

<b>Document Title</b>	Standardization Template
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	535
<b>Document Classification</b>	Standard

<b>Document Status</b>	Final
<b>Part of AUTOSAR Release</b>	4.2.2

<b>Document Change History</b>		
<b>Release</b>	<b>Changed by</b>	<b>Description</b>
4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• introduction of LifeCycleState for constraint and specification items</li> <li>• editorial changes</li> </ul>
4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• introduction of Blueprint Policy</li> <li>• include safety extension relevant items</li> <li>• extension of acceptance test items</li> </ul>
4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• editorial changes including tagged specification items</li> <li>• update content of specification levels</li> </ul>
4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• editorial changes including tagged specification items</li> <li>• extension of blueprinting to further AUTOSAR classes</li> </ul>
4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• editorial changes including tagged specification items</li> <li>• extension of blueprinting to further AUTOSAR classes (e.g. build action manifest)</li> <li>• introduction of life cycle support</li> <li>• improvement of document traceability</li> <li>• refinement of traceability support</li> </ul>
4.0.3	AUTOSAR Administration	Initial Release



## **Disclaimer**

This specification and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## **Advice for users**

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

## Table of Contents

1	Introduction	7
1.1	Document Conventions	7
1.2	Requirements Tracing	10
2	Support for Traceability	13
3	Life Cycle of AUTOSAR Definitions	24
4	The Principles of Blueprints	28
4.1	Abstract pattern for Blueprints	29
4.2	Mapping of Blueprints to blueprinted Elements	39
4.3	General Rules for Compliance of blueprint and blueprinted element	40
4.4	Applicable patterns to define names when deriving objects from blueprints	48
4.5	Applicable patterns to define blueprints expressions when deriving objects from blueprints	52
4.6	Ecu Configuration Parameters and Blueprints	54
5	Blueprintables defined in AUTOSAR Meta Model	54
5.1	Blueprinting AccessControl	54
5.2	Blueprinting AliasNameSet	54
5.3	Blueprinting ApplicationDataType	54
5.4	Blueprinting ARPackage	54
5.5	Blueprinting BswModuleDescription	55
5.6	Blueprinting BswModuleEntry	56
5.7	Blueprinting BuildActionManifest	56
5.8	Blueprinting CompuMethod	56
5.9	Blueprinting ConsistencyNeeds	57
5.10	Blueprinting DataConstr	60
5.11	Blueprinting DataTypeMappingSet	60
5.12	Blueprinting EcucDefinitionCollection	60
5.13	Blueprinting EcucModuleDef	60
5.14	Blueprinting FlatMap	60
5.15	Blueprinting ImplementationDataType	61
5.16	Blueprinting KeywordSet	61
5.17	Blueprinting LifeCycleStateDefinitionGroups and LifeCycleStates	61
5.18	Blueprinting ModeDeclarationGroup	61
5.19	Blueprinting PortPrototype	61
5.20	Blueprinting PortInterface	67
5.21	Blueprinting PortInterfaceMapping and PortInterfaceMappingSet	67
5.22	Blueprinting SwBaseType	74
5.23	Blueprinting SwComponentType	74
5.24	Blueprinting SwAddrMethods	74
5.25	Blueprinting VfbTiming	74

5.25.1	Example	75
6	Keywords	78
7	Deriving from AUTOSAR-provided Blueprints	81
A	Glossary	83
B	Change History	86
B.1	Change History R4.0.3	86
B.1.1	Added Constraints	86
B.1.2	Added Specification Items	86
B.2	Change History R4.1.1	87
B.2.1	Added Constraints	87
B.2.2	Added Specification Items	87
B.3	Change History R4.1.2	87
B.3.1	Added Constraints	87
B.3.2	Added Specification Items	88
B.4	Change History R4.1.3	88
B.4.1	Added Constraints in 4.1.3	88
B.4.2	Changed Constraints in 4.1.3	88
B.4.3	Deleted Constraints in 4.1.3	88
B.4.4	Added Traceables in 4.1.3	88
B.4.5	Changed Traceables in 4.1.3	88
B.4.6	Deleted Traceables in 4.1.3	89
B.5	Change History R4.2.1	89
B.5.1	Added Constraints in 4.2.1	89
B.5.2	Changed Constraints in 4.2.1	89
B.5.3	Deleted Constraints in 4.2.1	89
B.5.4	Added Traceables in 4.2.1	89
B.5.5	Changed Traceables in 4.2.1	90
B.5.6	Deleted Traceables in 4.2.1	90
B.6	Change History R4.2.2	90
B.6.1	Added Constraints in 4.2.2	90
B.6.2	Changed Constraints in 4.2.2	90
B.6.3	Deleted Constraints in 4.2.2	90
B.6.4	Added Traceables in 4.2.2	90
B.6.5	Changed Traceables in 4.2.2	90
B.6.6	Deleted Traceables in 4.2.2	91
C	Mentioned Class Tables	92
D	Variation Points in this Template	141

## Bibliography

- [1] Software Component Template  
AUTOSAR\_TPS\_SoftwareComponentTemplate
- [2] Requirements on Standardization Template  
AUTOSAR\_RS\_StandardizationTemplate
- [3] Specifications of Safety Extensions  
AUTOSAR\_TPS\_SafetyExtensions
- [4] Predefined Names in AUTOSAR  
AUTOSAR\_TR\_PredefinedNames
- [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] Specification of Timing Extensions  
AUTOSAR\_TPS\_TimingExtensions
- [13] SW-C and System Modeling Guide  
AUTOSAR\_TR\_SWCModelingGuide
- [14] Specification of Platform Types  
AUTOSAR\_SWS\_PlatformTypes
- [15] 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` 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. `PortPrototypes`. 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:

<b>Class</b>	<b>AUTOSAR</b>			
<b>Package</b>	M2::AUTOSARTemplates::AutosarTopLevelStructure			
<b>Note</b>	Root element of an AUTOSAR description, also the root element in corresponding XML documents.  <b>Tags:</b> xml.globalElement=true			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
adminData	AdminData	0..1	aggr	This represents the administrative data of an Autosar file.  <b>Tags:</b> xml.sequenceOffset=10
arPackage	<a href="#">ARPackage</a>	*	aggr	This is the top level package in an AUTOSAR model.  <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
introduction	<a href="#">Documentation Block</a>	0..1	aggr	This represents an introduction on the Autosar file. It is intended for example to represent disclaimers and legal notes.  <b>Tags:</b> 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.

**Datatype:** The datatype 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 attributes is aggregated in the class (*aggr*), an UML attribute in the class (*attr*), or just referenced by it (*ref*). Instance references are also indicated (*iref*) 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.

## 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]
[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 <code>shortName</code> 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 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]

Requirement	Description	Satisfied by
[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]
[RS_STDT_00018]	Shall allow to describe the dependencies of APIs (e.g. invocation and callback/polling interfaces)	[TPS_STDT_00014] [TPS_STDT_00048]
[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 VariableDataprototype	[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 PortBlueprint	[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]
[RS_STDT_00030]	Shall allow to standardize package structures	[TPS_STDT_00013] [TPS_STDT_00067]

Requirement	Description	Satisfied by
[RS_STDT_00031]	Shall support general specification items	[TPS_STDT_00042] [TPS_STDT_00056] [TPS_STDT_00057] [TPS_STDT_00058]
[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]
[RS_STDT_00038]	StandardizationTemplate shall specify the representation of constraint items in AUTOSAR documents	[TPS_STDT_00081]
[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]

**Table 1.2: RequirementsTracing**

## 2 Support for Traceability

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

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

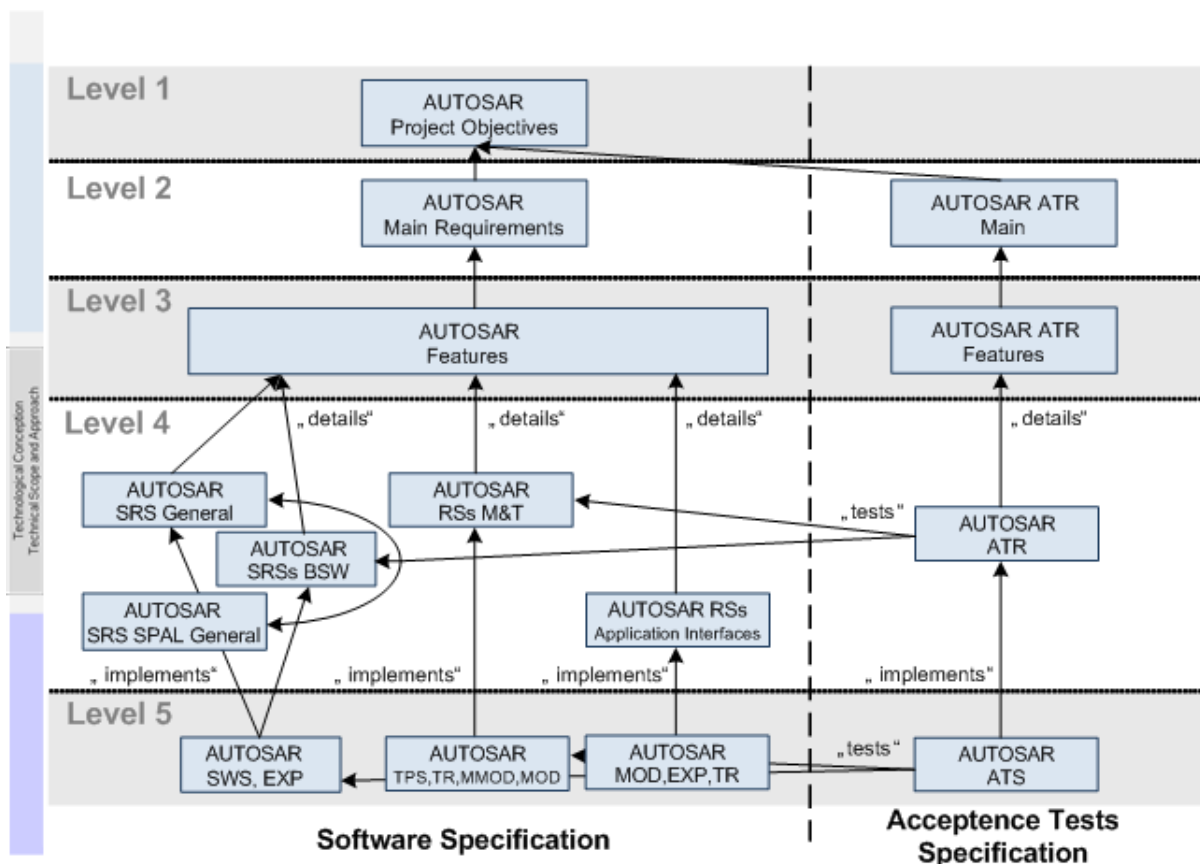


Figure 2.1: Specification levels

**[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](#))

**[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.

](RS\_STDT\_00037)

#### [TPS\_STDT\_00081] Representation of constraint items in AUTOSAR documents

[ AUTOSAR constraint items 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.<sup>1</sup>.
- 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) must 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

<sup>1</sup>Please refer to [https://svn.autosar.org/repos/work/24\\_Sources/branches/R4.0/ZAUX\\_Styles/10\\_ConstraintNumbers/constraint\\_numbers.csv](https://svn.autosar.org/repos/work/24_Sources/branches/R4.0/ZAUX_Styles/10_ConstraintNumbers/constraint_numbers.csv)

- Type, Description, Rationale, 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 2.2.

[SWS\_FOO\_07711] Formal Requirements shall look like this [

<b>Type:</b>	valid
<b>Description:</b>	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).
<b>Rationale:</b>	Why is this requirement important, what its omission could cause? We deliberately should harmonize the presentation of the AUTOSAR requirements.
<b>Use Case:</b>	A scenario that makes the requirement necessary or useful. [UC_FOO_00001], [UC_FOO_00001]
<b>Dependencies:</b>	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]
<b>Supporting Material:</b>	References to other documents, models etc.

](SRS\_FOO\_00815, SRS\_BAR\_00007)

**Figure 2.2: Requirements Table**

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 must 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) must 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 5 as in level 4, see figure 2.1.

The rendition is illustrated in figure 2.3.

[ATR\_FOO\_04711] Formal Tests shall look like this [

<b>Type:</b>	valid
<b>Description:</b>	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.
<b>Rationale:</b>	Why is this acceptance test important, what its omission could cause? We deliberately should harmonize the presentation of the AUTOSAR acceptance tests.
<b>Use Case:</b>	A scenario that makes the acceptance test necessary or useful. [UC_FOO_00001],[UC_FOO_00002]
<b>Dependencies:</b>	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]
<b>Supporting Material:</b>	References to other documents, models etc.
<b>Tested Items:</b>	[SWS_FOO_00815],[TPS_STDT_00042]

](RS\_BRF\_00123)

**Figure 2.3: 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

**SPECIFICATION\_ITEM** The text represents a particular item in the specification. Such an item is a requirement for the implementation of the software specification.

**REQUIREMENT\_ITEM** The text represents a particular requirement. Such an item is applicable primarily in requirement specifications.



**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.

**TEST\_ITEM** The text represents a short description of a test. Such an item is applicable primarily in test specifications.

**SAFETY\_\*** The text represents the type of safety requirements. The allowed values (\*) are defined in [TPS\_SAFEX\_00102] in [3].

⌋()

- **[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 Template 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 [4] in keyword set `Information-Categories`, entries with classification `TraceCategory`.
- `module` is either module abbreviation in [5] or an entry of the keyword set `DocumentAbbreviations` with classification `DocumentAbbreviation` in [4].
- `index` is a numerical index
- `special` is one of (SPEC, NA, GEN). Note that `special` may also have an optional index. This allows to provide different special items with more detailed information.

Note that this pattern is not yet applied in all AUTOSAR Documents. ⌋([RS\\_STDT\\_00009](#), [RS\\_STDT\\_00008](#), [RS\\_STDT\\_00001](#), [RS\\_STDT\\_00031](#))

**[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 created 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 2.4 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 2.4: Example for trace table using NA and SPEC**

[TPS\_STDT\_00052] Characteristics of `TraceableText` [ `TraceableText` should<sup>2</sup> be:

- **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.

<sup>2</sup>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.

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 marketplace 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 [4] in keyword set `InformationCategories`, entries with classification `DocumentCategory`.
- `DocumentName` is the `shortName` of the `Keyword` according to [4], keyword set `DocumentAbbreviation` entries with classification `DocumentAbbreviation` or the `shortName` of the module in [5]

]([RS\\_STDT\\_00009](#))

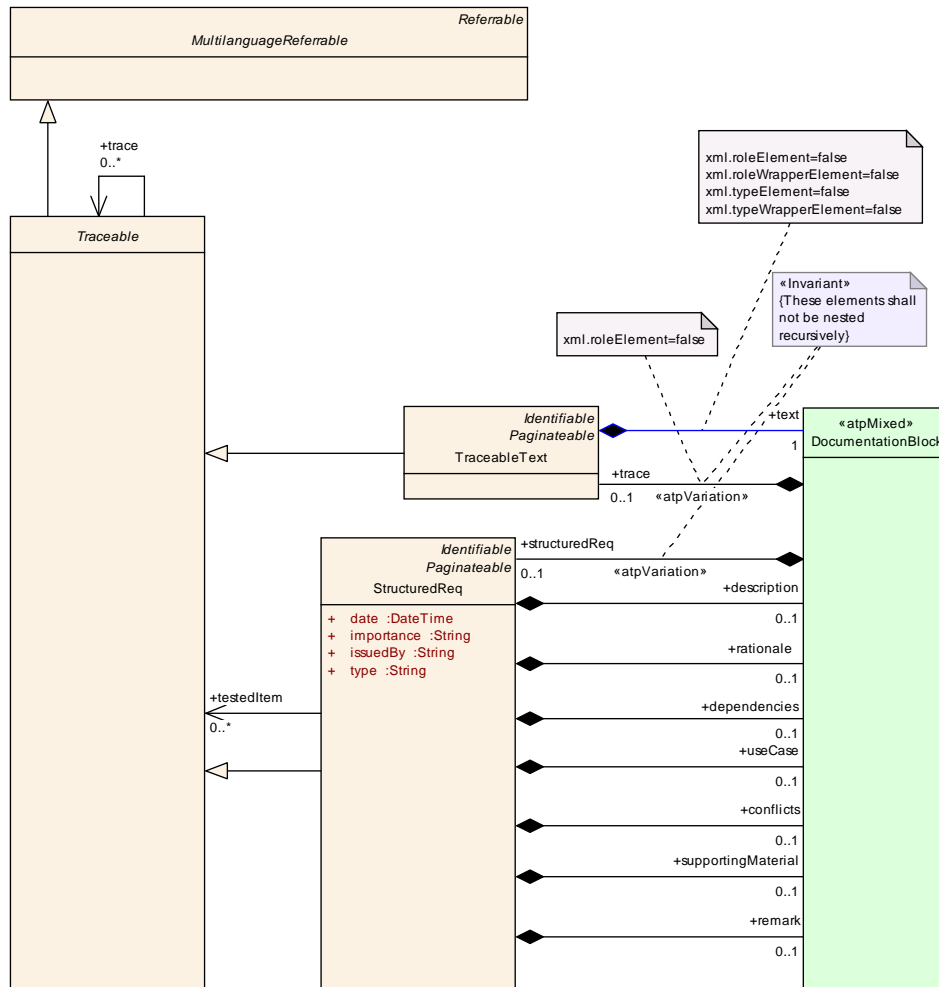


Figure 2.5: Requirements and Tracing

<b>Class</b>	<b>Traceable (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Documentation::BlockElements::RequirementsTracing			
<b>Note</b>	<p>This meta class represents the ability to be subject to tracing within an AUTOSAR model.</p> <p>Note that it is expected that its subclasses inherit either from <code>MultilanguageReferrable</code> or from <code>Identifiable</code>. Nevertheless it also inherits from <code>MultilanguageReferrable</code> in order to provide a common reference target for all <code>Traceables</code>.</p>			
<b>Base</b>	<code>ARObject</code> , <code>MultilanguageReferrable</code> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
trace	<a href="#">Traceable</a>	*	ref	<p>This association represents the ability to trace to upstream requirements / constraints. This supports for example the bottom up tracing</p> <p>ProjectObjectives &lt;- MainRequirements &lt;- Features &lt;- RequirementSpecs &lt;- BSW/AI</p> <p><b>Tags:</b> <code>xml.sequenceOffset=20</code></p>

