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

Document Version	1.3.0
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Description
31.03.2014	1.3.0	AUTOSAR Release Management	• editorial changes including tagged specification items • update content of specification levels
29.10.2013	1.2.0	AUTOSAR Release Management	• editorial changes including tagged specification items • extension of blueprinting to further AUTOSAR classes
28.02.2013	1.1.0	AUTOSAR Administration	• editorial changes including tagged specification items • extension of blueprinting to further AUTOSAR classes (e.g. build action manifest) • introduction of life cycle support • improvement of document traceability • refinement of traceability support
30.10.2011	1.0.0	AUTOSAR Administration	Initial Release

Table of Contents

References

[1] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[2] Requirements on Standardization Template
AUTOSAR_RS_StandardizationTemplate

[3] Specification of Predefined Names in AUTOSAR
AUTOSAR_TR_PredefinedNames

[4] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList

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

[6] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

[7] ANTLR parser generator V3

[8] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration

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

[10] Table of Application Interfaces
AUTOSAR_MOD_AITable

[11] Specification of Timing Extensions
AUTOSAR_TPS_TimingExtensions

[12] Explanation of Application Interfaces of the Powertrain Engine Domain
AUTOSAR_EXP_AIPowertrainEngine

[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`s for specific purposes.

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

The Standardization Template specifies the approach to overcome this situation.

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

1.1 Document Conventions

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

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

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

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

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

Class	**AUTOSAR**			
Package	M2::AUTOSARTemplates::AutosarTopLevelStructure			
Note	Root element of an AUTOSAR description, also the root element in corresponding XML documents. **Tags:** xml.globalElement=true			
Base	ARObject			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data of an Autosar file. **Tags:** xml.sequenceOffset=10
arPackage	ARPackage	*	aggr	This is the top level package in an AUTOSAR model. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
introductio n	Documentation Block	0..1	aggr	This represents an introduction on the Autosar file. It is intended for example to rpresent disclaimers and legal notes. **Tags:** xml.sequenceOffset=20

Table 1.1: AUTOSAR

The first rows in the table have the following meaning:

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

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

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

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

The headers in the table have the following meaning:

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

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]
[RS_STDT_00004]	Shall allow to represent shortName patterns	[TPS_STDT_00003] [TPS_STDT_00047] [TPS_STDT_00055]
[RS_STDT_00005]	Shall support keywords and keyword abbreviations	[TPS_STDT_00004] [TPS_STDT_00012] [TPS_STDT_00068] [TPS_STDT_00069] [TPS_STDT_00070]
[RS_STDT_00006]	Shall be implemented without compatibility problems to existing template	[TPS_STDT_00033] [TPS_STDT_00041] [TPS_STDT_00047]
[RS_STDT_00007]	Shall be based on the AUTOSAR schema	[TPS_STDT_00033] [TPS_STDT_00041] [TPS_STDT_00047]
[RS_STDT_00008]	Shall provide means to support analyzing the conformity of implementations with the AUTOSAR standards	[TPS_STDT_00001] [TPS_STDT_00003] [TPS_STDT_00012] [TPS_STDT_00042] [TPS_STDT_00048] [TPS_STDT_00052] [TPS_STDT_00054] [TPS_STDT_00059] [TPS_STDT_00060]
[RS_STDT_00009]	Shall be able to represent requirements stated in SWS	[TPS_STDT_00001] [TPS_STDT_00042] [TPS_STDT_00050] [TPS_STDT_00052] [TPS_STDT_00060]
[RS_STDT_00010]	Shall refer to ECUC parameter definition	[TPS_STDT_00025] [TPS_STDT_00040]
[RS_STDT_00011]	Shall be able to standardize components	[TPS_STDT_00024]
[RS_STDT_00012]	Shall be able to standardize architecture	[TPS_STDT_00024]
[RS_STDT_00013]	Shall be able to express parts of reference paths resp. package hierarchies	[TPS_STDT_00013] [TPS_STDT_00051]
[RS_STDT_00014]	Shall be able to express levels of obligation	[TPS_STDT_00028] [TPS_STDT_00053] [TPS_STDT_00067]

Requirement	Description	Satisfied by
[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]

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)
5. AUTOSAR Specifications (SWS, TPS, AI, TR, MOD, EXP etc.)

