DescriptionEvent*			
Subclasses	TDEventModeDeclaration, TDEventOperation, TDEventTrigger, TDEventVariableDataPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
isExternal	Boolean	1	attr	This attribute is used to refer to external events that are related to hardware I/O, like physical sensors and actuators, at Virtual Functional Bus (VFB) level.
port	PortPrototype	0..1	ref	The port scope of the timing event.
portPrototype Blueprint	PortPrototypeBlueprint	0..1	ref	The PortPrototypeBlueprint is the scope of the timing event.

Table D.86: TDEventVfbPort

Class	**UnresolvedReferenceRestrictionWithSeverity**			
Package	M2::AUTOSARTemplates::CommonStructure::StandardizationTemplate::DataExchangePoint::Data FormatTailoring			
Note	This restriction defines the severity level of unresolved references.			
Base	*ARObject*, *RestrictionWithSeverity*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.87: UnresolvedReferenceRestrictionWithSeverity

Class	**VariableDataPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	A VariableDataPrototype is used to contain values in an ECU application. This means that most likely a VariableDataPrototype allocates "static" memory on the ECU. In some cases optimization strategies might lead to a situation where the memory allocation can be avoided. In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

Table D.88: VariableDataPrototype

Class	**VfbTiming**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing			
Note	A model element used to define timing descriptions and constraints at VFB level. TimingDescriptions aggregated by VfbTiming are restricted to event chains referring to events which are derived from the class TDEventVfb. **Tags:**atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*, *TimingExtension*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
component	SwComponentType	1	ref	This defines the scope of a VfbTiming. All corresponding timing descriptions and constraints shall be defined within this scope.

Table D.89: VfbTiming

E Variation Points in this Template

Variation Point	***Latest Binding Time***
BlueprintPolicyList.maxNumberOfElements	blueprintDerivationTime
BlueprintPolicyList.minNumberOfElements	blueprintDerivationTime
ClientServerInterfaceToBswModuleEntryBlueprintMapping.operationMapping	preCompileTime
ClientServerInterfaceToBswModuleEntryBlueprintMapping.portDefinedArgument Blueprint	preCompileTime
ConsistencyNeedsBlueprintSet.consistencyNeeds	preCompileTime
SwDataDefProps	codeGenerationTime
SwDataDefProps.swValueBlockSize	preCompileTime
SwDataDefProps.swValueBlockSizeMult	preCompileTime
SwTextProps.swMaxTextSize	preCompileTime
ValueList.vf	preCompileTime

Table E.1: Usage of variation points