Table 2.1: Traceable

<b>Class</b>	<b>TraceableText</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Documentation::BlockElements::RequirementsTracing			
<b>Note</b>	<p>This meta-class represents the ability to denote a traceable text item such as requirements etc.</p> <p>The following approach applies:</p> <ul style="list-style-type: none"> <li>• <b>shortName</b> represents the tag for tracing</li> <li>• <b>longName</b> represents the head line</li> <li>• <b>category</b> represents the kind of the tagged text</li> </ul>			
<b>Base</b>	ARObject, DocumentViewSelectable, <a href="#">Identifiable</a> , Multilanguage Referrable, Paginateable, <a href="#">Referrable</a> , <a href="#">Traceable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
text	<a href="#">Documentation Block</a>	1	aggr	<p>This represents the text to which the tag applies.</p> <p><b>Tags:</b> xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false</p>

**Table 2.2: TraceableText**

<b>Class</b>	<b>StructuredReq</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Documentation::BlockElements::RequirementsTracing			
<b>Note</b>	<p>This represents a structured requirement. This is intended for a case where specific requirements for features are collected.</p> <p>Note that this can be rendered as a labeled list.</p>			
<b>Base</b>	ARObject, DocumentViewSelectable, <a href="#">Identifiable</a> , Multilanguage Referrable, Paginateable, <a href="#">Referrable</a> , <a href="#">Traceable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
conflicts	<a href="#">Documentation Block</a>	0..1	aggr	<p>This represents an informal specification of conflicts.</p> <p><b>Tags:</b> xml.sequenceOffset=40</p>
date	DateTime	1	attr	<p>This represents the date when the requirement was initiated.</p> <p><b>Tags:</b> xml.sequenceOffset=5</p>
dependencies	<a href="#">Documentation Block</a>	0..1	aggr	<p>This represents an informal specification of dependencies. Note that upstream tracing should be formalized in the property trace provided by the superclass Traceable.</p> <p><b>Tags:</b> xml.sequenceOffset=30</p>
description	<a href="#">Documentation Block</a>	0..1	aggr	<p>This represents the general description of the requirement.</p> <p><b>Tags:</b> xml.sequenceOffset=10</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
importance	String	1	attr	This allows to represent the importance of the requirement.  <b>Tags:</b> xml.sequenceOffset=8
issuedBy	String	1	attr	This represents the person, organization or authority which issued the requirement.  <b>Tags:</b> xml.sequenceOffset=6
rationale	<a href="#">Documentation Block</a>	0..1	aggr	This represents the rationale of the requirement.  <b>Tags:</b> xml.sequenceOffset=20
remark	<a href="#">Documentation Block</a>	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.  <b>Tags:</b> xml.sequenceOffset=60
supporting Material	<a href="#">Documentation Block</a>	0..1	aggr	This represents an informal specification of the supporting material.  <b>Tags:</b> xml.sequenceOffset=50
testedItem	<a href="#">Traceable</a>	*	ref	This association represents the ability to trace on the same specification level. This supports for example the of acceptance tests.  <b>Tags:</b> 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.  <b>Tags:</b> xml.sequenceOffset=7
useCase	<a href="#">Documentation Block</a>	0..1	aggr	This describes the relevant use cases. Note that formal references to use cases should be done in the trace relation.  <b>Tags:</b> xml.sequenceOffset=35

**Table 2.3: 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`

```
<ADMIN-DATA>
  <LANGUAGE>EN</LANGUAGE>
```

```

<USED-LANGUAGES>
  <L-10 L="EN" xml:space="default">English</L-10>
</USED-LANGUAGES>
</ADMIN-DATA>
<AR-PACKAGES>
<!-- AR-Package: AUTOSAR -->
  <AR-PACKAGE>
    <SHORT-NAME>AUTOSAR</SHORT-NAME>
    <AR-PACKAGES>
      <AR-PACKAGE>
        <!-- AR-Package: GenDef -->
          <SHORT-NAME>GenDef</SHORT-NAME>
          <AR-PACKAGES>
            <AR-PACKAGE>
              <!-- AR-Package: LifeCycleStateDefinitionGroups -->
                <SHORT-NAME>LifeCycleStateDefinitionGroups</SHORT-NAME>
                <CATEGORY>STANDARD</CATEGORY>
                <ELEMENTS>
                  <!-- 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>
                            <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>
                            <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>
                            <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>
    <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>
    <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>
    <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>
</ELEMENTS>
</AR-PACKAGE>
</AR-PACKAGES>
</AR-PACKAGE>
</AR-PACKAGES>
</AR-PACKAGE>
</AR-PACKAGES>
</AUTOSAR>
```

## 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.

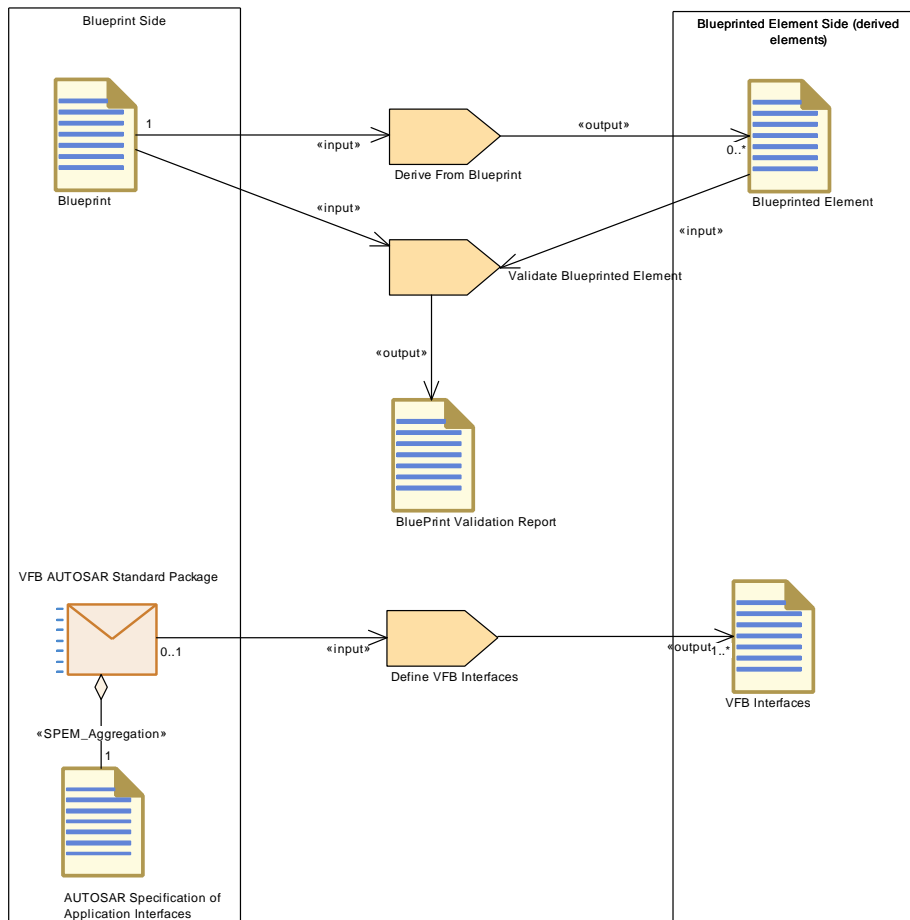


Figure 4.1: Blueprint methodology approach

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.

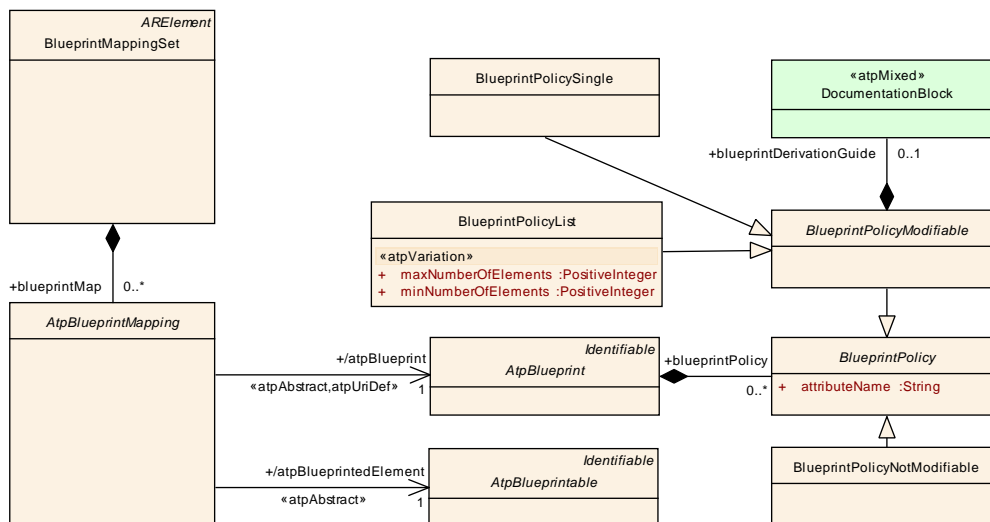
But there might be additional elements to support the fact that it is a blueprint. An example for this is that `PortPrototypeBlueprint` also specifies `init-Values` which is not the case for `PortPrototype` which get their initial values from appropriate `ComSpecs`.

- **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.

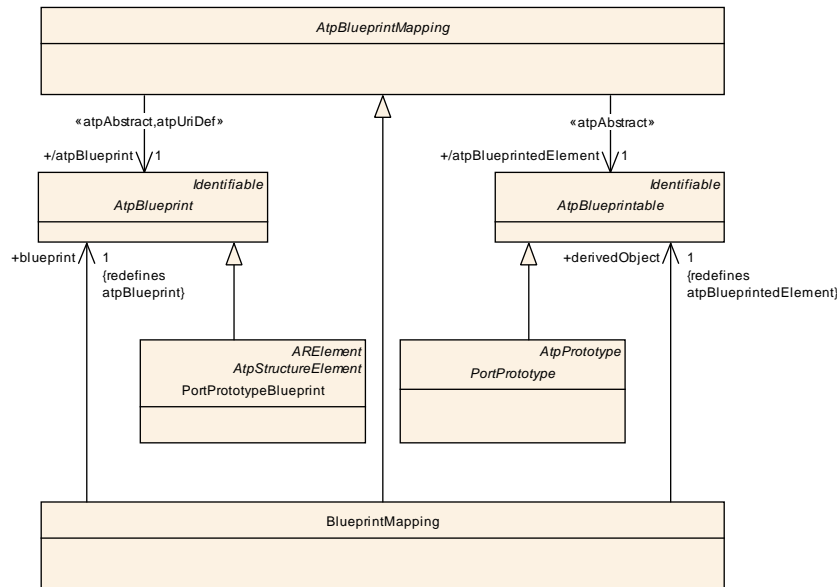
Note that the subsequent processing of blueprinted elements (e.g. RTE generation) do not refer to the blueprints anymore.

- **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.



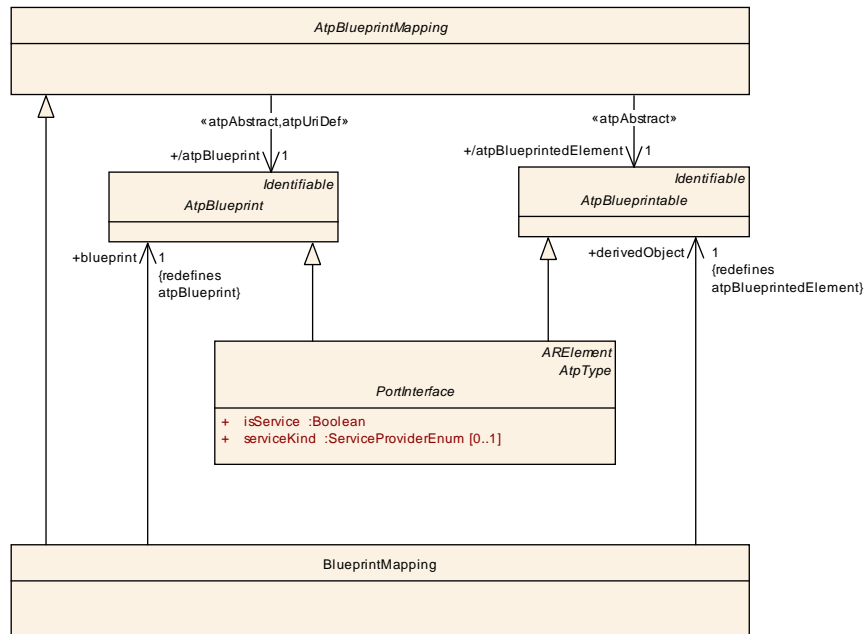
**Figure 4.3: Port Blueprints as an example for separate meta-classes for Blueprint and blueprinted Element**

**[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`.

**[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 it 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">0x00 is locked</L-1>
        </P>
        <P>
          <L-1 L="EN">0x01...0x3F is configuration dependent</L-1>
        </P>
        <P>
          <L-1 L="EN">0x40...0xFF is Reserved by Document</L-1>
        </P>
      </BLUEPRINT-DERIVATION-GUIDE>
      <MAX-NUMBER-OF-ELEMENTS>3</MAX-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>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>
<SHORT-NAME-PATTERN>{anyName}</SHORT-NAME-PATTERN>
<INTERFACE-REF DEST="SENDER-RECEIVER-INTERFACE" BASE="PortInterfaces">
  /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>
    