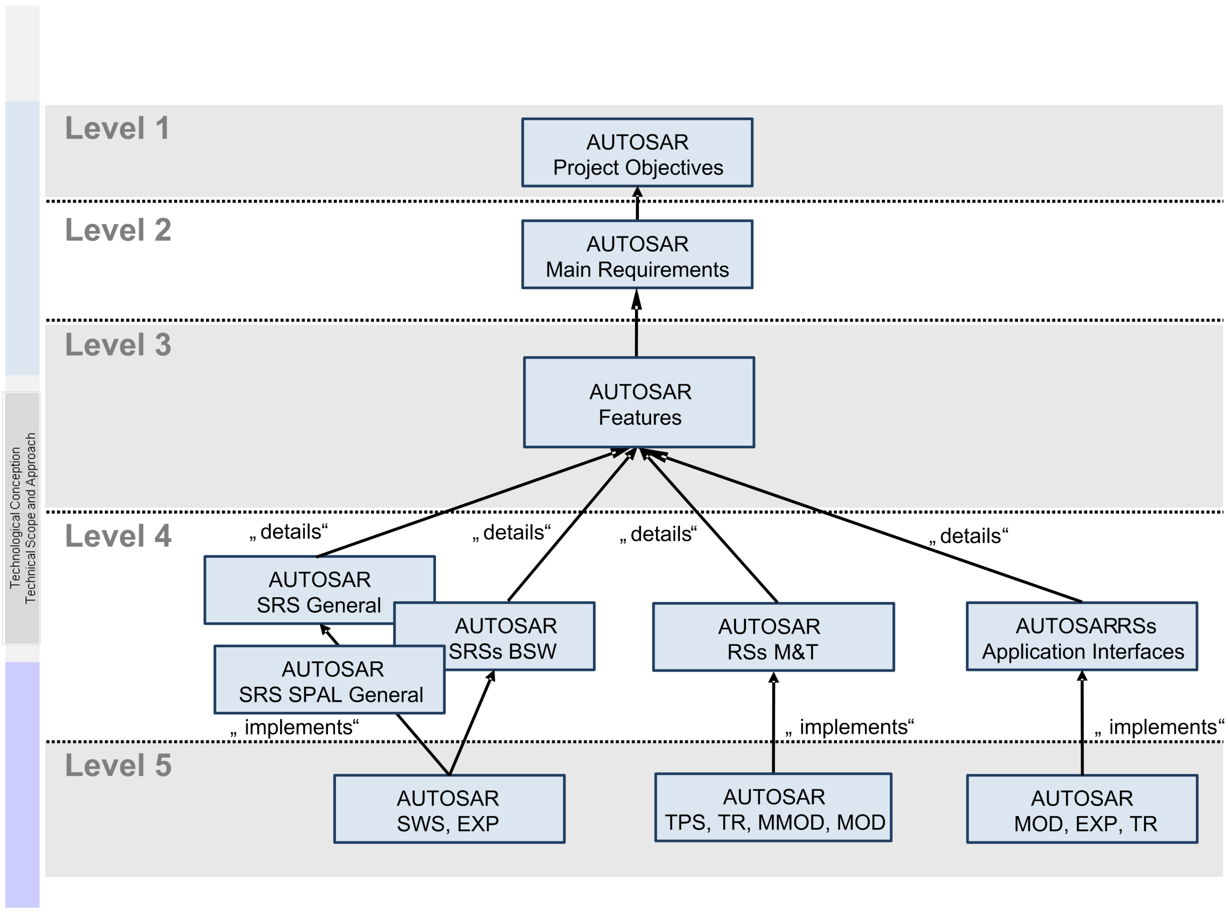


Figure 2.1: Specification levels

[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]. 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.[1].
- After the Id the constraint title (long name) follows.
- The constraint content shall be written inside the opening and closing half bracket.

⌋*(RS_STDT_00038)*

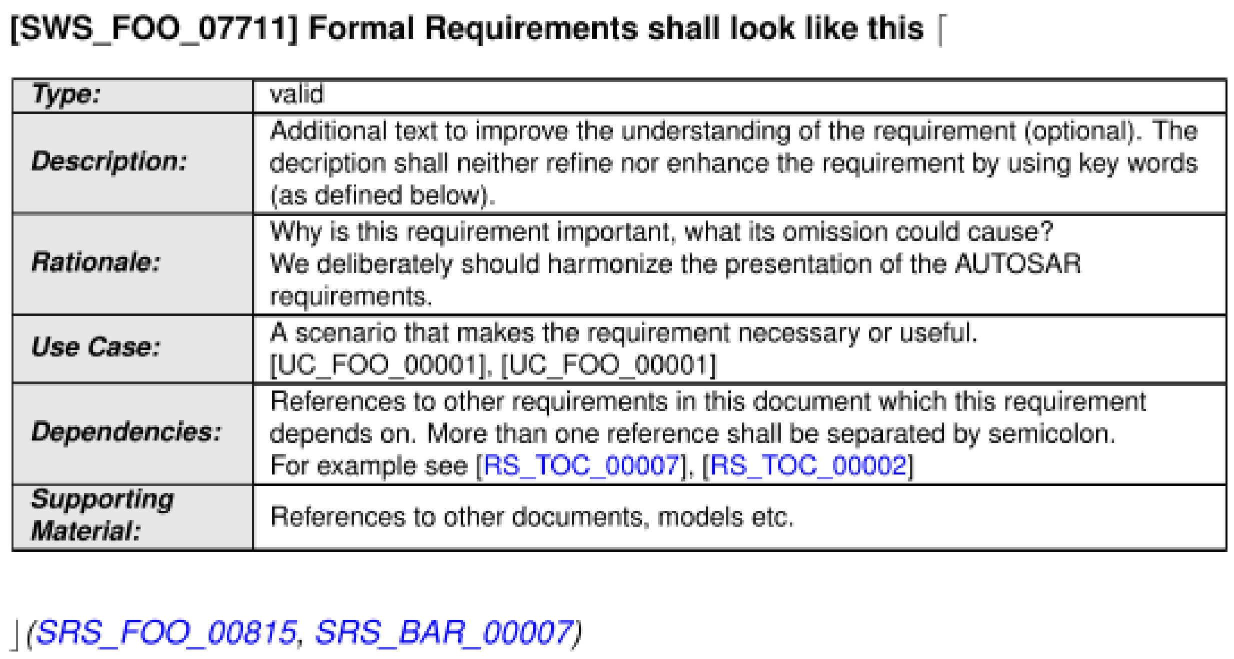
[TPS_STDT_00078] Representation of requirements in AUTOSAR documents ⌈ AUTOSAR requirements are represented using the structure of [TPS_STDT_00060] where the following attributes are presented as a table:

- 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
- 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)*

[1]Please refer to https://svn.autosar.org/repos/work/24_Sources/branches/R4.0/ZAUX_Styles/10_ConstraintNumbers/constraint_numbers.csv

The rendition is illustrated in figure 2.2.