```

<b>Class</b>	<b>AtpBlueprint (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
blueprintPolicy	<a href="#">BlueprintPolicy</a>	*	aggr	This role indicates whether the blueprintable element will be modifiable or not modifiable.

**Table 4.2: AtpBlueprint**

<b>Class</b>	<b>AtpBlueprintable (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 4.3: AtpBlueprintable**

<b>Class</b>	<b>AtpBlueprintMapping (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
atpBlueprint	<a href="#">AtpBlueprint</a>	1	ref	This represents the blueprint.  <b>Stereotypes:</b> atpAbstract; atpUriDef <b>Tags:</b> xml.sequenceOffset=50
atpBlueprintedElement	<a href="#">AtpBlueprintable</a>	1	ref	This represents the blueprinted elements which shall be mapped to the blueprint.  <b>Stereotypes:</b> atpAbstract <b>Tags:</b> xml.sequenceOffset=60

**Table 4.4: AtpBlueprintMapping**

<b>Class</b>	<b>BlueprintPolicy (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
<b>Note</b>	This meta-class represents the ability to indicate whether blueprintable elements will be modifiable or not modifiable.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>

<i>Attribute</i>	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>
attributeName	String	1	attr	

**Table 4.5: BlueprintPolicy**

<i>Class</i>	<b>BlueprintPolicyList</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
<b>Note</b>	The class represents that the related attribute is modifiable during the blueprinting. It applies only to attribute with upper multiplicity greater than 1.			
<b>Base</b>	ARObject, <a href="#">BlueprintPolicy</a> , <a href="#">BlueprintPolicyModifiable</a>			
<i>Attribute</i>	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>
maxNumberOfElements	PositiveInteger	1	attr	Maximum number of elements in list. If the maximum number is not constraint it shall be set to "undefined".  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=blueprintDerivationTime
minNumberOfElements	PositiveInteger	1	attr	Minimum number of elements in the list. If the minimum number is not constraint it shall be set to "undefined".  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=blueprintDerivationTime

**Table 4.6: BlueprintPolicyList**

<i>Class</i>	<b>BlueprintPolicyModifiable (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
<b>Note</b>	The class represents that the related attribute is modifiable during the blueprinting.			
<b>Base</b>	ARObject, <a href="#">BlueprintPolicy</a>			
<i>Attribute</i>	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>
blueprintDerivationGuide	<a href="#">Documentation Block</a>	0..1	aggr	This role offers the possibility to give additional information to the policy.

**Table 4.7: BlueprintPolicyModifiable**

<i>Class</i>	<b>BlueprintPolicyNotModifiable</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
<b>Note</b>	The class represents that the related attribute is not modifiable during the blueprinting.			
<b>Base</b>	ARObject, <a href="#">BlueprintPolicy</a>			
<i>Attribute</i>	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>
–	–	–	–	–

**Table 4.8: BlueprintPolicyNotModifiable**

<b>Class</b>	<b>BlueprintPolicySingle</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
<b>Note</b>	The class represents that the related attribute is modifiable during the blueprinting. It applies only to attribute with upper multiplicity equal 1.			
<b>Base</b>	ARObject, <a href="#">BlueprintPolicy</a> , <a href="#">BlueprintPolicyModifiable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table 4.9: BlueprintPolicySingle**

## 4.2 Mapping of Blueprints to blueprinted Elements

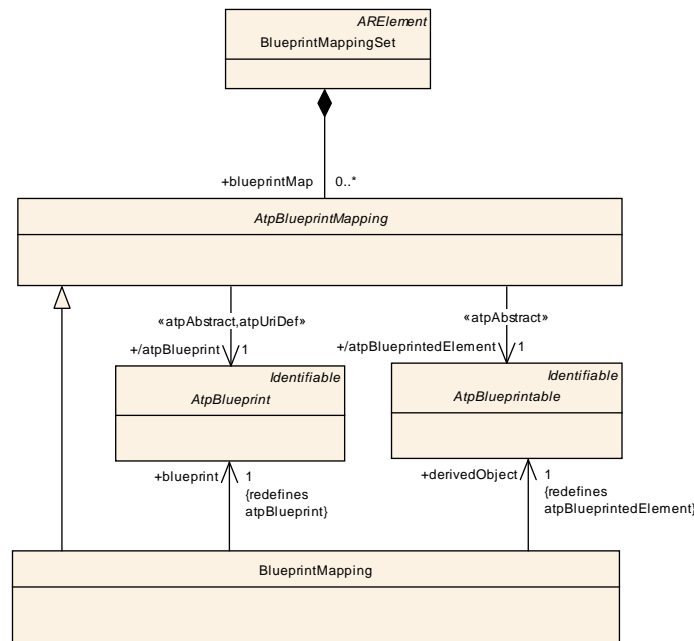
In many cases it will be necessary to identify the relationship of a blueprinted element (e.g. [PortPrototype](#)) to the corresponding blueprint (e.g. [PortPrototypeBlueprint](#)) after the blueprinted element has been created according to the blueprint.

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](#))<sup>1</sup>.

<sup>1</sup>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. ]()

<b>Class</b>	<b>BlueprintMappingSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintMapping			
<b>Note</b>	This represents a container of mappings between "actual" model elements and the "blueprint" that has been taken for their creation.  <b>Tags:</b> atp.recommendedPackage=BlueprintMappingSets			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">Multilanguage</a> <a href="#">Referrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
blueprintMap	<a href="#">AtpBlueprintMapping</a>	*	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 [[constr\\_2555](#)] 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` may **not** be copied. ]()

**[constr\_2555] 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

]()

**[constr\_2542] Compatibility of `longName`, `desc` and `introduction` of blueprint and blueprinted element** [ Elements derived from blueprints are allowed to

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

]()

Note that [\[constr\\_2542\]](#) 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. This is done in `blueprintCondition` resp. `blueprintValue`. See [\[TPS\\_STDT\\_00048\]](#) for details.

**[constr\_2543] Specify a name pattern in blueprints** [ For each blueprint, a `namePattern` 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 ([\[constr\\_2553\]](#)). ]()

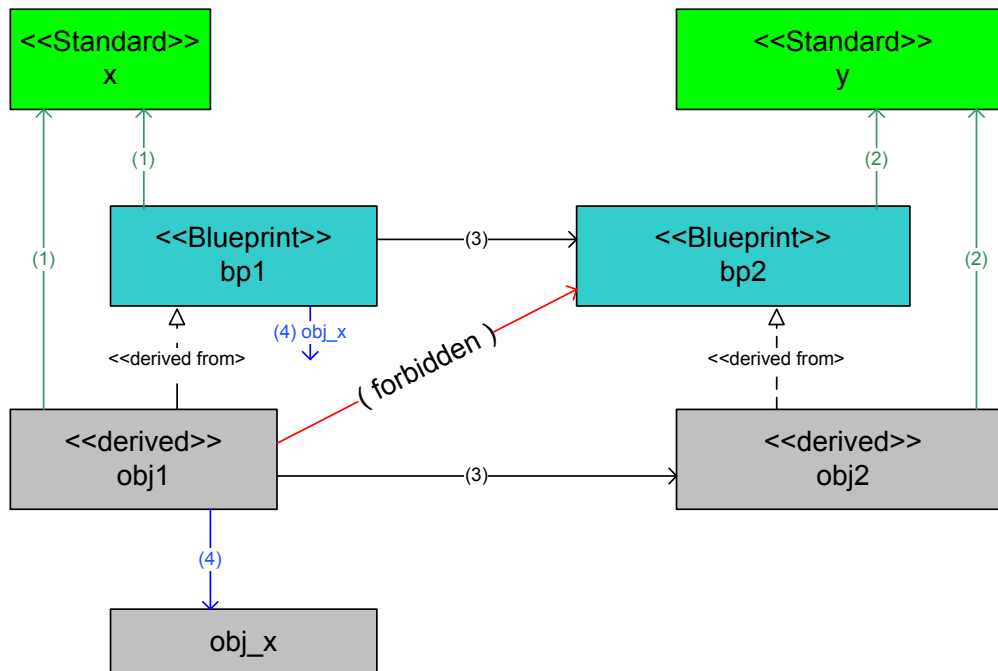
**[constr\_2553] `shortName` shall follow the pattern defined in the Blueprint** [ The `shortName` respectively `symbol` of the derived objects shall follow the pattern defined in `namePattern` of the blueprint according to [\[constr\\_2543\]](#) ]()

**[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 `AtpBlueprintMappings`. 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 from Blueprint to Blueprint need to be replaced in derived objects** [ 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 [5.7](#) 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`.

The attributes `Identifier.namePattern` resp. `CIdentifier.namePattern` 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` 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` 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/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`.

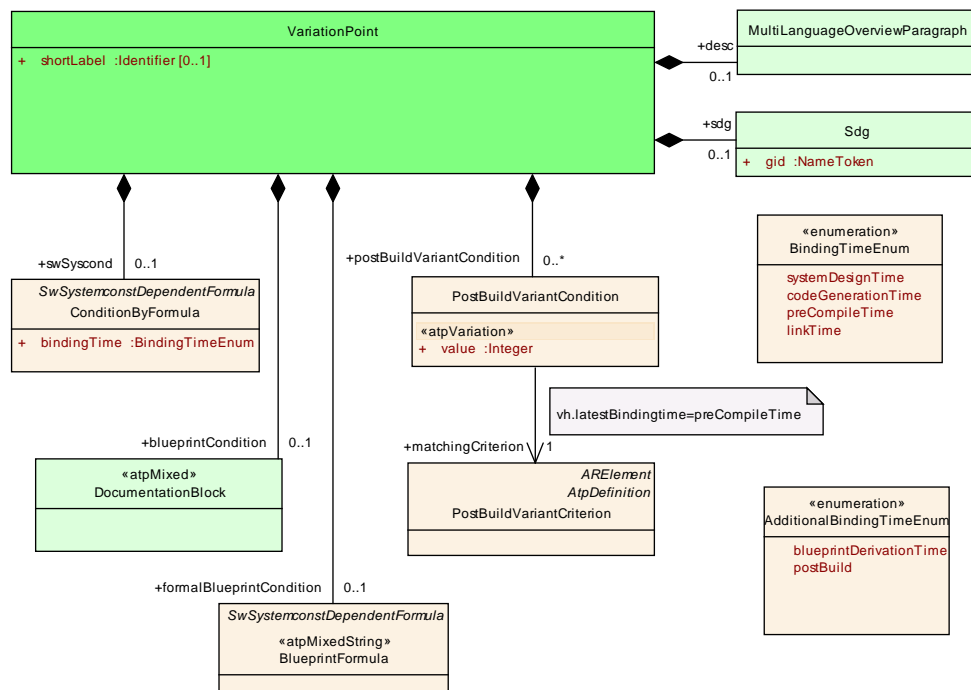


Figure 4.7: Variation Point

<b>Class</b>	<b>VariationPoint</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
<b>Note</b>	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 postBuildVariantCriterion is fulfilled.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
desc	MultiLanguage OverviewParagr aph	0..1	aggr	This allows to describe shortly the purpose of the variation point.  <b>Tags:</b> xml.sequenceOffset=20
blueprintC ondition	<a href="#">Documentation Block</a>	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.  <b>Tags:</b> xml.sequenceOffset=28
formalBlue printCondit ion	<a href="#">BlueprintFormul a</a>	0..1	aggr	This denotes a formal blueprintCondition. This shall be not in contradiction with blueprintCondition. It is recommended only to use one of the two.  <b>Tags:</b> xml.sequenceOffset=29
postBuildV ariantCond ition	PostBuildVarian tCondition	*	aggr	This is the set of post build variant conditions which all shall be fulfilled in order to (postbuild) bind the variation point.  <b>Tags:</b> 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.  <b>Tags:</b> xml.sequenceOffset=50
shortLabel	<a href="#">Identifier</a>	0..1	ref	This provides a name to the particular variation point to support the RTE generator. It is necessary for supporting splittable 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.  <b>Tags:</b> xml.sequenceOffset=10
swSyscon d	ConditionByFor mula	0..1	aggr	This condition acts as Binding Function for the VariationPoint. Note that the multiplicity is 0..1 in order to support pure postBuild variants.  <b>Tags:</b> 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 the information provided in [VariationPoint.blueprintCondition](#):

- is resolved manually when deriving objects.
- is resolved by a module generator. The resolver approach is not formalized but hard coded in the module generator. Note that in this case it is also likely that multiple objects are created by the module generator. This shall also be noted in the [blueprintCondition](#).
- 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 [[constr\\_2555](#)]). Thereby the `shortLabel` of the [VariationPoint](#) may be adapted according to the description in the [blueprintCondition](#). ]([RS\\_STDT\\_00020](#))

**[constr\_2556] No Blueprint Motivated [VariationPoints](#) in AUTOSAR Descriptions** [ AUTOSAR descriptions which are not blueprints shall not have [blueprintCondition](#) nor [blueprintValue](#). ]()

**[constr\_2569] Purely Blueprint Motivated [VariationPoints](#)** [ [VariationPoints](#) with `vh.latestBindingTime` set to `blueprintDerivationTime` shall have only [blueprintCondition](#) respectively [blueprintValue](#). ]()

**[TPS\_STDT\_00045] Transferring Objects in General** [ Objects resp. references without [VariationPoint](#) shall be transferred to the derived objects. Thereby the `namePatterns` of the referenced Blueprints also apply for rewriting the `shortName` path in the reference. ]([RS\\_STDT\\_00020](#))

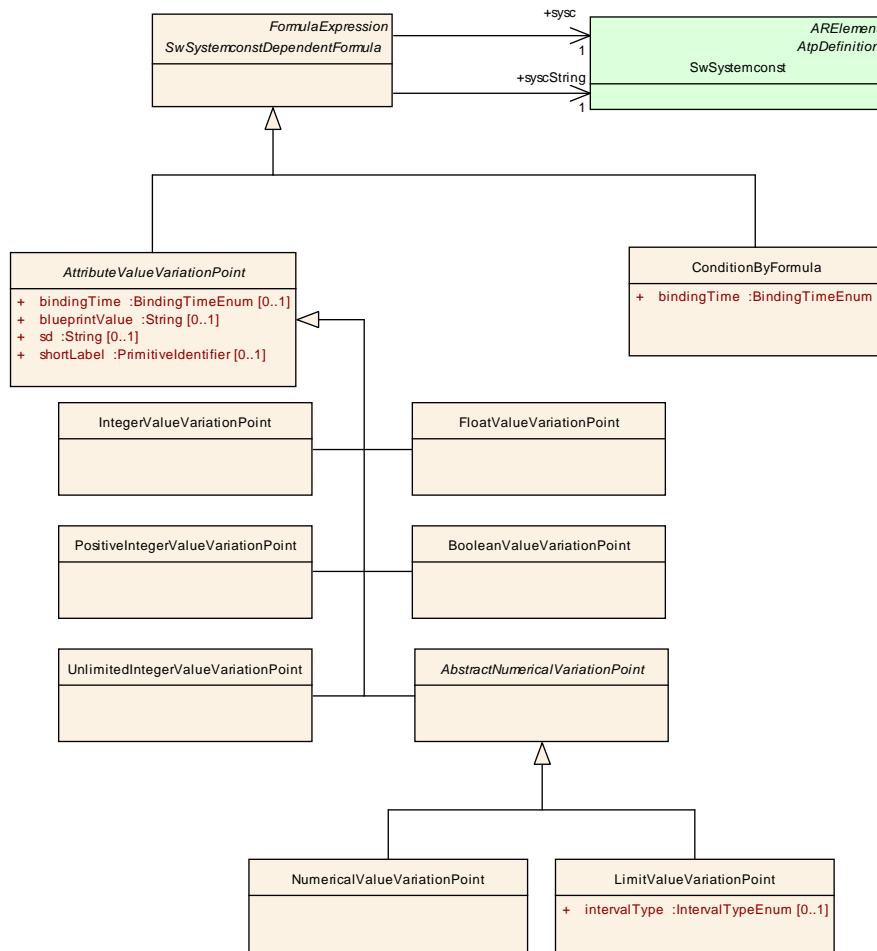
For more details about [VariationPoint](#) refer to [7], as all constraints are summarized there.

**[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](#). This attribute correlates to [blueprintCondition](#) in [VariationPoint](#). ]([RS\\_STDT\\_00020](#))

An example for [[TPS\\_STDT\\_00046](#)] is:

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



**Figure 4.8: Attribute Value Variation Point**

<b>Class</b>	«atpMixedString» <b>AttributeValueVariationPoint (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling::AttributeValueVariationPoints			
<b>Note</b>	This class represents the ability to derive the value of the Attribute from a system constant (by SwSystemconstDependentFormula). It also provides a bindingTime.			
<b>Base</b>	ARObject,FormulaExpression,SwSystemconstDependentFormula			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
bindingTime	BindingTimeEnum	0..1	attr	<p>This is the binding time in which the attribute value needs to be bound.</p> <p>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.</p> <p><b>Tags:</b> xml.attribute=true</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint.  <b>Tags:</b> 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.  <b>Tags:</b> 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.  <b>Tags:</b> xml.attribute=true

**Table 4.12: AttributeValueVariationPoint**

## 4.4 Applicable patterns to define names when deriving objects from blueprints

[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](#))

<b>Primitive</b>	<b>Identifier</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
<b>Note</b>	<p>An Identifier is a string with a number of constraints on its appearance, satisfying the requirements typical programming languages define for their Identifiers.</p> <p>This datatype represents a string, that can be used as a c-Identifier.</p> <p>It shall start with a letter, may consist of letters, digits and underscores.</p> <p><b>Tags:</b> xml.xsd.customType=IDENTIFIER; xml.xsd.maxLength=128; xml.xsd.pattern=[a-zA-Z][a-zA-Z0-9_]*; xml.xsd.type=string</p>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>



<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
namePattern	String	0..1	attr	<p>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.</p> <p>For more details refer to TPS_StandardizationTemplate.</p> <p><b>Tags:</b> xml.attribute=true</p>

**Table 4.13: Identifier**

<b>Primitive</b>	<b>CIdentifier</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
<b>Note</b>	<p>This datatype represents a string, that follows the rules of C-identifiers.</p> <p><b>Tags:</b> xml.xsd.customType=C-IDENTIFIER; xml.xsd.pattern=[a-zA-Z][a-zA-Z0-9_]*; xml.xsd.type=string</p>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
namePattern	String	0..1	attr	<p>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.</p> <p>For more details refer to TPS_StandardizationTemplate.</p> <p><b>Tags:</b> xml.attribute=true</p>

**Table 4.14: CIdentifier**

**[TPS\_STDT\_00055] General Syntax for Name Patterns** [ The name pattern uses the following syntax defined according to ANTLR [9].

**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 : '/' ;

```

|(RS\_STDT\_00004)

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. [\]\(RS\\_STDT\\_00021\)](#)

**typed** This represents an indicator based on the type of the object.

## 4.5 Applicable patterns to define blueprints expressions when deriving objects from blueprints

**[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` resp. `blueprintCondition`. [\]\(RS\\_STDT\\_00019\)](#)

**[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\_00021] Specialization of `BlueprintFormula`** [ These specialization(s) express the extension of the Formula Language to provide formalized `blueprintValue` resp. `blueprintCondition`:

- `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'

[\]\(RS\\_STDT\\_00019\)](#)

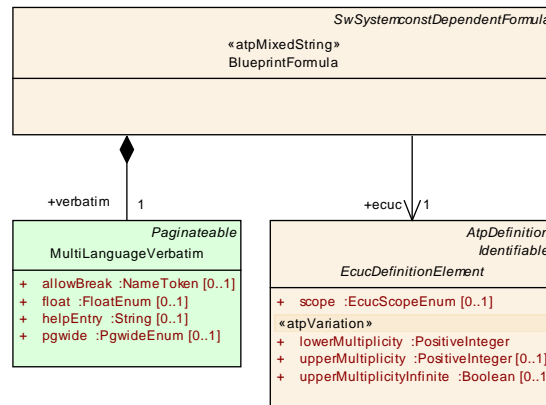


Figure 4.9: Blueprint Formula

Listing 4.9 illustrates valid expression patterns. Note that `blueprintValue`, `blueprintCondition` etc. denotes a placeholder.

```
{blueprintCondition}:
blueprintCondition = <ECUC-REF DEST="ECUC-ENUMERATION-PARAM-DEF">
    NvM/NvMCommon/NvMApiConfigClass
</ECUC-REF>
```

Listing 4.9: Blueprint Formula taken from AUTOSAR\_MOD\_BSWServiceInterfaces\_Blueprint.arxml

```
<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")
  & amp; & amp;
  <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

```
<VARIATION-POINT>
  <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>
</VARIATION-POINT>
```

## 4.6 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 `EcucModuleDef.refinedModuleDef`.

Nevertheless for company specific applications there is some support for ECU configuration in Standardization Template. ]([RS\\_STDT\\_00022](#), [RS\\_STDT\\_00010](#))

See chapter [5.12](#) resp. chapter [5.13](#) 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)

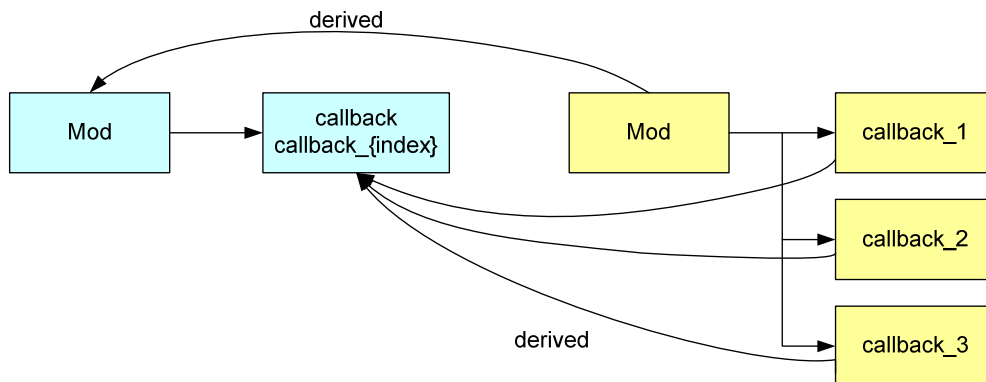
Blueprints for BswModuleDescription are used in particular to describe dependencies to other modules. Note that in this case all references to other modules and module entries are targeting blueprints of the intended module. These references need to be replaced when deriving objects from the blueprint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

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

- BswModuleDescription.providedEntry
- BswModuleDescription.outgoingCallback
- BswModuleDescription.bswModuleDependency.requiredEntry
- BswModuleDescription.bswModuleDependency.expectedCallback

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 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 BuildActionManifest

[TPS\_STDT\_00063] **Blueprinting `BuildActionManifest`** [ `BuildActionManifest` can be blueprinted. [TPS\_STDT\_00065] applies such that blueprints of `BuildAction` and `BuildActionEnvironments` are aggregated in a blueprint of `BuildActionManifest`. ](*RS\_STDT\_00033*)

## 5.8 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 `variation-Point` at `CompuScale`. ](*RS\_STDT\_00002, RS\_STDT\_00029*)



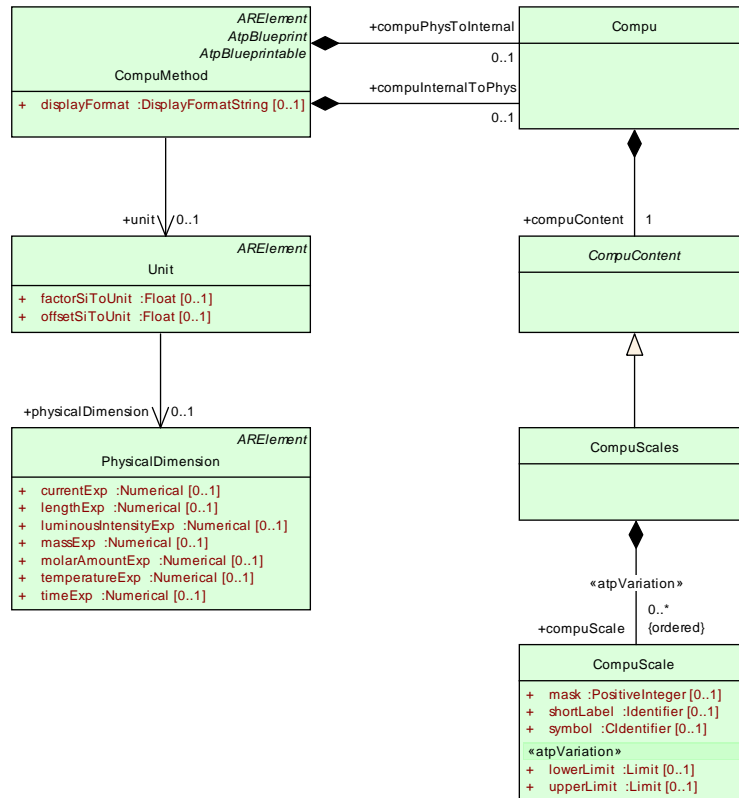


Figure 5.2: A **CompuMethod** and its attributes define data semantics

## 5.9 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*)

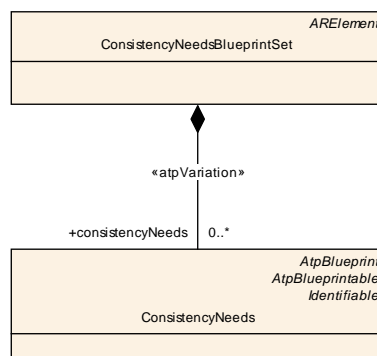
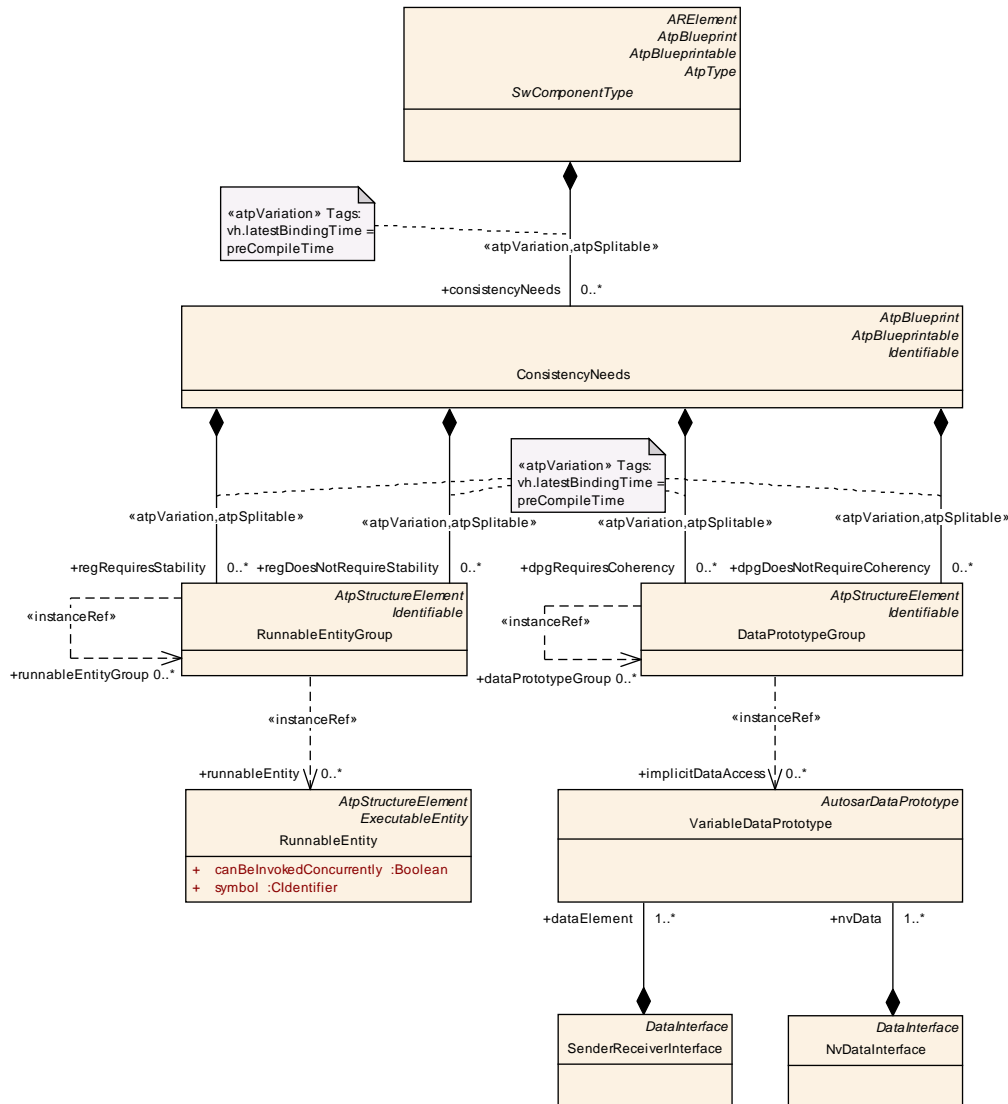


Figure 5.3: Blueprinting **ConsistencyNeeds**



**Figure 5.4: ConsistencyNeeds**

**[TPS\_STDT\_00073] Early definition of ConsistencyNeeds** [ Grouping of Data shall be possible before the `RunnableEntity` 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 `RunnableEntities` of the software component.

**ALL\_RUNNABLES\_WRITING\_TO\_DATA\_PROTOTYP\_GROUP** `RunnableEntityGroup` of the `ConsistencyNeeds` shall contain all `RunnableEntities` 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 `RunnableEntities` 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 RunnableEntities 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 RunnableEntities according functional requirements

](RS\_STDT\_00034)

## 5.10 Blueprinting DataConstr

[TPS\_STDT\_00016] Blueprinting **DataConstr** [ DataConstr can be blueprinted. ](RS\_STDT\_00029)

## 5.11 Blueprinting DataTypeMappingSet

[TPS\_STDT\_00017] Blueprinting **DataTypeMappingSet** [ DataTypeMappingSet can be blueprinted. ](RS\_STDT\_00029)

## 5.12 Blueprinting EcucDefinitionCollection

[TPS\_STDT\_00018] Blueprinting **EcucDefinitionCollection** [ EcucDefinitionCollection can be blueprinted. ](RS\_STDT\_00029)

## 5.13 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 4.6.

## 5.14 Blueprinting FlatMap

[TPS\_STDT\_00035] Blueprinting **FlatMap** [ FlatMap can be blueprinted. ](RS\_STDT\_00029)

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

## 5.15 Blueprinting ImplementationDataType

[TPS\_STDT\_00020] Blueprinting **ImplementationDataType** [ **ImplementationDataType** can be blueprinted. ](*RS\_STDT\_00029*)

## 5.16 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
- [*constr\_2542*] 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*)

## 5.17 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*)

## 5.18 Blueprinting ModeDeclarationGroup

[TPS\_STDT\_00031] Blueprinting **ModeDeclarationGroup** [ **ModeDeclarationGroup** can be blueprinted. ](*RS\_STDT\_00024*)

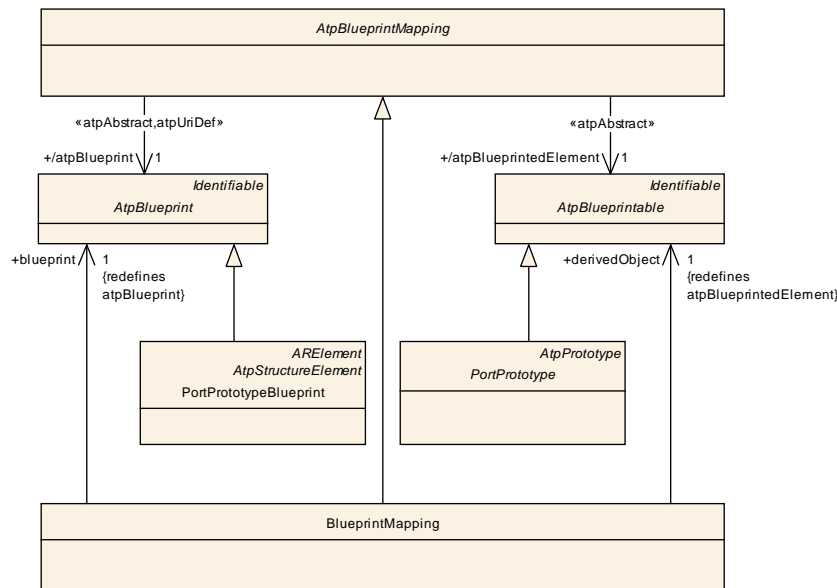
## 5.19 Blueprinting PortPrototype

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.

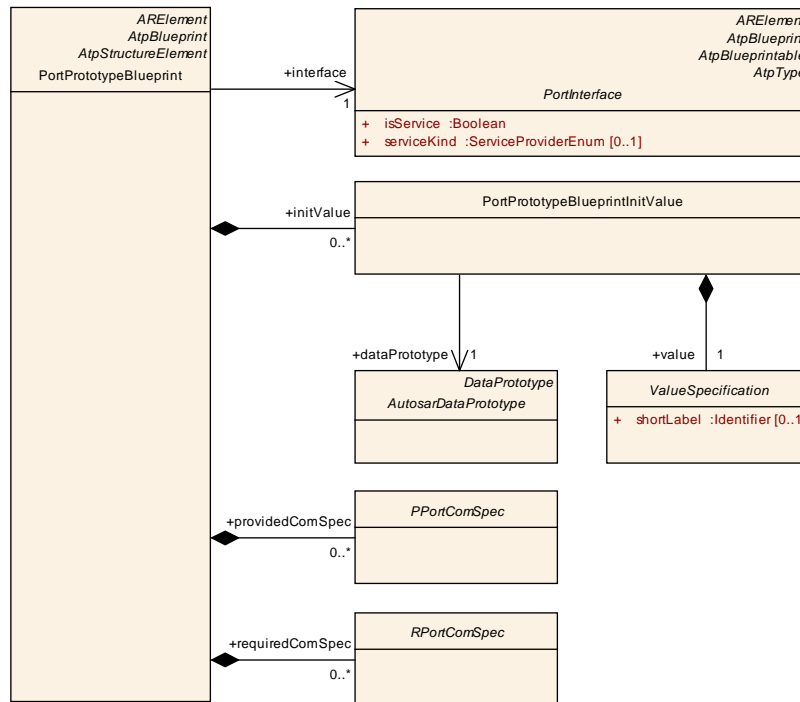
Due to the structure of the AUTOSAR meta-model it is not possible to merely express a standardized `PortPrototype` because for good reasons the latter does not exist on its own but is always owned by a `SwComponentType`.

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 5.5: Mapping of Port Prototype Blueprints**



**Figure 5.6: 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<sup>1</sup> 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-](#)

<sup>1</sup>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>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>
  <SHORT-NAME-PATTERN>{anyName}</SHORT-NAME-PATTERN>
  <INTERFACE-REF DEST="SENDER-RECEIVER-INTERFACE" BASE="PortInterfaces">/
  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/
            BaseTypeBlueprint/nineBitMsbFirst1c</BASE-TYPE-REF>
            <COMPU-METHOD-REF DEST="COMPU-METHOD">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>

```

<b>Class</b>	<b>PortPrototypeBlueprint</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintDedicated::PortPrototypeBlueprint			
<b>Note</b>	<p>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 PortInterfaces.</p> <p><b>Tags:</b> atp.recommendedPackage=PortPrototypeBlueprints</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
initValue	<a href="#">PortPrototypeBlueprintInitValue</a>	*	aggr	This specifies the init values for the dataElements in the particular PortPrototypeBlueprint.
interface	<a href="#">PortInterface</a>	1	ref	This is the interface for which the blueprint is defined. It may be a blueprint itself or a standardized PortInterface
providedComSpec	<a href="#">PPortComSpec</a>	*	aggr	Provided communication attributes per interface element (data element or operation).
requiredComSpec	<a href="#">RPortComSpec</a>	*	aggr	Required communication attributes, one for each interface element.

**Table 5.1: PortPrototypeBlueprint**

<b>Class</b>	<b>PortPrototypeBlueprintInitValue</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintDedicated::PortPrototypeBlueprint			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
dataPrototype	AutosarDataPrototype	1	ref	This is the data prototype for which the init value applies  <b>Tags:</b> xml.sequenceOffset=30
value	ValueSpecification	1	aggr	This is the init value for the particular data prototype.  <b>Tags:</b> xml.sequenceOffset=40

**Table 5.2: 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`.  $\square()$

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").

## 5.20 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.  $\square()$

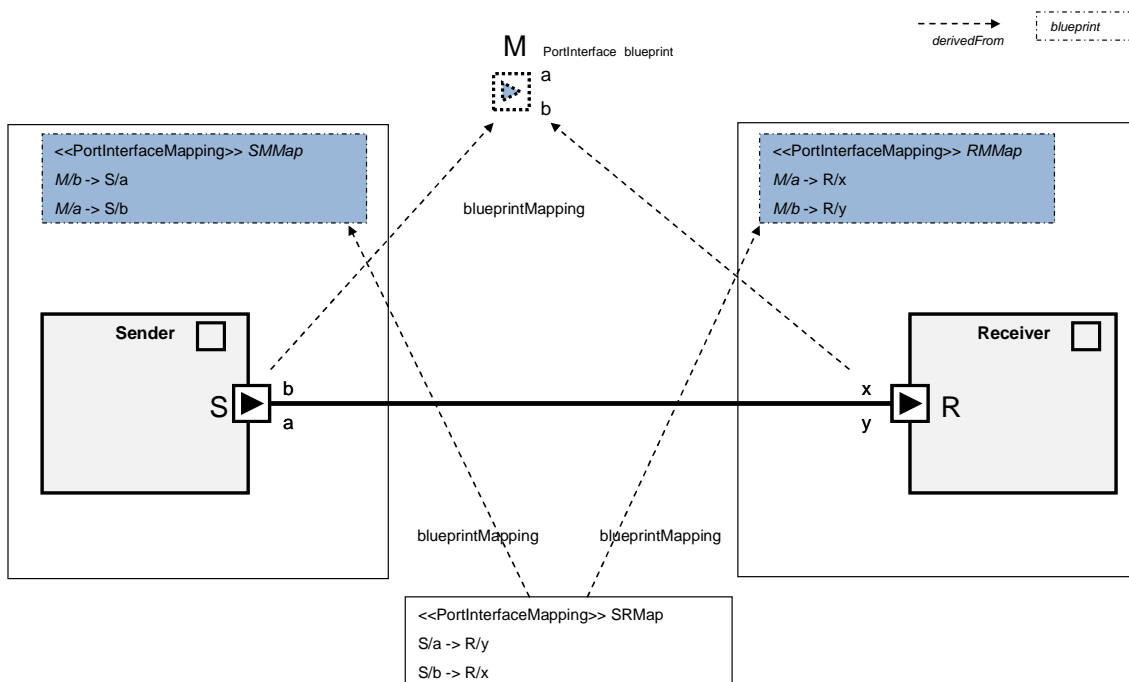
Note that [constr\_2500] is a special case of [constr\_2566].

## 5.21 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 5.7. 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) ( $SMMMap$  and  $RMMMap$ ). From this, it is possible to

1. derive `PortInterfaceMapping` ( $SRMap$ ) between ( $S$  and  $R$ ) which is then derived from two blueprints ( $SMMMap$  and  $RMMMap$ )
2. propose connectors between two components using the interfaces ( $S$  and  $R$ )



**Figure 5.7: 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 `Port-Interface`s 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 `PortInterface`s `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` (*SMap*) in step 3 to the involved `PortInterfaceMapping`(blueprints) (*SMap* and *RMap*).

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 *SMap*.

#### 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>
          </DATA-ELEMENTS>
        </SENDER-RECEIVER-INTERFACE>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>

```

```

        </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}">SMap</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-MAPPINGS>
                </VARIABLE-AND-PARAMETER-INTERFACE-MAPPING>
            </PORT-INTERFACE-MAPPINGS>
        </PORT-INTERFACE-MAPPING-SET>
    </ELEMENTS>
</AR-PACKAGE>

```

```

        <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 `RMMMap`.