[SWS_FOO_07711] Formal Requirements shall look like this ⌈

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

⌋*(SRS_FOO_00815, SRS_BAR_00007)*

Figure 2.2: 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".

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.

⌋

- **[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 [3] in keyword set InformationCategories, entries with classification TraceCategory.
- module is either module abbreviation in [4] or an entry of the keyword set DocumentAbbreviations with classification DocumentAbbreviation in [3].
- 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 crated which traces back to the requirement items which need additional items in the individual specification.

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

Figure 2.3 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 2.3: Example for trace table using NA and SPEC

[TPS_STDT_00052] Characteristics of `TraceableText` ⌈ `TraceableText` should[2] 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.

[2]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 [5].

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 [3] in keyword set `InformationCategories`, entries with classification `DocumentCategory`.
- `DocumentName` is the `shortName` of the `Keyword` according to [3], keyword set `DocumentAbbreviation` entries with classification `DocumentAbbreviation` or the shortName of the module in [4]

⌋*(RS_STDT_00009)*

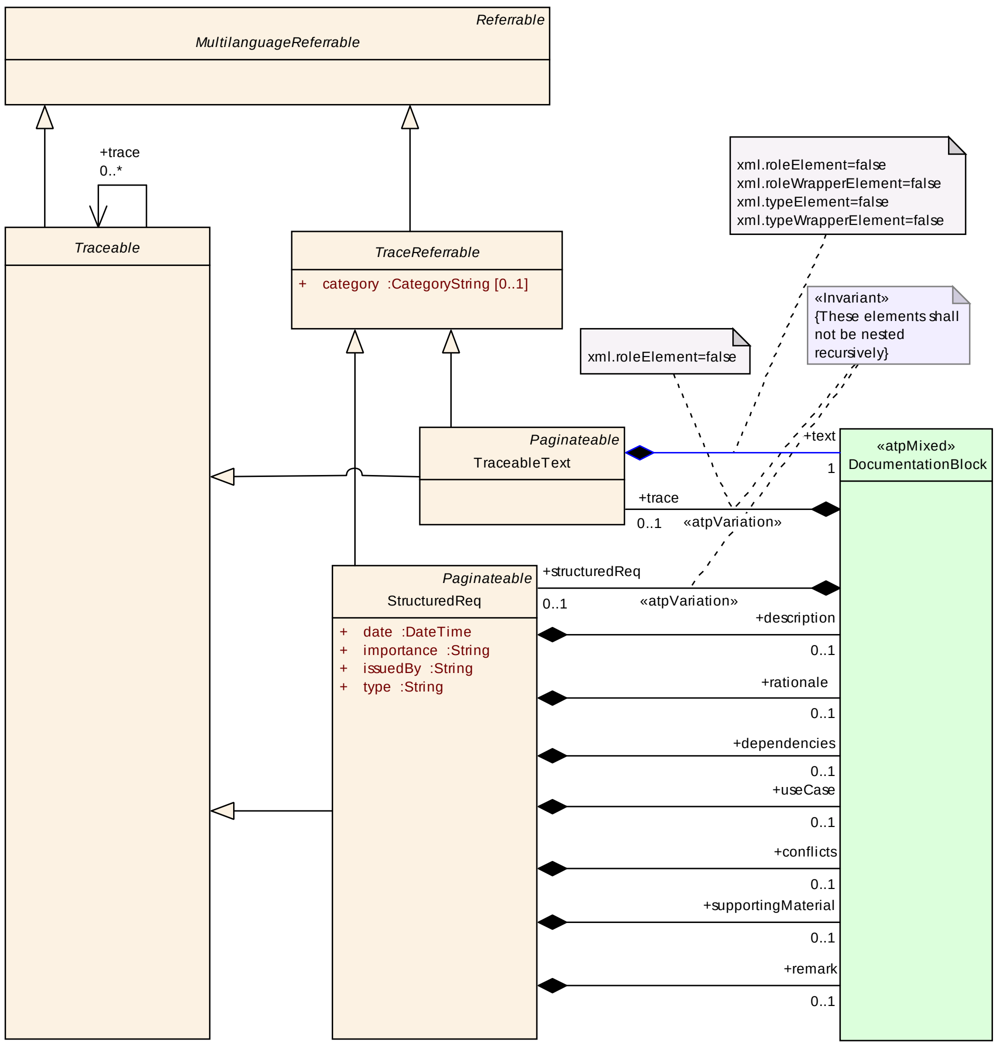


Figure 2.4: Requirements and Tracing

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

Attribute	Datatype	Mul.	Kind	Note

Table 2.1: Traceable

Class	TraceableText			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Documentation::BlockElements::RequirementsTracing			
Note	This meta-class represents the ability to denote a traceable text item such as requirements etc. The following approach appliles: • **shortName** represents the tag for tracing • **longName** represents the head line • **category** represents the kind of the tagged text			
Base	ARObject,DocumentViewSelectable,MultilanguageReferrable,Paginateable,Referrable,TraceReferrable,Traceable			
Attribute	**Datatype**	**Mul.**	**Kind**	**Note**
text	DocumentationBlock	1	aggr	This represents the text to which the tag applies. **Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false

Table 2.2: TraceableText

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

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
description	Documentation Block	0..1	aggr	Ths represents the general description of the requirement. **Tags:** xml.sequenceOffset=10
importance	String	1	attr	This allows to represent the importance of the requirement. **Tags:** xml.sequenceOffset=8
issuedBy	String	1	attr	This represents the person, organization or authority which issued the requirement. **Tags:** xml.sequenceOffset=6
rationale	Documentation Block	0..1	aggr	This represents the rationale of the requirement. **Tags:** xml.sequenceOffset=20
remark	Documentation Block	0..1	aggr	This represents an informal remark. Note that this is not modeled as annotation, since these remark is still essential part of the requirement. **Tags:** xml.sequenceOffset=60
supporting Material	Documentation Block	0..1	aggr	This represents an informal specifiaction of the supporting material. **Tags:** xml.sequenceOffset=50
type	String	1	attr	This attribute allows to denote the type of requirement to denote for example is it an "enhancement", "new feature" etc. **Tags:** xml.sequenceOffset=7
useCase	Documentation Block	0..1	aggr	This describes the relevant use cases. Note that formal references to use cases should be done in the trace relation. **Tags:** xml.sequenceOffset=35

Table 2.3: 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 [6].

[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.

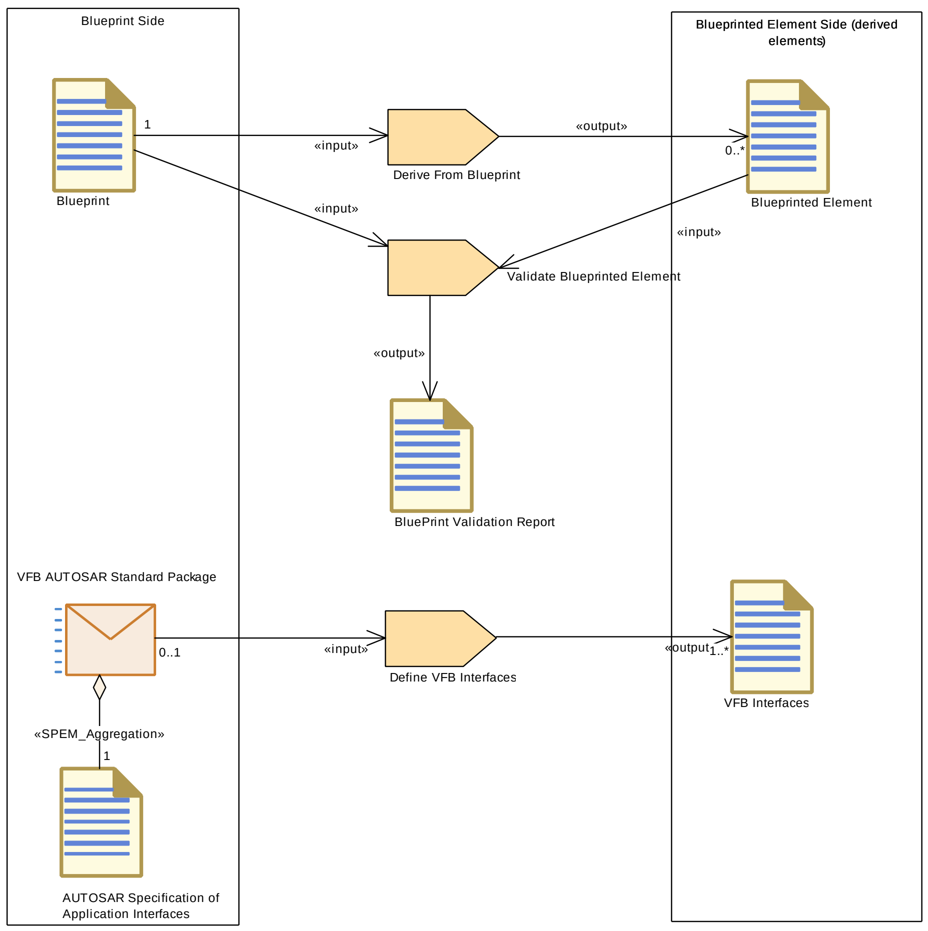


Figure 4.1: Blueprint methdology approach

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 `initValue`s which is not the case for `PortPrototype` which get their initial values from appropriate `ComSpec`s.

- **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.