#### 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}">RMap</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 "Left-Company". Thereby it maps *S* to *R* in `PortInterfaceMapping SRMap`. This is derived from two blueprints (`SMap` 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
          SMap and RMap simultaneously</L-2>
      </DESC>
      <BLUEPRINT-MAPS>
        <BLUEPRINT-MAPPING>
          <BLUEPRINT-REF DEST="PORT-INTERFACE-MAPPING">/LeftCompany
            /PortInterfaceMappingSets_Blueprint/BP/SMap</
            BLUEPRINT-REF>
          <DERIVED-OBJECT-REF DEST="PORT-INTERFACE-MAPPING">/
            Project/PortInterfaceMappingSets/Set1/SMap</DERIVED-
            OBJECT-REF>
        </BLUEPRINT-MAPPING>
        <BLUEPRINT-MAPPING>
          <BLUEPRINT-REF DEST="PORT-INTERFACE-MAPPING">/
            RightCompany/PortInterfaceMappingSets_Blueprint/BP/
            RMap</BLUEPRINT-REF>
      </BLUEPRINT-MAPS>
    </BLUEPRINT-MAPPING-SET>
  </ELEMENTS>
</AR-PACKAGE>

```

```

<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>

```

## 5.22 Blueprinting SwBaseType

[TPS\_STDT\_00022] Blueprinting **SwBaseType** [ **SwBaseType** can be blueprinted. ]([RS\\_STDT\\_00029](#))

## 5.23 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**].

## 5.24 Blueprinting SwAddrMethods

[TPS\_STDT\_00026] Blueprinting **SwAddrMethod** [ **SwAddrMethod** can be blueprinted. ]([RS\\_STDT\\_00029](#))

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

Figure 5.8 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.

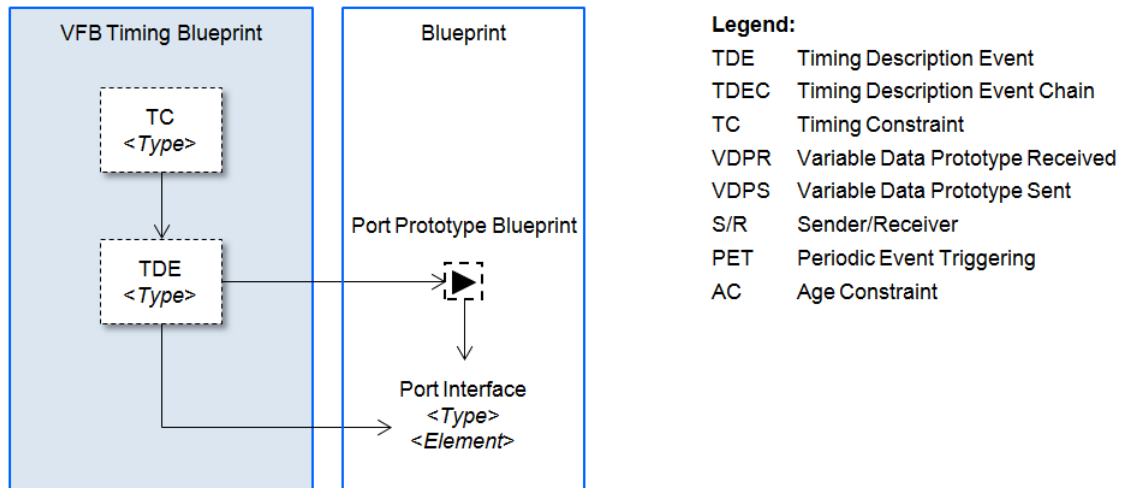


Figure 5.8: VFB Timing Blueprint

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” [12].

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. ]()

### 5.25.1 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” [?].

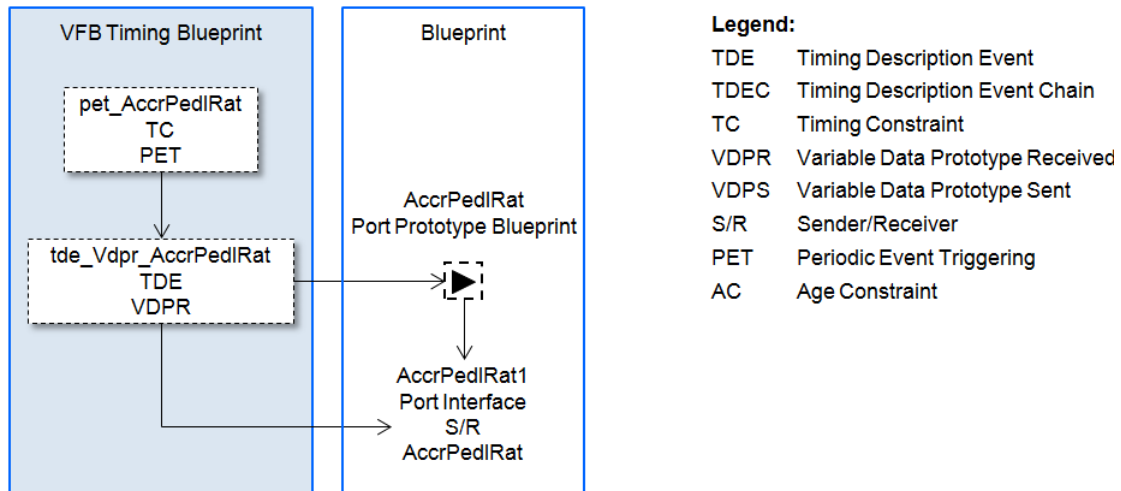


Figure 5.9: VFB Timing Blueprint Simple Example

As sketched in Figure 5.9 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 5.9, but contains further timing description events and an additional age timing constraint imposed on the reception of the specific variable data prototype.

Listing 5.6: Example for VFB Timing Blueprint

```

<AR-PACKAGES>
  <AR-PACKAGE S="" UUID="">
    <SHORT-NAME>VfbTimingBlueprint</SHORT-NAME>
    <CATEGORY>BLUEPRINT</CATEGORY>
    <SHORT-NAME-PATTERN>{anyName}</SHORT-NAME-PATTERN>
    <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>
        </TIMING-DESCRIPTIONS>
      </VFB-TIMING>
    </ELEMENTS>
  </AR-PACKAGE>
</AR-PACKAGES>
    
```

```

    <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 5.10 shows the VFB Timing Blueprint and the derived VFB Timing for a specific software component called “SW-C\_A”.

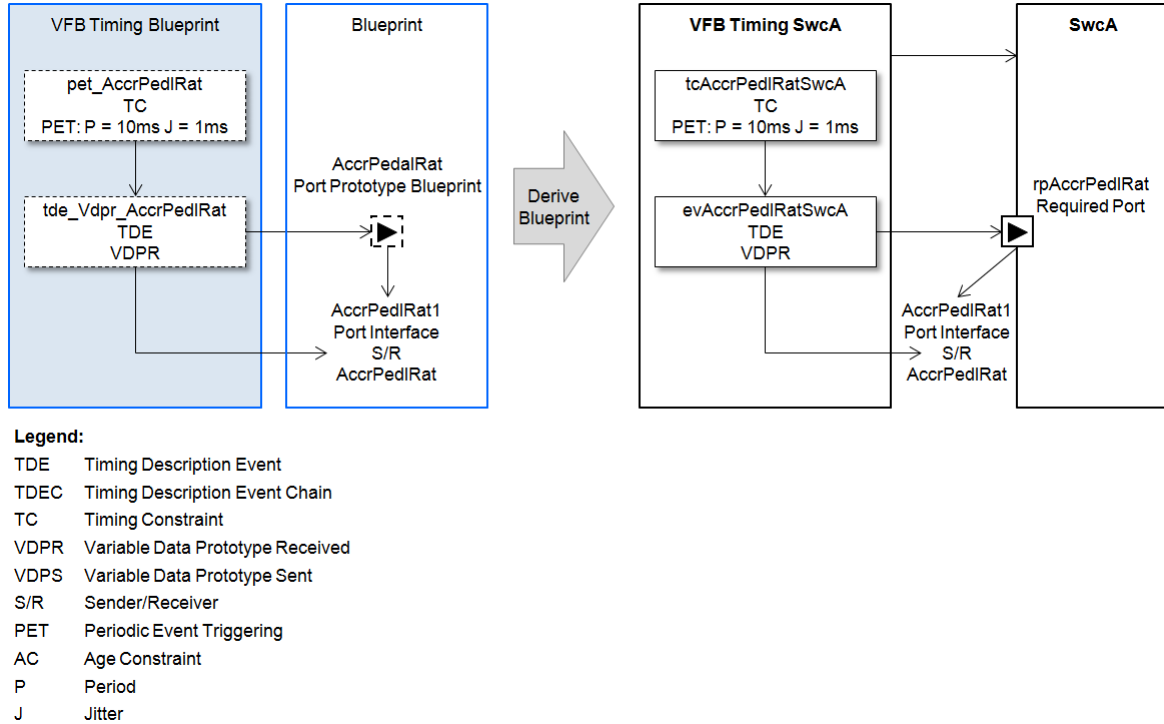


Figure 5.10: Deriving a VFB Timing Blueprint

## 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.

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

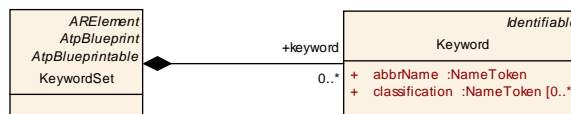


Figure 6.1: Keyword and KeywordSet

[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](#))

<b>Class</b>	<b>KeywordSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::Keyword			
<b>Note</b>	This meta–class represents the ability to collect a set of predefined keywords.  <b>Tags:</b> atp.recommendedPackage=KeywordSets			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
keyword	<a href="#">Keyword</a>	*	aggr	This is one particular keyword in the keyword set.

**Table 6.1: KeywordSet**

<b>Class</b>	<b>Keyword</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::Keyword			
<b>Note</b>	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 <code>shortName</code> . It could be <code>symbol</code> , or even <code>longName</code> . Application of keywords is not limited to particular names.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
abbrName	<code>NameToken</code>	1	attr	This attribute specifies an abbreviated name of a keyword. This abbreviation may e.g. be used for constructing valid <code>shortNames</code> according to the AUTOSAR naming conventions.  Unlike <code>shortName</code> , it may contain any name token. E.g. it may consist of digits only.

Attribute	Datatype	Mul.	Kind	Note
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 [13] 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 [4].

**Listing 6.1: example for keywords**

```
<?xml version="1.0" encoding="UTF-8"?>
<AUTOSAR xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://
  autosar.org/schema/r4.0" xsi:schemaLocation="http://autosar.org/schema
  /r4.0_AUTOSAR_4-1-3.xsd">
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>AUTOSAR</SHORT-NAME>
    </AR-PACKAGE>
    <AR-PACKAGES>
      <AR-PACKAGE>
        <SHORT-NAME>AISpecification</SHORT-NAME>
      </AR-PACKAGE>
      <AR-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>KeywordSets</SHORT-NAME>
          <ELEMENTS>
            <KEYWORD-SET>
              <SHORT-NAME>KeywordList</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-PACKAGES>
  </AUTOSAR>

```



```

    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>
</AR-PACKAGES>
</AR-PACKAGE>
</AR-PACKAGES>
</AUTOSAR>

```

[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 `Drvvr`, `Drvg` stem from `Drv`<sup>1</sup>. 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/Drvvr</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>