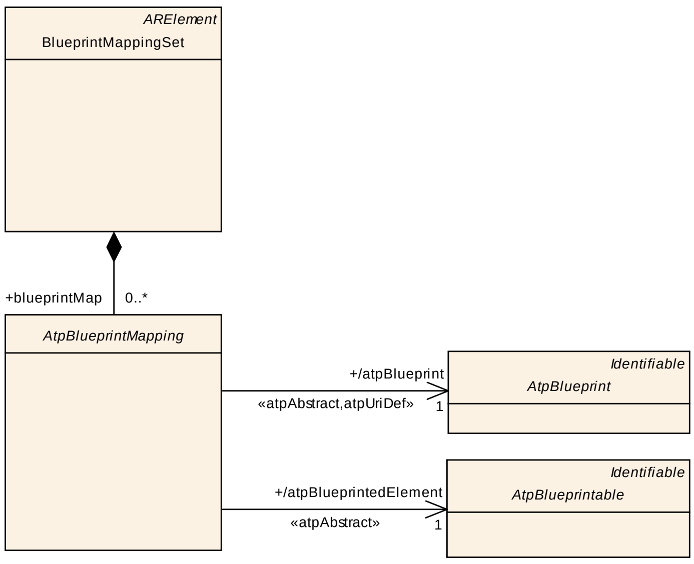


Figure 4.2: Abstract Blueprint Structure

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.

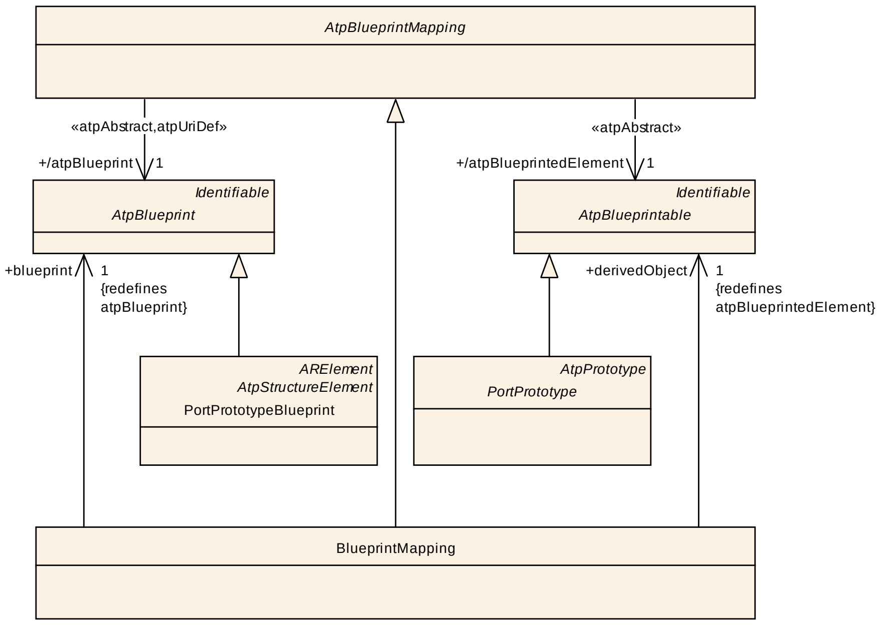


Figure 4.3: Port Blueprints as an example for separate meta-classes for Blueprint and blueprinted Element

[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 (For this reason, such blueprints also may violate the strict AUTOSAR schema).
- 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)*

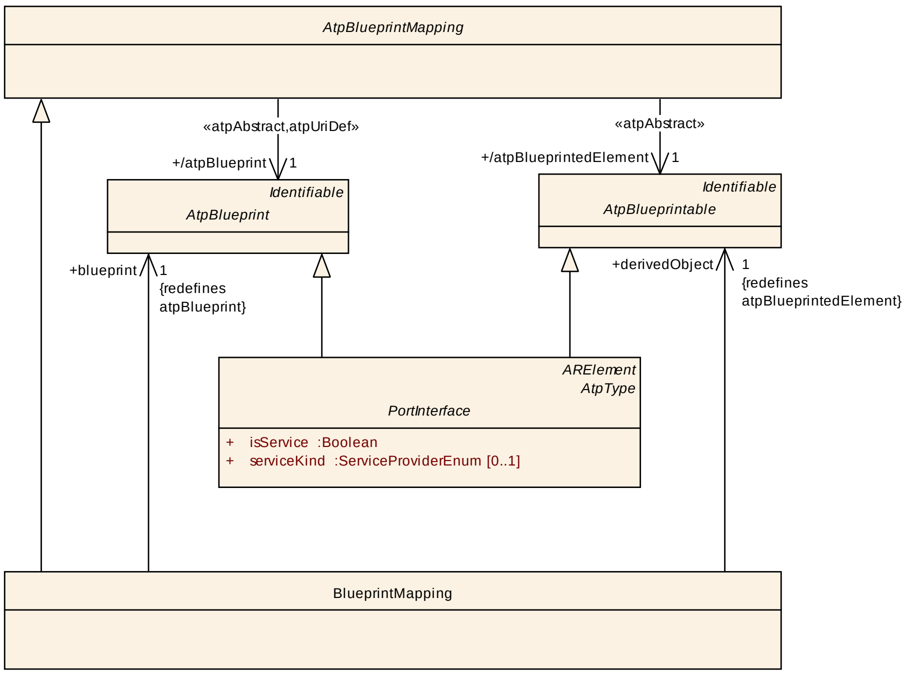


Figure 4.4: PortInterface Blueprints as an example for using the same meta-class for Blueprint and blueprinted Element

[TPS_STDT_00033] Recognize Blueprints ⌈ According to [6] 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`.

Class	**AtpBlueprint (abstract)**			
Package	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
Note	This meta-class represents the ability to act as a Blueprint. As this class is an abstract one, particular blueprint meta-classes inherit from this one.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***

Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
shortName Pattern	String	0..1	attr	This attribute represents the pattern which shall be used to build the shortName of the derived elements. As of now it is modeled as a String. In general it should follow the pattern: `pattern = (placeholder | namePart)*` `placeholder = "{" namePart "}"` `namePart = identifier | "_"` This is subject to be refined in subsequent versions. Note that this is marked as obsolete. Use the xml attribute namePattern instead as it applies to Identifier and CIdentifier (shortName, symbol etc.) **Tags:** atp.Status=obsolete

Table 4.1: AtpBlueprint

Class	**AtpBlueprintable (abstract)**			
Package	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
Note	This meta-class represents the ability to be derived from a Blueprint. As this class is an abstract one, particular blueprintable meta-classes inherit from this one.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable			
Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table 4.2: AtpBlueprintable

Class	**AtpBlueprintMapping (abstract)**			
Package	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
Note	This meta-class represents the ability to express a particular mapping between a blueprint and an element derived from this blueprint. Particular mappings are defined by specializations of this meta-class.			
Base	ARObject			
Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
atpBlueprint	AtpBlueprint	1	ref	This represents the blueprint. **Stereotypes:** atpAbstract; atpUriDef **Tags:** xml.sequenceOffset=50
atpBlueprintedElement	AtpBlueprintable	1	ref	This represents the bluprinted elements which shall be mapped to the blueprint. **Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=60

Table 4.3: AtpBlueprintMapping

4.2 Mapping of Blueprints to blueprinted Elements

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

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

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

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

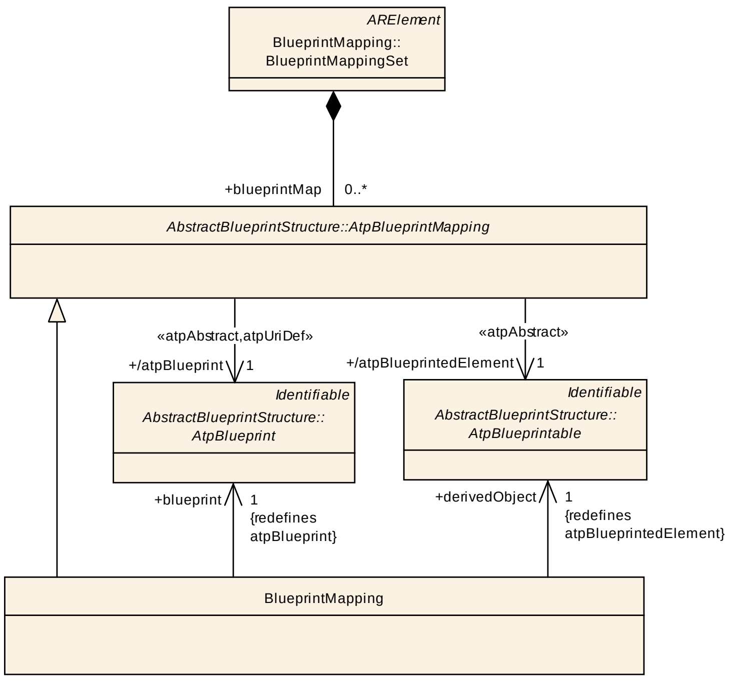


Figure 4.5: Mapping of Derived Objects and their Blueprints

[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. ⌋

[1]For compatibility reasons, the abstract patten was not changed. The previous specializations (PortInterfaceBlueprintMapping and PortPrototypeBlueprintMapping are obsolete, but kept in the schema.

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

Table 4.4: 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 `AtpBlueprintMapping`s. Due to ≪`atpUriDef`≫, the references from `AtpBlueprintMapping` do not need to be resolved in system descriptions. ⌋