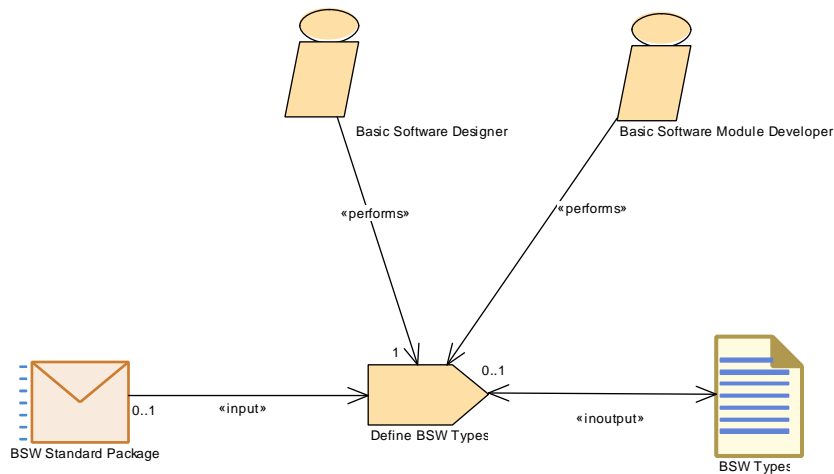
```

## 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 [?] but also for the Software Specifications of the BSW layer. These AUTOSAR delivered model elements follow the package structure specified in [TPS\_GST\_00080].

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.

<sup>1</sup>Note that `Drv` is not an [element](#) of this [Collection](#) since it is not a declination of itself.



**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 [14] blueprinted in

`/AUTOSAR/Platform/ImplementationDatatypes_Blueprint/uint8`

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

`/AUTOSAR_Platform/ImplementationDatatypes/uint8`

## A 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 ([15]).

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 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 `<<atpSplittable>>`. 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 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 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.

**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`.

## B Change History

### B.1 Change History R4.0.3

#### B.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 Blueprint Motivated VariationPoints
[constr_2570]	No Blueprints in system descriptions
[constr_2571]	Outgoing references from Blueprints

**Table B.1: Added Constraints in 4.0.3**

#### B.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 <code>TraceableText</code>
[TPS_STDT_00053]	Expression of obligation
[TPS_STDT_00054]	Organisation of <code>TraceableText</code>
[TPS_STDT_00055]	General Syntax for Name Patterns

**Table B.2: Added Specification Items in 4.0.3**

## B.2 Change History R4.1.1

### B.2.1 Added Constraints

Number	Heading

**Table B.3: Added Constraints in 4.1.1**

### B.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]	<code>TraceableText</code>
[TPS_STDT_00060]	<code>StructuredReq</code>
[TPS_STDT_00062]	Blueprinting Elements of <code>AccessControl</code>
[TPS_STDT_00063]	Blueprinting <code>BuildActionManifest</code>
[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 <code>PortInterface</code>
[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 <code>ConsistencyNeeds</code>
[TPS_STDT_00072]	Same Meta Class For Blueprints and Derived Objects
[TPS_STDT_00073]	Early definition of <code>ConsistencyNeeds</code>
[TPS_STDT_00074]	Categorization of Blueprints of <code>ConsistencyNeeds</code>
[TPS_STDT_00075]	Categories for <code>DataPrototypeGroup</code> in a Blueprint of <code>ConsistencyNeeds</code>
[TPS_STDT_00076]	Categories for <code>RunnableEntityGroup</code> in a Blueprint of <code>ConsistencyNeeds</code>
[TPS_STDT_00077]	Blueprinting <code>KeywordSet</code>
[TPS_STDT_00078]	Representation of requirements in AUTOSAR documents

**Table B.4: Added Specification Items in 4.1.1**

## B.3 Change History R4.1.2

### B.3.1 Added Constraints

Number	Heading

--	--

**Table B.5: Added Constraints in 4.1.2**

### B.3.2 Added Specification Items

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

**Table B.6: Added Specification Items in 4.1.2**

## B.4 Change History R4.1.3

### B.4.1 Added Constraints in 4.1.3

Number	Heading
[constr_2589]	In VFB Timing Blueprint <code>TDEventVfbPort</code> shall reference <code>PortPrototypeBlueprint</code>

**Table B.7: Added Constraints in 4.1.3**

### B.4.2 Changed Constraints in 4.1.3

none

### B.4.3 Deleted Constraints in 4.1.3

none

### B.4.4 Added Traceables in 4.1.3

Id	Heading
[TPS_STDT_00026]	Blueprinting <code>SwAddrMethod</code>

**Table B.8: Added Traceables in 4.1.3**

### B.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 B.9: Changed Traceables in 4.1.3**

#### B.4.6 Deleted Traceables in 4.1.3

none

### B.5 Change History R4.2.1

#### B.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 B.10: Added Constraints in 4.2.1**

#### B.5.2 Changed Constraints in 4.2.1

Id	Heading
[constr_2540]	Tagged text category

**Table B.11: Changed Constraints in 4.2.1**

#### B.5.3 Deleted Constraints in 4.2.1

none

#### B.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 <a href="#">BlueprintPolicy</a>
[TPS_STDT_00061]	<a href="#">PortPrototypeBlueprint</a> can own both <a href="#">RPortComSpecs</a> and <a href="#">PPortComSpecs</a>
[TPS_STDT_00082]	Multiple existence of <code>initValue</code> in the context of a <a href="#">PortPrototypeBlueprint</a>

**Table B.12: Added Traceables in 4.2.1**

### B.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 <a href="#">BlueprintFormula</a>
[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 B.13: Changed Traceables in 4.2.1**

### B.5.6 Deleted Traceables in 4.2.1

none

## B.6 Change History R4.2.2

### B.6.1 Added Constraints in 4.2.2

none

### B.6.2 Changed Constraints in 4.2.2

Id	Heading
[constr_2592]	No BlueprintPolicy

**Table B.14: Changed Constraints in 4.2.2**

### B.6.3 Deleted Constraints in 4.2.2

none

### B.6.4 Added Traceables in 4.2.2

none

### B.6.5 Changed Traceables in 4.2.2

Id	Heading
----	---------

[TPS_STDT_00039]	Xpath Expressions for <a href="#">BlueprintPolicy</a>
[TPS_STDT_00077]	Blueprinting <a href="#">KeywordSet</a>
[TPS_STDT_00080]	Representation of specification items in AUTOSAR documents
[TPS_STDT_00081]	Representation of constraint items in AUTOSAR documents

**Table B.15: Changed Traceables in 4.2.2****B.6.6 Deleted Traceables in 4.2.2**

none

## C 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.

<b>Class</b>	<b>ARElement (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
<b>Note</b>	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).			
<b>Base</b>	ARObject,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table C.1: ARElement**

<b>Class</b>	<b>ARPackage</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
<b>Note</b>	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.			
<b>Base</b>	ARObject,AtpBlueprint,AtpBlueprintable,CollectableElement,Identifiable,MultilanguageReferrable,Referrable			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
arPackage	ARPackage	*	aggr	This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableElement	*	aggr	Elements that are part of this package  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
referenceBase	ReferenceBase	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=shortLabel xml.sequenceOffset=10</p>

**Table C.2: ARPackage**

<b>Class</b>	<b>AcObjectSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::RolesAndRights			
<b>Note</b>	<p>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</p> <ul style="list-style-type: none"> <li>• the nature of objects</li> <li>• the involved blueprints</li> <li>• the artifact in which the objects are serialized</li> <li>• the definition of the object (in a definition - value pattern)</li> <li>• individual reference objects</li> </ul> <p><b>Tags:</b> atp.recommendedPackage=AcObjectSets</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
acObjectClass	ReferrableSubtypesEnum	*	attr	This specifies that the considered objects as instances of the denoted meta class.
acScope	AcScopeEnum	1	attr	this indicates the scope of the referenced objects.
collection	<a href="#">Collection</a>	0..1	ref	This indicates that the relevant objects are specified via a collection.
derivedFromBlueprint	<a href="#">AtpBlueprint</a>	*	ref	<p>This association indicates that the considered objects are the ones being derived from the associated blueprint.</p> <p><b>Stereotypes:</b> atpUriDef</p>
engineeringObject	AutosarEngineeringObject	*	aggr	<p>This indicates an engineering object. The AcPermission relates to all objects in this partial model.</p> <p>This also implies that the other objects in this set shall be placed in the specified engineering object.</p> <p>Note that semantic constraints apply with respect to «atpSplitable»</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
object	<a href="#">Referrable</a>	*	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 atpUriDef.  <b>Stereotypes:</b> atpUriDef

**Table C.3: AclObjectSet**

<b>Class</b>	<b>AclOperation</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::RolesAndRights			
<b>Note</b>	This meta class represents the ability to denote a particular operation which may be performed on objects in an AUTOSAR model.  <b>Tags:</b> atp.recommendedPackage=AclOperations			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
impliedOperation	<a href="#">AclOperation</a>	*	ref	This indicates that the related operations are also implied. Therefore the permission is also granted for this operation.

**Table C.4: AclOperation**

<b>Class</b>	<b>AclPermission</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::RolesAndRights			
<b>Note</b>	This meta class represents the ability to represent permissions granted on objects in an AUTOSAR model.  <b>Tags:</b> atp.recommendedPackage=AclPermissions			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
aclContext	NameToken	*	attr	This attribute is intended to specify the context under which the AclPermission is applicable. The values are subject to mutual agreement between the involved stakeholders.  For examples the values can be the names of binding times.
aclObject	<a href="#">AclObjectSet</a>	*	ref	This denotes an object to which the AclPermission applies.

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
aclOperation	<a href="#">AclOperation</a>	*	ref	This denotes an operation which is granted by the given AclPermission.
aclRole	<a href="#">AclRole</a>	*	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 C.5: AclPermission**

<b>Class</b>	<b>AclRole</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::RolesAndRights			
<b>Note</b>	<p>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.</p> <p><b>Tags:</b> atp.recommendedPackage=AclRoles</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
ldapUrl	UriString	0..1	attr	This is an URL which allows to represent users or organizations taking the particular role.

**Table C.6: AclRole**

<b>Class</b>	<b>AliasNameSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::FlatMap			
<b>Note</b>	<p>This meta-class represents a set of AliasNames. The AliasNameSet can for example be an input to the A2L-Generator. It shall not be used by the RTE generator to generate the MC-Support.</p> <p>In a given instance of AliasNameSet in the bound system there must be at most one aliasName per FlatInstanceDescriptor.</p> <p><b>Tags:</b> atp.recommendedPackage=AliasNameSets</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
aliasName	AliasNameAssignment	1..*	aggr	<p>AliasNames contained in the AliasNameSet.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation</p> <p><b>Tags:</b> atp.Splitkey=shortLabel vh.latestBindingTime=preCompileTime</p>

**Table C.7: AliasNameSet**

<b>Class</b>	<b>ApplicationDataType (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	<p>ApplicationDataType defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake.</p> <p>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, endianness, etc.</p> <p>It should be possible to model the application level aspects of a VFB system by using ApplicationDataTypes only.</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table C.8: ApplicationDataType**

<b>Class</b>	<b>AtomicSwComponentType (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	An atomic software component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
internalBehavior	SwcInternalBehavior	0..1	aggr	<p>The SwcInternalBehaviors owned by an AtomicSwComponentType can be located in a different physical file. Therefore the aggregation is «atpSplitable».</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=internalBehavior, variationPoint.shortLabel            vh.latestBindingTime=preCompileTime</p>
symbolProps	SymbolProps	0..1	aggr	<p>This represents the SymbolProps for the AtomicSwComponentType.</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b> atp.Splitkey=shortName</p>

**Table C.9: AtomicSwComponentType**



<b>Class</b>	<b>BswModuleDescription</b>			
<b>Package</b>	M2::AUTOSARTemplates::BswModuleTemplate::BswOverview			
<b>Note</b>	<p>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.</p> <p><b>Tags:</b> atp.recommendedPackage=BswModuleDescriptions</p>			
<b>Base</b>	<p><a href="#">ARElement</a>, <a href="#">ARObject</a>, <a href="#">AtpBlueprint</a>, <a href="#">AtpBlueprintable</a>, <a href="#">AtpClassifier</a>, <a href="#">AtpFeature</a>, <a href="#">AtpStructureElement</a>, <a href="#">CollectableElement</a>, <a href="#">Identifiable</a>, <a href="#">MultilanguageReferrable</a>, <a href="#">PackageableElement</a>, <a href="#">Referrable</a></p>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
bswModuleDependency	<a href="#">BswModuleDependency</a>	*	aggr	<p>Describes the dependency to another BSW module.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20</p>
bswModuleDocumentation	SwComponentDocumentation	0..1	aggr	<p>This adds a documentation to the BSW module.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=bswModuleDocumentation, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=6</p>
internalBehavior	<a href="#">BswInternalBehavior</a>	*	aggr	<p>The various BswInternalBehaviors associated with a BswModuleDescription can be distributed over several physical files. Therefore the aggregation is «atpSplittable».</p> <p><b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=shortName xml.sequenceOffset=65</p>
moduleId	PositiveInteger	0..1	attr	<p>Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen.</p> <p><b>Tags:</b> xml.sequenceOffset=5</p>
outgoingCallback	<a href="#">BswModuleEntry</a>	*	ref	<p>Specifies a callback, which will be called from this module if required by another module.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=outgoingCallback, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=15</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
providedClientServerEntry	BswModuleClientServerEntry	*	aggr	<p>Specifies that this module provides a client server entry which can be called from another partition 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.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45</p>
providedData	VariableDataPrototype	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=55</p>
providedEntry	BswModuleEntry	*	ref	<p>Specifies an entry provided by this module which can be called by other modules. This includes "main" functions and interrupt routines, but not callbacks (because the signature of a callback is defined by the caller).</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=providedEntry, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=10</p>
providedModeGroup	ModeDeclarationGroupPrototype	*	aggr	<p>A set of modes which is owned and provided by this module or cluster. It can be connected to the requiredModeGroups 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, EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25</p>

Attribute	Datatype	Mul.	Kind	Note
releasedTrigger	Trigger	*	aggr	<p>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, EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=35</p>
requiredClientServerEntry	BswModuleClientServerEntry	*	aggr	<p>Specifies that this module requires a client server entry which can be implemented on another partition 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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=50</p>
requiredData	VariableDataPrototype	*	aggr	<p>Specifies a data prototype required by this module in order to be provided from another partition or core. The requiredData 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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=60</p>
requiredModeGroup	ModeDeclarationGroupPrototype	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=30</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
requiredTrigger	Trigger	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=40</p>

**Table C.10: BswModuleDescription**

<b>Enumeration</b>	<b>BindingTimeEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling
<b>Note</b>	This enumerator specifies the applicable binding times for the pre build variation points.
<b>Literal</b>	<b>Description</b>
codeGenerationTime	<ul style="list-style-type: none"> <li>• Coding by hand, based on requirements document.</li> <li>• Tool based code generation, e.g. from a model.</li> <li>• The model may contain variants.</li> <li>• Only code for the selected variant(s) is actually generated.</li> </ul>
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)
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.
systemDesignTime	<ul style="list-style-type: none"> <li>• Designing the VFB.</li> <li>• Software Component types (PortInterfaces).</li> <li>• SWC Prototypes and the Connections between SWCprototypes.</li> <li>• Designing the Topology</li> <li>• ECUs and interconnecting Networks</li> <li>• Designing the Communication Matrix and Data Mapping</li> </ul>

**Table C.11: BindingTimeEnum**

<b>Class</b>	«atpMixedString» <b>BlueprintFormula</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintFormula			
<b>Note</b>	This class express the extension of the Formula Language to provide formalized blueprint-Value resp. blueprintCondition.			
<b>Base</b>	ARObject,FormulaExpression,SwSystemconstDependentFormula			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
ecuc	EcucDefinitionElement	1	ref	The EcucDefinitionElement serves as a argument for the formular.
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 C.12: BlueprintFormula**

<b>Class</b>	<b>BlueprintMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintDedicated::GenericBlueprint			
<b>Note</b>	This meta-class represents the ability to map two an object and its blueprint.			
<b>Base</b>	ARObject, <a href="#">AtpBlueprintMapping</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
blueprint	<a href="#">AtpBlueprint</a>	1	ref	This represents the mapped blueprint.
derivedObject	<a href="#">AtpBlueprintable</a>	1	ref	This represents the object which was derived from the blueprint.

**Table C.13: BlueprintMapping**

<b>Class</b>	<b>BswInternalBehavior</b>			
<b>Package</b>	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
<b>Note</b>	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 BswModuleDescription.			
<b>Base</b>	ARObject,AtpClassifier,AtpFeature,AtpStructureElement, <a href="#">Identifiable</a> ,InternalBehavior,MultiLanguageReferrable, <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
bswPerInstanceMemoryPolicy	BswPerInstanceMemoryPolicy	*	aggr	arTypedPerInstanceMemory specific policy  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
clientPolicy	BswClientPolicy	*	aggr	<b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=clientPolicy, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
distinguish edPartition	BswDistinguish edPartition	*	aggr	Indicates an abstract partition context in which the enclosing BswModuleEntity can be executed.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.ShortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
entity	BswModuleEntit y	1..*	aggr	A code entity for which the behavior is described  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=5
event	BswEvent	*	aggr	An event required by this module behavior.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=10
exclusiveA reaPolicy	BswExclusiveAr eaPolicy	*	aggr	<b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=exclusiveAreaPolicy, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
includedDa taTypeSet	IncludedDataTy peSet	*	aggr	The includedDataTypeSet is used by a basic software module for its implementation.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=includedDataTypeSet
internalTrig geringPoin t	BswInternalTrig geringPoint	*	aggr	An internal triggering point.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=2
internalTrig geringPoin tPolicy	BswInternalTrig geringPointPolic y	*	aggr	<b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=internalTriggeringPointPolicy, variationPoint.shortPoint vh.latestBindingTime=preCompileTime
modeRecei verPolicy	BswModeRecei verPolicy	*	aggr	Implementation policy for the reception of mode switches.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=modeReceiverPolicy, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
modeSenderPolicy	BswModeSenderPolicy	*	aggr	<p>Implementation policy for providing a mode group.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b> atp.Splitkey=modeSenderPolicy, variationPoint.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=20</p>
parameterPolicy	BswParameterPolicy	*	aggr	<p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b> atp.Splitkey=parameterPolicy, variationPoint.shortLabel            vh.latestBindingTime=preCompileTime</p>
perInstanceParameter	ParameterDataPrototype	*	aggr	<p>Describes a read only memory object containing characteristic value(s) needed by this BswInternalBehavior. The role name perInstanceParameter is chosen in analogy to the similar role in the context of SwcInternalBehavior.</p> <p>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.</p> <p>The aggregation is subject to variability with the purpose to support implementation variants.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b> atp.Splitkey=atp.Splitkey shortName, variationPoint.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=45</p>
receptionPolicy	BswDataReceptionPolicy	*	aggr	<p>Data reception policy for inter-partition and/or inter-core communication.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b> atp.Splitkey=receptionPolicy, variationPoint.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=55</p>
releasedTriggerPolicy	BswReleasedTriggerPolicy	*	aggr	<p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b> atp.Splitkey=releasedTriggerPolicy, variationPoint.shortLabel            vh.latestBindingTime=preCompileTime</p>
schedulerNamePrefix	BswSchedulerNamePrefix	*	aggr	<p>Optional definition of one or more prefixes to be used for the BswScheduler.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b> atp.Splitkey=schedulerNamePrefix, variationPoint.ShortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=50</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
sendPolicy	BswDataSendPolicy	*	aggr	<b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=sendPolicy, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
serviceDependency	BswServiceDependency	*	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.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=serviceDependency, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
triggerDirectImplementation	BswTriggerDirectImplementation	*	aggr	Specifies a trigger to be directly implemented via OS calls.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=triggerDirectImplementation, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=15
variationPointProxy	VariationPointProxy	*	aggr	Proxy of a variation points in the C/C++ implementation.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=shortName

**Table C.14: BswInternalBehavior**

<b>Class</b>	<b>BswModuleDependency</b>			
<b>Package</b>	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces			
<b>Note</b>	This class collects the dependencies of a BSW module or cluster on a certain other BSW module.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , MultilanguageReferrable, <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
expectedCallback	<a href="#">BswModuleEntry</a>	*	ref	Indicates a callback expected to be called from another module and implemented by this module.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=15



Attribute	Datatype	Mul.	Kind	Note
requiredEntry	<a href="#">BswModuleEntry</a>	*	ref	<p>Indicates an entry into another modules which is required by this module.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b> vh.latestBindingTime=preCompileTime                      xml.sequenceOffset=10</p>
serviceItem	ServiceNeeds	*	aggr	<p>A single item (example: Nv block) for which the quality of a service is defined.</p> <p>The aggregation is marked as «atpSplitable» to allow for extension during the ECU configuration process.</p> <p>This association is deprecated since R4.0.3, since ServiceNeeds shall be associated with the new element BswServiceDependency within the BswInternalBehavior.</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b> atp.Splitkey=shortName; atp.Status=removed                      xml.sequenceOffset=20</p>
targetModuleId	PositiveInteger	0..1	attr	<p>AUTOSAR identifier of the target module of which the dependencies are defined.</p> <p>This information is optional, because the target module may also be identified by targetModuleRef.</p> <p><b>Tags:</b> xml.sequenceOffset=5</p>
targetModuleRef	<a href="#">BswModuleDescription</a>	0..1	ref	<p>Reference to the target module. It is an «atpUriDef» because the reference shall be used to identify the target module without actually needing the description of that target module.</p> <p><b>Stereotypes:</b> atpUriDef; atpVariation  <b>Tags:</b> vh.latestBindingTime=preCompileTime                      xml.sequenceOffset=7</p>

**Table C.15: BswModuleDependency**

<b>Class</b>	<b>BswModuleDescription</b>			
<b>Package</b>	M2::AUTOSARTemplates::BswModuleTemplate::BswOverview			
<b>Note</b>	<p>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.</p> <p><b>Tags:</b> atp.recommendedPackage=BswModuleDescriptions</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
bswModuleDependency	<a href="#">BswModuleDependency</a>	*	aggr	<p>Describes the dependency to another BSW module.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=20</p>
bswModuleDocumentation	SwComponentDocumentation	0..1	aggr	<p>This adds a documentation to the BSW module.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=bswModuleDocumentation, variationPoint.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=6</p>