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

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

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

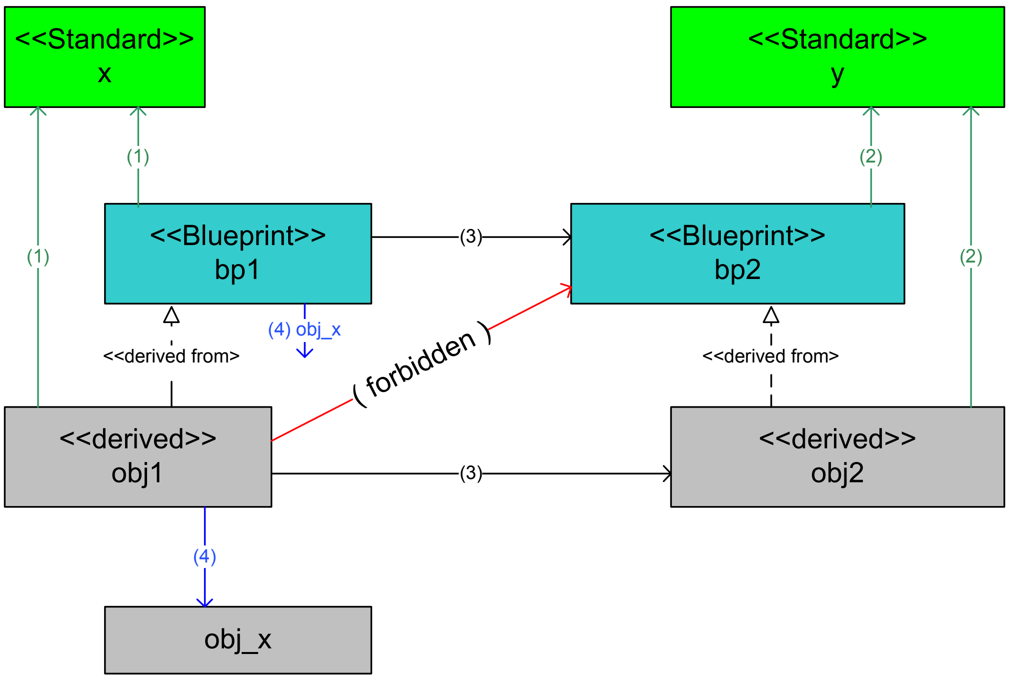


Figure 4.6: Relations between Blueprints, "Derived Objects" and "Standardized Objects"

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

[TPS_STDT_00051] Handling references when deriving objects from blueprints ⌈

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

⌋*(RS_STDT_00013, RS_STDT_00017)*

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

[constr_2546] References 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 [6] for the following aspects:

- Even if BindingTimeEnum does not contain the value blueprintDerivationTime, there are sill 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 [6] 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.

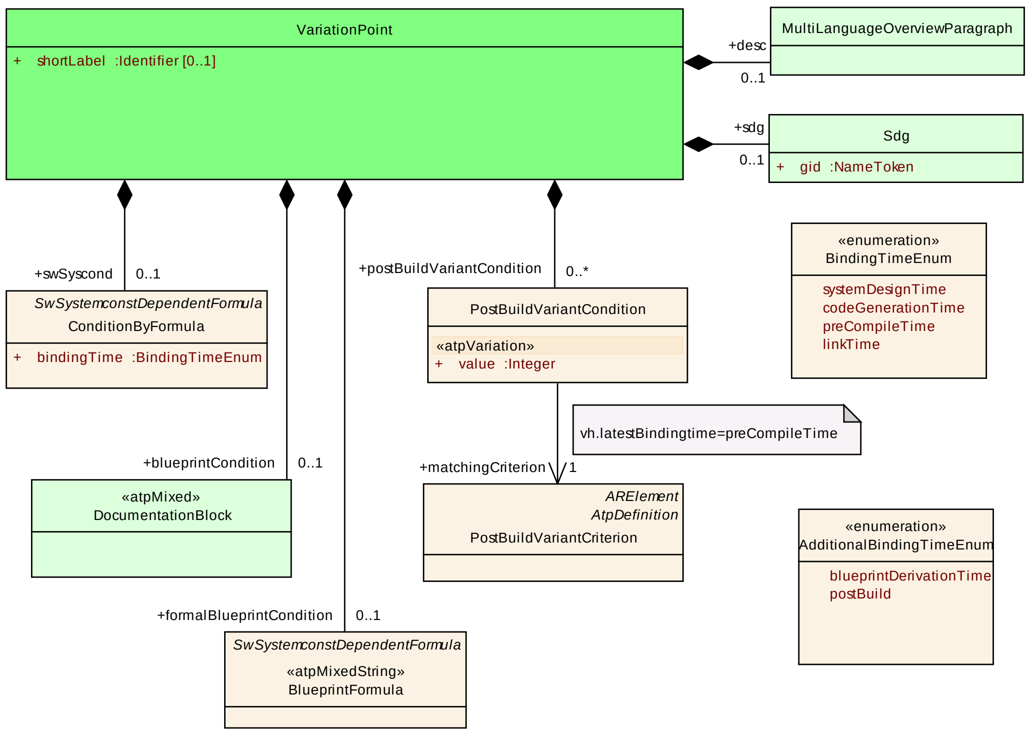


Figure 4.7: Variation Point

Class	**VariationPoint**			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
Note	This meta-class represents the ability to express a "structural variation point". The container of the variation point is part of the selected variant if swSyscond evaluates to true and each postBuildVariantCriterion is fulfilled.			
Base	ARObject			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
desc	MultiLanguage OverviewParagr aph	0..1	aggr	This allows to describe shortly the purpose of the variation point. **Tags:** xml.sequenceOffset=20
blueprintC ondition	Documentation Block	0..1	aggr	This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint. Note that variationPoints are not allowed within a blueprintCondition. **Tags:** xml.sequenceOffset=28
formalBlue printCondit ion	BlueprintFormul a	0..1	aggr	This denotes a formal blueprintCondition. This shall be not in contradiction with blueprintCondition. It is recommanded only to use one of the two. **Tags:** xml.sequenceOffset=29

Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
postBuildVariantCondition	PostBuildVariantCondition	*	aggr	This is the set of post build variant conditions which all shall be fulfilled in order to (postbuild) bind the variation point. **Tags:** xml.sequenceOffset=40
sdg	Sdg	0..1	aggr	An optional special data group is attached to every variation point. These data can be used by external software systems to attach application specific data. For example, a variant management system might add an identifier, an URL or a specific classifier. **Tags:** xml.sequenceOffset=50
shortLabel	Identifier	0..1	ref	This provides a name to the particular variation point to support the RTE generator. It is necessary for supporting splitable aggregations and if binding time is later than codeGenerationTime, as well as some RTE conditions. It needs to be unique with in the enclosing Identifiables with the same ShortName. **Tags:** xml.sequenceOffset=10
swSyscond	ConditionByFormula	0..1	aggr	This condition acts as Binding Function for the VariationPoint. Note that the mulitplicity is 0..1 in order to support pure postBuild variants. **Tags:** xml.sequenceOffset=30