internalBehavior	<a href="#">BswInternalBehavior</a>	*	aggr	<p>The various BswInternalBehaviors associated with a BswModuleDescription can be distributed over several physical files. Therefore the aggregation is «atpSplitable».</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b> atp.Splitkey=shortName            xml.sequenceOffset=65</p>
moduleId	PositiveInteger	0..1	attr	<p>Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen.</p> <p><b>Tags:</b> xml.sequenceOffset=5</p>
outgoingCallback	<a href="#">BswModuleEntry</a>	*	ref	<p>Specifies a callback, which will be called from this module if required by another module.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=outgoingCallback, variationPoint.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=15</p>
providedClientServerEntry	BswModuleClientServerEntry	*	aggr	<p>Specifies that this module provides a client server entry which can be called from another partition 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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=45</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
providedData	VariableDataPrototype	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=55</p>
providedEntry	BswModuleEntry	*	ref	<p>Specifies an entry provided by this module which can be called by other modules. This includes "main" functions and interrupt routines, but not callbacks (because the signature of a callback is defined by the caller).</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=providedEntry, variation Point.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=10</p>
providedModeGroup	ModeDeclarationGroupPrototype	*	aggr	<p>A set of modes which is owned and provided by this module or cluster. It can be connected to the requiredModeGroups 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, EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=25</p>
releasedTrigger	Trigger	*	aggr	<p>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, EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel            vh.latestBindingTime=preCompileTime            xml.sequenceOffset=35</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
requiredClientServerEntry	BswModuleClientServerEntry	*	aggr	<p>Specifies that this module requires a client server entry which can be implemented on another partition 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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=50</p>
requiredData	VariableDataPrototype	*	aggr	<p>Specifies a data prototype required by this module in order to be provided from another partition or core. The requiredData 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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60</p>
requiredModeGroup	ModeDeclarationGroupPrototype	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=30</p>
requiredTrigger	Trigger	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40</p>

**Table C.16: BswModuleDescription**

<b>Class</b>	<b>BswModuleEntry</b>			
<b>Package</b>	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces			
<b>Note</b>	<p>This class represents a single API entry (C-function prototype) into the BSW module or cluster.</p> <p>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.</p> <p><b>Tags:</b> atp.recommendedPackage=BswModuleEntrys</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
argument (ordered)	<a href="#">SwServiceArg</a>	*	aggr	<p>An argument belonging to this BswModuleEntry.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=45</p>
callType	BswCallType	1	attr	<p>The type of call associated with this service.</p> <p><b>Tags:</b> xml.sequenceOffset=25</p>
executionContext	BswExecutionContext	1	attr	<p>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.</p> <p><b>Tags:</b> xml.sequenceOffset=30</p>
isReentrant	Boolean	1	attr	<p>Reentrancy from the viewpoint of function callers:</p> <ul style="list-style-type: none"> <li>• True: Enables the service to be invoked again, before the service has finished.</li> <li>• False: It is prohibited to invoke the service again before is has finished.</li> </ul> <p><b>Tags:</b> xml.sequenceOffset=15</p>
isSynchronous	Boolean	1	attr	<p>Synchronicity from the viewpoint of function callers:</p> <ul style="list-style-type: none"> <li>• True: This calls a synchronous service, i.e. the service is completed when the call returns.</li> <li>• False: The service (on semantical level) may not be complete when the call returns.</li> </ul> <p><b>Tags:</b> xml.sequenceOffset=20</p>
returnType	<a href="#">SwServiceArg</a>	0..1	aggr	<p>The return type belonging to this bswModuleEntry.</p> <p><b>Tags:</b> xml.sequenceOffset=40</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
role	<a href="#">Identifier</a>	0..1	ref	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).  <b>Tags:</b> 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.  <b>Tags:</b> xml.sequenceOffset=5
swServiceImpPolicy	SwServiceImpPolicyEnum	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.  <b>Tags:</b> xml.sequenceOffset=35

**Table C.17: BswModuleEntry**

<b>Class</b>	<b>BuildAction</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::BuildActionManifest			
<b>Note</b>	This meta-class represents the ability to specify a build action.			
<b>Base</b>	ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">BuildActionEntity</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
createdData	BuildActionIoElement	*	aggr	This represents the artifacts which are cated by the processor.
followUpAction	<a href="#">BuildAction</a>	*	ref	This association specifies a set of follow up actions.  <b>Tags:</b> 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 modified by the action.
predecessorAction	<a href="#">BuildAction</a>	*	ref	This association specifies a set of predecessors. These actions must be finished before but necessarily immediately after the given action..  These actions need to be performed in the specified order.  <b>Tags:</b> xml.sequenceOffset=-90
requiredEnvironment	<a href="#">BuildActionEnvironment</a>	1	ref	This represents the environment which is required to use the specified Processor.

Attribute	Datatype	Mul.	Kind	Note
-----------	----------	------	------	------

**Table C.18: BuildAction**

<b>Class</b>	<b>BuildActionEnvironment</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::BuildActionManifest			
<b>Note</b>	This meta-class represents the ability to specify a build action environment.			
<b>Base</b>	ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">Identifiable</a> , Multilanguage Referrable, <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
sdg	Sdg	*	aggr	This represents a general data structure intended to denote parameters for the BuildActionEnvironment.

**Table C.19: BuildActionEnvironment**

<b>Class</b>	<b>BuildActionManifest</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::BuildActionManifest			
<b>Note</b>	This meta-class represents the ability to specify a manifest for processing artifacts. An example use case is the processing of ECUC parameter values.  <b>Tags:</b> atp.recommendedPackage=BuildActionManifests xml.globalElement=false			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , Collectable Element, <a href="#">Identifiable</a> , MultilanguageReferrable, <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
buildAction	<a href="#">BuildAction</a>	*	aggr	This represents a particular action in the build chain.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=blueprintDerivationTime
buildActionEnvironment	<a href="#">BuildActionEnvironment</a>	*	aggr	This represents a build action environment.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=blueprintDerivationTime
dynamicAction	<a href="#">BuildAction</a>	*	ref	This denotes an Action which is to be executed as part of the dynamic action set.
startAction	<a href="#">BuildAction</a>	*	ref	This specifies the list of actions to be performed at the beginning of the process.  <b>Tags:</b> xml.sequenceOffset=-90
tearDownAction	<a href="#">BuildAction</a>	*	ref	This specifies the set of action which shall be performed after all other actions in the manifest were performed.  <b>Tags:</b> xml.sequenceOffset=-80

**Table C.20: BuildActionManifest**

<b>Class</b>	<b>Collection</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Element Collection			
<b>Note</b>	<p>This meta-class specifies a collection of elements. A collection can be utilized to express additional aspects for a set of elements.</p> <p>Note that Collection is an ARElement. Therefore it is applicable e.g. for EvaluatedVariant, even if this is not obvious.</p> <p>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]).</p> <p>In this case the collection represents an association from "sourceElement" to "targetElement" in the role "role".</p> <p><b>Tags:</b> atp.recommendedPackage=Collections</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
autoCollect	AutoCollectEnum	0..1	attr	<p>This attribute reflects how far the referenced objects are part of the collection.</p> <p><b>Tags:</b> xml.sequenceOffset=20</p>
collectedInstance	AtpFeature	*	iref	<p>This instance ref supports the use case that a particular instance is part of the collection.</p> <p><b>Tags:</b> xml.sequenceOffset=60</p>
element	<a href="#">Identifiable</a>	*	ref	<p>This is an element in the collection. Note that Collection itself is collectable. Therefore collections can be nested.</p> <p>In case of category="RELATION" this represents the target end of the relation.</p> <p><b>Tags:</b> xml.sequenceOffset=40</p>
elementRole	<a href="#">Identifier</a>	0..1	ref	<p>This attribute allows to denote a particular role of the collection. Note that the applicable semantics shall be mutually agreed between the two parties.</p> <p>In particular it denotes the role of element in the context of sourceElement.</p> <p><b>Tags:</b> xml.sequenceOffset=30</p>
sourceElement	<a href="#">Identifiable</a>	*	ref	<p>Only if Category = "RELATION". This represents the source of a relation.</p> <p><b>Tags:</b> xml.sequenceOffset=50</p>
sourceInstance	AtpFeature	*	iref	<p>Only if Category = "RELATION". This represents the source instance of a relation.</p> <p><b>Tags:</b> xml.sequenceOffset=70</p>

**Table C.21: Collection**



<b>Class</b>	<b>CompuMethod</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
<b>Note</b>	<p>This meta-class represents the ability to express the relationship between a physical value and the mathematical representation.</p> <p>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.</p> <p><b>Tags:</b> atp.recommendedPackage=CompuMethods</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
compuInternalToPhys	Compu	0..1	aggr	<p>This specifies the computation from internal values to physical values.</p> <p><b>Tags:</b> xml.sequenceOffset=80</p>
compuPhysToInternal	Compu	0..1	aggr	<p>This represents the computation from physical values to the internal values.</p> <p><b>Tags:</b> xml.sequenceOffset=90</p>
displayFormat	DisplayFormatString	0..1	attr	<p>This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools.</p> <p><b>Tags:</b> xml.sequenceOffset=20</p>
unit	Unit	0..1	ref	<p>This is the physical unit of the Physical values for which the CompuMethod applies.</p> <p><b>Tags:</b> xml.sequenceOffset=30</p>

**Table C.22: CompuMethod**

<b>Class</b>	<b>CompuScale</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
<b>Note</b>	This meta-class represents the ability to specify one segment of a segmented computation method.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
desc	MultiLanguageOverviewParagraph	0..1	aggr	<p>&lt;desc&gt; represents a general but brief description of the object in question.</p> <p><b>Tags:</b> xml.sequenceOffset=30</p>
compuInverseValue	CompuConst	0..1	aggr	<p>This is the inverse value of the constraint. This supports the case that the scale is not reversible per se.</p> <p><b>Tags:</b> xml.sequenceOffset=60</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
compuScaleContents	CompuScaleContents	0..1	aggr	<p>This represents the computation details of the scale.</p> <p><b>Tags:</b> xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=70; xml.typeElement=false; xml.typeWrapperElement=false</p>
lowerLimit	Limit	0..1	ref	<p>This specifies the lower limit of the scale.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=40</p>
mask	PositiveInteger	0..1	attr	<p>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.</p> <p>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.</p> <p>The processing has to be done in order of the COMPU-SCALE elements.</p> <p><b>Tags:</b> xml.sequenceOffset=35</p>
shortLabel	Identifier	0..1	ref	<p>This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier.</p> <p><b>Tags:</b> xml.sequenceOffset=20</p>
symbol	CIdentifier	0..1	ref	<p>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.</p> <p><b>Tags:</b> xml.sequenceOffset=25</p>
upperLimit	Limit	0..1	ref	<p>This specifies the upper limit of a of the scale.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=50</p>

**Table C.23: CompuScale**

<b>Class</b>	<b>ConsistencyNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
<b>Note</b>	This meta-class represents the ability to define requirements on the implicit communication behavior.			
<b>Base</b>	ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">Identifiable</a> , Multilanguage Referrable, <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
dpgDoesNotRequireCoherency	<a href="#">DataPrototypeGroup</a>	*	aggr	<p>This group of VariableDataPrototypes does not require coherency with respect to the implicit communication behavior.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime</p>
dpgRequiresCoherency	<a href="#">DataPrototypeGroup</a>	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime</p>
regDoesNotRequireStability	<a href="#">RunnableEntityGroup</a>	*	aggr	<p>This group of RunnableEntitys does not require stability with respect to the implicit communication behavior.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime</p>
regRequiresStability	<a href="#">RunnableEntityGroup</a>	*	aggr	<p>This group of RunnableEntitys 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.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime</p>

**Table C.24: ConsistencyNeeds**

<b>Class</b>	<b>ConsistencyNeedsBlueprintSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintDedicated::ConsistencyNeedsBlueprintSet			
<b>Note</b>	This meta class represents the ability to specify a set of blueprint for ConsistencyNeeds.  <b>Tags:</b> atp.recommendedPackage=ConsistencyNeedsBlueprintSets			
<b>Base</b>	AElement,ARObject,CollectableElement,Identifiable,Multilanguage Referrable,PackageableElement,Referrable			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
consistencyNeeds	ConsistencyNeeds	*	aggr	This represents a particular blueprint of consistencyNeeds. Note that it is  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table C.25: ConsistencyNeedsBlueprintSet**

<b>Class</b>	<b>DataConstr</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::GlobalConstraints			
<b>Note</b>	This meta-class represents the ability to specify constraints on data.  <b>Tags:</b> atp.recommendedPackage=DataConstrs			
<b>Base</b>	AElement,ARObject,AtpBlueprint,AtpBlueprintable,Collectable Element,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
dataConstrRule	DataConstrRule	*	aggr	This is one particular rule within the data constraints.  <b>Tags:</b> xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false

**Table C.26: DataConstr**

<b>Class</b>	<b>DataPrototypeGroup</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
<b>Note</b>	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.			
<b>Base</b>	ARObject,AtpClassifier,AtpFeature,AtpStructureElement,Identifiable,Multilanguage Referrable,Referrable			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
dataPrototypeGroup	DataPrototypeGroup	*	iref	This represents the ability to define nested groups of VariableDataPrototypes.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
implicitDataAccess	VariableDataPrototype	*	iref	This represents a collection of VariableDataPrototypes that belong to the enclosing DataPrototypeGroup  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table C.27: DataPrototypeGroup**

<b>Class</b>	<b>DataTypeMappingSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	This class represents a list of mappings between ApplicationDataTypes and ImplementationDataTypes. In addition, it can contain mappings between ImplementationDataTypes and ModeDeclarationGroups.  <b>Tags:</b> atp.recommendedPackage=DataTypeMappingSets			
<b>Base</b>	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
dataTypeMap	DataTypeMap	*	aggr	This is one particular association between an ApplicationDataType and its ImplementationDataType.
modeRequestTypeMap	ModeRequestTypeMap	*	aggr	This is one particular association between an ModeDeclarationGroup and its ImplementationDataType.

**Table C.28: DataTypeMappingSet**

<b>Class</b>	<b>Documentation</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::DocumentationOnM1			
<b>Note</b>	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.  <b>Tags:</b> atp.recommendedPackage=Documentations			
<b>Base</b>	ARElement,ARObject,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
context	DocumentationContext	*	aggr	This is the context of the particular documentation.
documentationContent	PredefinedChapter	0..1	aggr	This is the content of the documentation related to the specified contexts.  <b>Tags:</b> xml.sequenceOffset=200

**Table C.29: Documentation**

<b>Class</b>	«atpMixed» DocumentationBlock			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Documentation::BlockElements			
<b>Note</b>	This class represents a documentation block. It is made of basic text structure elements which can be displayed in a table cell.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
defList	DefList	0..1	aggr	This represents a definition list in the documentation block.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=40
figure	MIFigure	0..1	aggr	This represents a figure in the documentation block.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=70
formula	MIFormula	0..1	aggr	This is a formula in the definition block.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=60
labeledList	LabeledList	0..1	aggr	This represents a labeled list.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=50
list	List	0..1	aggr	This represents numbered or unnumbered list.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=30
msrQuery P2	MsrQueryP2	0..1	aggr	
note	Note	0..1	aggr	This represents a note in the text flow.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=80
p	MultiLanguageParagraph	0..1	aggr	This is one particular paragraph.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=10
structured Req	<a href="#">StructuredReq</a>	0..1	aggr	This aggregation supports structured requirements embedded in a documentation block.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=100

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
trace	<a href="#">TraceableText</a>	0..1	aggr	<p>This represents traceable text in the documentation block. This allows to specify requirements/constraints in any documentation block.</p> <p>The kind of the trace is specified in the category.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=90</p>
verbatim	MultiLanguageVerbatim	0..1	aggr	<p>This represents one particular verbatim text.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild xml.sequenceOffset=20</p>

**Table C.30: DocumentationBlock**

<b>Class</b>	<b>EcucDefinitionCollection</b>			
<b>Package</b>	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
<b>Note</b>	<p>This represents the anchor point of an ECU Configuration Parameter Definition within the AUTOSAR templates structure.</p> <p><b>Tags:</b> atp.recommendedPackage=EcucDefinitionCollections</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
module	<a href="#">EcucModuleDef</a>	1..*	ref	References to the module definitions of individual software modules.

**Table C.31: EcucDefinitionCollection**

<b>Class</b>	<b>EcucModuleDef</b>			
<b>Package</b>	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
<b>Note</b>	<p>Used as the top-level element for configuration definition for Software Modules, including BSW and RTE as well as ECU Infrastructure.</p> <p><b>Tags:</b> atp.recommendedPackage=EcucModuleDefs</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpDefinition</a> , <a href="#">CollectableElement</a> , <a href="#">EcucDefinitionElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
apiServicePrefix	<a href="#">CIdentifier</a>	0..1	ref	<p>For CDD modules this attribute holds the apiServicePrefix.</p> <p>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.</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
container	EcucContainerDef	1..*	aggr	Aggregates the top-level container definitions of this specific module definition.  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=shortName xml.sequenceOffset=11
postBuildVariantSupport	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.
refinedModuleDef	<a href="#">EcucModuleDef</a>	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.  <b>Stereotypes:</b> atpUriDef
supportedConfigVariant	EcucConfigurationVariantEnum	*	attr	Specifies which ConfigurationVariants are supported by this software module. This attribute is optional if the EcucModuleDef 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 C.32: EcucModuleDef**

<b>Class</b>	<b>FlatMap</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::FlatMap			
<b>Note</b>	<p>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.</p> <p>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.</p> <p><b>Tags:</b> atp.recommendedPackage=FlatMaps</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>



<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
instance	FlatInstanceDescriptor	1..*	aggr	<p>A descriptor instance aggregated in the flat map.</p> <p>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.</p> <p>The aggregation has been made splittable 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.</p> <p><b>Stereotypes:</b> atpSplittable; atpVariation  <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel            vh.latestBindingTime=postBuild</p>

**Table C.33: FlatMap**

<b>Class</b>	<b>Identifiable (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
<b>Note</b>	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.			
<b>Base</b>	ARObject, MultilanguageReferrable, <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
desc	MultiLanguageOverviewParagraph	0..1	aggr	<p>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.</p> <p>More elaborate documentation, (in particular how the object is built or used) should go to "introduction".</p> <p><b>Tags:</b> xml.sequenceOffset=-60</p>
category	CategoryString	0..1	attr	<p>The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints.</p> <p><b>Tags:</b> xml.sequenceOffset=-50</p>
adminData	AdminData	0..1	aggr	<p>This represents the administrative data for the identifiable object.</p> <p><b>Tags:</b> xml.sequenceOffset=-40</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
annotation	Annotation	*	aggr	<p>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.</p> <p><b>Tags:</b> xml.sequenceOffset=-25</p>
introduction	<a href="#">Documentation Block</a>	0..1	aggr	<p>This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock.</p> <p><b>Tags:</b> xml.sequenceOffset=-30</p>
uuid	String	0..1	attr	<p>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.</p> <p><b>Tags:</b> xml.attribute=true</p>

**Table C.34: Identifiable**

<b>Class</b>	<b>ImplementationDataType</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
<b>Note</b>	<p>Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code.</p> <p><b>Tags:</b> atp.recommendedPackage=ImplementationDataTypes</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">Autosar DataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Packageable Element</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
subElement (ordered)	ImplementationDataTypeElement	*	aggr	Specifies an element of an array, struct, or union data type.  The aggregation of ImplementationDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a ImplementationDataType representing a structure.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the ImplementationDataType.  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=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 C.35: ImplementationDataType**

<b>Class</b>	<b>LifeCycleInfo</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::LifeCycles			
<b>Note</b>	LifeCycleInfo describes the life cycle state of an element together with additional information like what to use instead			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
lcObject	<a href="#">Referrable</a>	1	ref	Element(s) have the life cycle as described in lcState.
lcState	<a href="#">LifeCycleState</a>	0..1	ref	This denotes the particular state assigned to the object. If no lcState is given then the default life cycle state of LifeCycleInfoSet 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 LifeCycleInfoSet is assumed.
remark	<a href="#">Documentation Block</a>	0..1	aggr	Remark describing for example <ul style="list-style-type: none"> <li>• why the element was given the specified life cycle</li> <li>• the semantics of useInstead</li> </ul>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
useInstead	<a href="#">Referrable</a>	*	ref	Element(s) that should be used instead of the one denoted in <a href="#">referrable</a> .  Only relevant in case of life cycle states <a href="#">lcState</a> unlike "valid". In case there are multiple references the exact semantics must be individually described in the remark.

**Table C.36: LifeCycleInfo**

<b>Class</b>	<b>LifeCycleInfoSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::LifeCycles			
<b>Note</b>	<p>This meta class represents the ability to attach a life cycle information to a particular set of elements.</p> <p>The information can be defined for a particular period. This supports the definition of transition plans.</p> <p>If no period is specified, the life cycle state applies forever.</p> <p><b>Tags:</b> atp.recommendedPackage=LifeCycleInfoSets</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
defaultLcState	<a href="#">LifeCycleState</a>	1	ref	This denotes the default life cycle state. To be used in all <a href="#">LifeCycleInfo</a> elements within the <a href="#">LifeCycleInfoSet</a> if no life cycle state is stated there explicitly. I.e. the <a href="#">defaultLcState</a> can be overwritten in <a href="#">LifeCycleInfo</a> elements.
defaultPeriodBegin	<a href="#">LifeCyclePeriod</a>	0..1	aggr	Default starting point of period in which all the specified <a href="#">LifeCycleInfo</a> apply. Note that the default period can be overridden for each <a href="#">LifeCycleInfo</a> individually.
defaultPeriodEnd	<a href="#">LifeCyclePeriod</a>	0..1	aggr	Default expiry date, i.e. default end point of period for which all specified <a href="#">LifeCycleInfo</a> apply. Note that the default period can be overridden for each <a href="#">LifeCycleInfo</a> individually.
lifeCycleInfo	<a href="#">LifeCycleInfo</a>	*	aggr	This represents one particular life cycle information.
usedLifeCycleStateDefinitionGroup	<a href="#">LifeCycleStateDefinitionGroup</a>	1	ref	This denotes the life cycle states applicable to the current life cycle info set.

**Table C.37: LifeCycleInfoSet**

<b>Class</b>	<b>LifeCycleState</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::LifeCycles			
<b>Note</b>	This meta class represents one particular state in the LifeCycle.			
<b>Base</b>	ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">Identifiable</a> , Multilanguage Referrable, <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table C.38: LifeCycleState**

<b>Class</b>	<b>LifeCycleStateDefinitionGroup</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::LifeCycles			
<b>Note</b>	This meta class represents the ability to define the states and properties of one particular life cycle.			
	<b>Tags:</b> atp.recommendedPackage=LifeCycleStateDefintionGroups			
<b>Base</b>	<a href="#">ARElement</a> , ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , Collectable Element, <a href="#">Identifiable</a> , MultilanguageReferrable, <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
lcState	<a href="#">LifeCycleState</a>	*	aggr	Describes a single life cycle state of this life cycle state definition group.

**Table C.39: LifeCycleStateDefinitionGroup**

<b>Class</b>	<b>ModeDeclarationGroup</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
<b>Note</b>	A collection of Mode Declarations. Also, the initial mode is explicitly identified.			
	<b>Tags:</b> atp.recommendedPackage=ModeDeclarationGroups			
<b>Base</b>	<a href="#">ARElement</a> , ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , MultilanguageReferrable, <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
initialMode	ModeDeclaration	1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred.
modeDeclaration	ModeDeclaration	1..*	aggr	The ModeDeclarations collected in this ModeDeclarationGroup.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=blueprintDerivationTime
modeManagerErrorBehavior	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 available ModeTransitions of the ModeDeclarationGroup

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
modeUserErrorBehavior	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).
onTransitionValue	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 C.40: ModeDeclarationGroup**

<b>Class</b>	<b>NonqueuedReceiverComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Communication attributes specific to non-queued receiving.			
<b>Base</b>	ARObject, <a href="#">RPortComSpec</a> , ReceiverComSpec			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
aliveTimeout	TimeValue	1	attr	<p>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.</p> <p>If the aliveTimeout attribute is 0 no timeout monitoring shall be performed.</p>
enableUpdate	Boolean	1	attr	This attribute controls whether application code is entitled to check whether the value of the corresponding VariableDataPrototype has been updated.
filter	DataFilter	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement.
handleDataStatus	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.
handleNeverReceived	Boolean	1	attr	<p>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.</p> <ul style="list-style-type: none"> <li>• If the value of this attribute is set to "true" the flag is required.</li> <li>• If set to "false", the RTE shall not support the "never received" functionality for the corresponding VariableDataPrototype.</li> </ul>
handleTimeoutType	HandleTimeoutEnum	1	attr	This attribute controls the behavior with respect to the handling of timeouts.

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
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.

**Table C.41: NonqueuedReceiverComSpec**

<b>Class</b>	<b>NonqueuedSenderComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	Communication attributes for non-queued sender/receiver communication (sender side)			
<b>Base</b>	ARObject, <a href="#">PPortComSpec</a> , SenderComSpec			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
initValue	ValueSpecification	1	aggr	Initial value to be sent if sender component is not yet fully initialized, but receiver needs data already.

**Table C.42: NonqueuedSenderComSpec**

<b>Class</b>	<b>PPortComSpec (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table C.43: PPortComSpec**

<b>Class</b>	<b>PPortPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	Component port providing a certain port interface.			
<b>Base</b>	ARObject, AbstractProvidedPortPrototype, <a href="#">AtpBlueprintable</a> , AtpFeature, AtpPrototype, <a href="#">Identifiable</a> , MultilanguageReferrable, <a href="#">PortPrototype</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
providedInterface	<a href="#">PortInterface</a>	1	tref	The interface that this port provides.  <b>Stereotypes:</b> isOfType

**Table C.44: PPortPrototype**

<b>Class</b>	<b>PRPortPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	This kind of PortPrototype can take the role of both a required and a provided PortPrototype.			
<b>Base</b>	ARObject, AbstractProvidedPortPrototype, AbstractRequiredPortPrototype, <a href="#">AtpBlueprintable</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PortPrototype</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
providedRequiredInterface	<a href="#">PortInterface</a>	1	tref	This represents the PortInterface used to type the PRPortPrototype  <b>Stereotypes:</b> isOfType

**Table C.45: PRPortPrototype**

<b>Class</b>	<b>PackageableElement (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
<b>Note</b>	This meta-class specifies the ability to be a member of an AUTOSAR package.			
<b>Base</b>	ARObject, CollectableElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table C.46: PackageableElement**

<b>Class</b>	<b>PortInterface (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	Abstract base class for an interface that is either provided or required by a port of a software component.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
isService	Boolean	1	attr	This flag is set if the PortInterface is to be used for communication between an <ul style="list-style-type: none"> <li>• ApplicationSwComponentType or</li> <li>• ServiceProxySwComponentType or</li> <li>• SensorActuatorSwComponentType or</li> <li>• ComplexDeviceDriverSwComponentType</li> <li>• ServiceSwComponentType</li> <li>• EcuAbstractionSwComponentType</li> </ul> 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.



<i>Attribute</i>	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>
------------------	-----------------	-------------	-------------	-------------

**Table C.47: PortInterface**

<b>Class</b>	<b>PortInterfaceMapping (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	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).			
<b>Base</b>	ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">Identifiable</a> , Multilanguage Referrable, <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table C.48: PortInterfaceMapping**

<b>Class</b>	<b>PortInterfaceMappingSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	Specifies a set of (one or more) PortInterfaceMappings.  <b>Tags:</b> atp.recommendedPackage=PortInterfaceMappingSets			
<b>Base</b>	<a href="#">ARElement</a> , ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , Collectable Element, <a href="#">Identifiable</a> , MultilanguageReferrable, <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
portInterfaceMapping	<a href="#">PortInterfaceMapping</a>	1..*	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).  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=blueprintDerivationTime

**Table C.49: PortInterfaceMappingSet**

<b>Class</b>	<b>PortPrototype (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">AtpBlueprintable</a> , AtpFeature, AtpPrototype, <a href="#">Identifiable</a> , Multilanguage Referrable, <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
clientServerAnnotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/server communication.

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
delegatedPortAnnotation	DelegatedPortAnnotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstractionServerAnnotation	IoHwAbstractionServerAnnotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePortAnnotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPortAnnotation	NvDataPortAnnotation	*	aggr	Annotations on this non volatile data port.
parameterPortAnnotation	ParameterPortAnnotation	*	aggr	Annotations on this parameter port.
senderReceiverAnnotation	SenderReceiverAnnotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPortAnnotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

**Table C.50: PortPrototype**

<b>Class</b>	<b>RPortComSpec (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table C.51: RPortComSpec**

<b>Class</b>	<b>RPortPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	Component port requiring a certain port interface.			
<b>Base</b>	ARObject, AbstractRequiredPortPrototype, <a href="#">AtpBlueprintable</a> , AtpFeature, AtpPrototype, <a href="#">Identifiable</a> , MultilanguageReferrable, <a href="#">PortPrototype</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
requiredInterface	<a href="#">PortInterface</a>	1	tref	The interface that this port requires, i.e. the port depends on another port providing the specified interface.  <b>Stereotypes:</b> isOfType

**Table C.52: RPortPrototype**

<b>Class</b>	<b>Referrable (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
<b>Note</b>	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
shortName	Identifier	1	ref	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.  <b>Tags:</b> xml.enforceMinMultiplicity=true; xml.sequenceOffset=-100
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments.  <b>Tags:</b> xml.sequenceOffset=-90

**Table C.53: Referrable**

<b>Class</b>	<b>RunnableEntity</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
<b>Note</b>	A RunnableEntity represents the smallest code-fragment that is provided by an AtomicSwComponentType 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.			
<b>Base</b>	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Executable Entity, Identifiable, MultilanguageReferrable, Referrable			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
argument (ordered)	RunnableEntityArgument	*	aggr	This represents the formal definition of a an argument to a RunnableEntity.
asynchronousServerCallResultPoint	AsynchronousServerCallResultPoint	*	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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
canBeInvokedConcurrently	Boolean	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 AtomicSwComponentType). 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".

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
dataReadAccess	VariableAccess	*	aggr	<p>RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.</p> <p>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.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
dataReceivePointByArgument	VariableAccess	*	aggr	<p>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.</p> <p>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.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
dataReceivePointByValue	VariableAccess	*	aggr	<p>RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.</p> <p>The result is passed back to the application by means of the return value. The aggregation of dataReceivePointByValue 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.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
dataSendPoint	VariableAccess	*	aggr	<p>RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.</p> <p>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.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
dataWriteAccess	VariableAccess	*	aggr	<p>RunnableEntity has implicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.</p> <p>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.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
externalTriggeringPoint	ExternalTriggeringPoint	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
internalTriggeringPoint	InternalTriggeringPoint	*	aggr	<p>The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
modeAccessPoint	ModeAccessPoint	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
modeSwitchPoint	ModeSwitchPoint	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
parameter Access	ParameterAccess	*	aggr	<p>The presence of a ParameterAccess implies that a RunnableEntity needs read only access to a ParameterDataPrototype which may either be local or within a PortPrototype.</p> <p>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 ParameterAccess (points) in the implementation.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
readLocal Variable	VariableAccess	*	aggr	<p>The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableDataPrototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable.</p> <p>The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicitInterRunnableVariable or the variant existence of readLocalVariable (points) in the implementation.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
serverCall Point	ServerCallPoint	*	aggr	<p>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.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
symbol	<a href="#">CIdentifier</a>	1	ref	<p>The symbol describing this RunnableEntity's entry point. This is considered the API of the RunnableEntity and is required during the RTE contract phase.</p>
waitPoint	WaitPoint	*	aggr	<p>The WaitPoint associated with the RunnableEntity.</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
writtenLocalVariable	VariableAccess	*	aggr	<p>The presence of a writtenLocalVariable implies that a RunnableEntity needs write access to a VariableDataPrototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable.</p> <p>The aggregation of writtenLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicitInterRunnableVariable or the variant existence of writtenLocalVariable (points) in the implementation.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>

**Table C.54: RunnableEntity**

<b>Class</b>	<b>RunnableEntityGroup</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
<b>Note</b>	This meta-class represents the ability to define a collection of RunnableEntities. The collection can be nested.			
<b>Base</b>	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, <a href="#">Identifiable</a> , Multilanguage Referrable, <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
runnableEntity	<a href="#">RunnableEntity</a>	*	iref	<p>This represents a collection of RunnableEntities that belong to the enclosing RunnableEntityGroup.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
runnableEntityGroup	<a href="#">RunnableEntityGroup</a>	*	iref	<p>This represents the ability to define nested groups of RunnableEntities.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>

**Table C.55: RunnableEntityGroup**

<b>Primitive</b>	<b>SectionInitializationPolicyType</b>
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes

<b>Note</b>	<p>SectionInitializationPolicyType describes the intended initialization of MemorySections. The following values are standardized in AUTOSAR Methodology:</p> <ul style="list-style-type: none"> <li>• <b>NO-INIT</b>: No initialization and no clearing is performed. Such data elements shall not be read before one has written a value into it.</li> <li>• <b>INIT</b>: To be used for data that are initialized by every reset to the specified value (initValue).</li> <li>• <b>POWER-ON-INIT</b>: 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.</li> <li>• <b>CLEARED</b>: To be used for data that are initialized by every reset to zero.</li> <li>• <b>POWER-ON-CLEARED</b>: To be used for data that are initialized by "Power On" to zero. Note: there might be several resets between power on resets.</li> </ul> <p>Please note that the values are defined similar to the representation of enumeration types in the XML schema to ensure backward compatibility.</p> <p><b>Tags:</b> xml.xsd.customType=SECTION-INITIALIZATION-POLICY-TYPE; xml.xsd.type=NMOKEN</p>
-------------	---

**Table C.56: SectionInitializationPolicyType**

<b>Class</b>	<b>SenderReceiverInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	A sender/receiver interface declares a number of data elements to be sent and received.			
	<b>Tags:</b> atp.recommendedPackage=PortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DataInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
dataElement	<a href="#">VariableDataPrototype</a>	1..*	aggr	The data elements of this SenderReceiverInterface.
invalidationPolicy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement

**Table C.57: SenderReceiverInterface**



<b>Class</b>	<b>SwAddrMethod</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::AuxillaryObjects			
<b>Note</b>	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.  <b>Tags:</b> atp.recommendedPackage=SwAddrMethods			
<b>Base</b>	AElement,ARObject,AtpBlueprint,AtpBlueprintable,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
memoryAllocationKeywordPolicy	MemoryAllocationKeywordPolicyType	0..1	attr	Enumeration to specify the name pattern of the Memory Allocation Keyword.
option	Identifier	*	ref	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 addressing method.

**Table C.58: SwAddrMethod**

<b>Class</b>	<b>SwBaseType</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::BaseTypes			
<b>Note</b>	This meta-class represents a base type used within ECU software.  <b>Tags:</b> atp.recommendedPackage=BaseTypes			
<b>Base</b>	AElement,ARObject,AtpBlueprint,AtpBlueprintable,BaseType,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table C.59: SwBaseType**

<b>Class</b>	<b>SwComponentPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
<b>Note</b>	Role of a software component within a composition.			
<b>Base</b>	ARObject, AtpFeature, AtpPrototype, <a href="#">Identifiable</a> , MultilanguageReferrable, <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
type	<a href="#">SwComponentType</a>	1	tref	Type of the instance.  <b>Stereotypes:</b> isOfType

**Table C.60: SwComponentPrototype**

<b>Class</b>	<b>SwComponentType (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	Base class for AUTOSAR software components.			
<b>Base</b>	<a href="#">ARElement</a> , ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , AtpClassifier, AtpType, CollectableElement, <a href="#">Identifiable</a> , MultilanguageReferrable, <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
consistencyNeeds	<a href="#">ConsistencyNeeds</a>	*	aggr	This represents the collection of ConsistencyNeeds owned by the enclosing SwComponentType.  <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
port	<a href="#">PortPrototype</a>	*	aggr	The ports through which this component can communicate. The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes.  <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
swComponentDocumentation	SwComponentDocumentation	0..1	aggr	This adds a documentation to the SwComponentType.  <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=swComponentDocumentation, 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 C.61: SwComponentType**

<b>Class</b>	<b>SwServiceArg</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceProcessTask			
<b>Note</b>	<p>Specifies the properties of a data object exchanged during the call of an SwService, e.g. an argument or a return value.</p> <p>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.</p>			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
direction	ArgumentDirectionEnum	0..1	attr	<p>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.</p> <p>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".</p> <p><b>Tags:</b> xml.sequenceOffset=10</p>
swArraysize	ValueList	0..1	aggr	<p>This turns the argument of the service to an array.</p> <p><b>Tags:</b> xml.sequenceOffset=20</p>
swDataDefProps	SwDataDefProps	0..1	aggr	<p>Data properties of this SwServiceArg.</p> <p><b>Tags:</b> xml.sequenceOffset=30</p>

**Table C.62: SwServiceArg**

<b>Class</b>	<b>TDEventVfbPort (abstract)</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventVfb			
<b>Note</b>	This is the abstract parent class to describe specific timing event types at Virtual Function Bus (VFB) level.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , TDEventVfb, TimingDescription, TimingDescriptionEvent			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
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 Function Bus (VFB) level.
port	<a href="#">PortPrototype</a>	0..1	ref	The port scope of the timing event.
portPrototypeBlueprint	<a href="#">PortPrototypeBlueprint</a>	0..1	ref	The PortPrototypeBlueprint is the scope of the timing event.

**Table C.63: TDEventVfbPort**

<b>Class</b>	<b>VariableDataPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	<p>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.</p> <p>In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes.</p>			
<b>Base</b>	ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype, DataPrototype, <a href="#">Identifiable</a> , MultilanguageReferrable, <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

**Table C.64: VariableDataPrototype**

<b>Class</b>	<b>VfbTiming</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Timing			
<b>Note</b>	<p>A model element used to define timing descriptions and constraints at VFB level.</p> <p>TimingDescriptions aggregated by VfbTiming are restricted to event chains referring to events which are derived from the class TDEventVfb.</p> <p><b>Tags:</b> atp.recommendedPackage=TimingExtensions</p>			
<b>Base</b>	<a href="#">ARElement</a> , ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , CollectableElement, <a href="#">Identifiable</a> , MultilanguageReferrable, <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , TimingExtension			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
component	<a href="#">SwComponentType</a>	1	ref	This defines the scope of a VfbTiming. All corresponding timing descriptions and constraints must be defined within this scope.

**Table C.65: VfbTiming**

<b>Class</b>	<b>VariationPoint</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
<b>Note</b>	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 postBuildVariantCriterion is fulfilled.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
desc	MultiLanguageOverviewParagraph	0..1	aggr	<p>This allows to describe shortly the purpose of the variation point.</p> <p><b>Tags:</b> xml.sequenceOffset=20</p>

<b>Attribute</b>	<b>Datatype</b>	<b>Mul.</b>	<b>Kind</b>	<b>Note</b>
blueprintCondition	Documentation Block	0..1	aggr	<p>This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint.</p> <p>Note that variationPoints are not allowed within a blueprintCondition.</p> <p><b>Tags:</b> xml.sequenceOffset=28</p>
formalBlueprintCondition	BlueprintFormula	0..1	aggr	<p>This denotes a formal blueprintCondition. This shall be not in contradiction with blueprintCondition. It is recommended only to use one of the two.</p> <p><b>Tags:</b> xml.sequenceOffset=29</p>
postBuildVariantCondition	PostBuildVariantCondition	*	aggr	<p>This is the set of post build variant conditions which all shall be fulfilled in order to (postbuild) bind the variation point.</p> <p><b>Tags:</b> xml.sequenceOffset=40</p>
sdg	Sdg	0..1	aggr	<p>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.</p> <p><b>Tags:</b> xml.sequenceOffset=50</p>
shortLabel	Identifier	0..1	ref	<p>This provides a name to the particular variation point to support the RTE generator. It is necessary for supporting splittable 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.</p> <p><b>Tags:</b> xml.sequenceOffset=10</p>
swSyscond	ConditionByFormula	0..1	aggr	<p>This condition acts as Binding Function for the VariationPoint. Note that the multiplicity is 0..1 in order to support pure postBuild variants.</p> <p><b>Tags:</b> xml.sequenceOffset=30</p>

**Table C.66: VariationPoint**

## D Variation Points in this Template

<b>Variation Point</b>	<b>Latest Binding Time</b>
BlueprintPolicyList.maxNumberOfElements	blueprintDerivationTime
BlueprintPolicyList.minNumberOfElements	blueprintDerivationTime
ConsistencyNeedsBlueprintSet.consistencyNeeds	preCompileTime

---

**Table D.1: Usage of variation points**