Table 4.5: 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, `VariationPoint`s 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 Bluprint 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 [6], 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
```

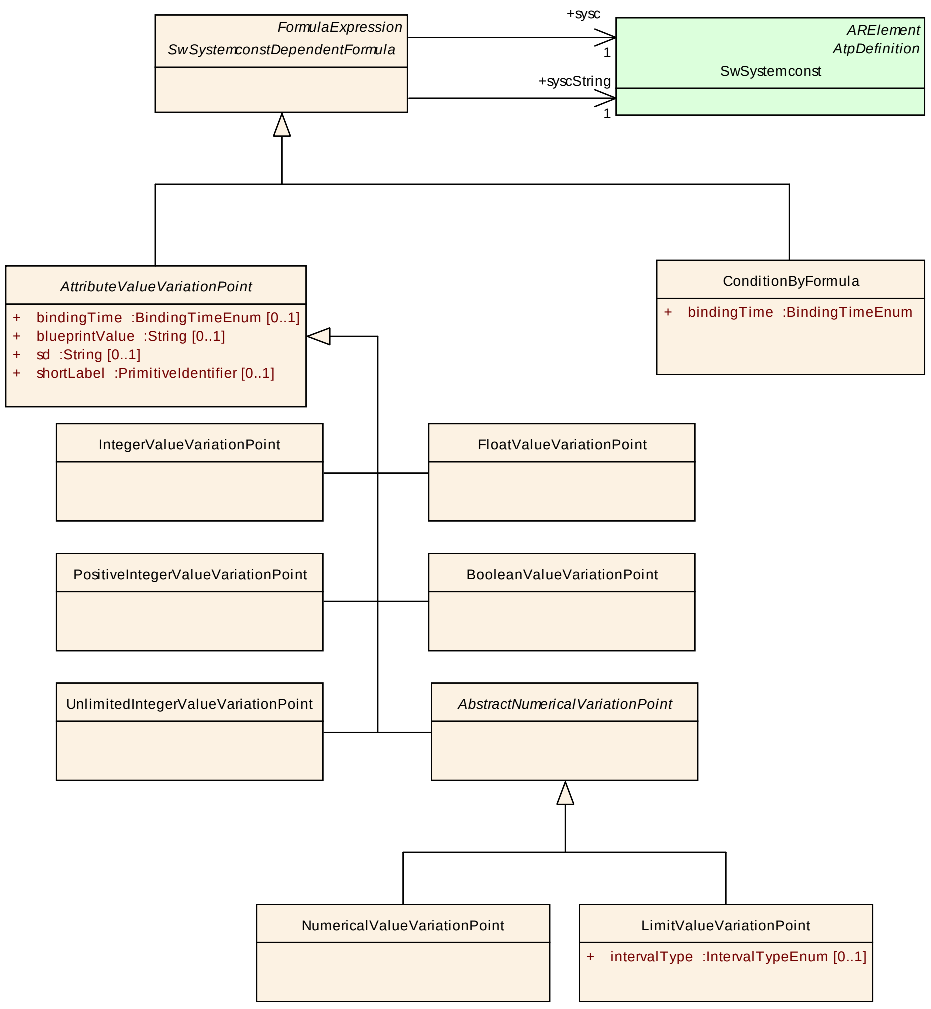

FormulaExpression
SwSystemconstDependentFormula
+sysc
1
+syscString
1
ARElement
AtpDefinition
SwSystemconst
AttributeValueVariationPoint
+ bindingTime :BindingTimeEnum [0..1]
+ blueprintValue :String [0..1]
+ sd :String [0..1]
+ shortLabel :PrimitiveIdentifier [0..1]
ConditionByFormula
+ bindingTime :BindingTimeEnum
IntegerValueVariationPoint
FloatValueVariationPoint
PositiveIntegerValueVariationPoint
BooleanValueVariationPoint
UnlimitedIntegerValueVariationPoint
AbstractNumericalVariationPoint
NumericalValueVariationPoint
LimitValueVariationPoint
+ intervalType :IntervalTypeEnum [0..1]

Figure 4.8: Attribute Value Variation Point

Class	≪atpMixedString≫ **AttributeValueVariationPoint (abstract)**			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling::AttributeValueVariationPoints			
Note	This class represents the ability to derive the value of the Attribute from a system constant (by SwSystemconstDependentFormula). It also provides a bindingTime.			
Base	ARObject,FormulaExpression,SwSystemconstDependentFormula			
Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
bindingTime	BindingTimeEnum	0..1	attr	This is the binding time in which the attribute value needs to be bound. If this attribute is missing, the attribute is not a variation point. In particular this means that It needs to be a single value according to the type specified in the pure model. It is an error if it is still a formula. **Tags:** xml.attribute=true

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint. **Tags:** xml.attribute=true
sd	String	0..1	attr	This special data is provided to allow synchronization of Attribute value variation points with variant management systems. The usage is subject of agreement between the involved parties. **Tags:** xml.attribute=true
shortLabel	PrimitiveIdentifier	0..1	attr	This allows to identify the variation point. It is also intended to allow RTE support for CompileTime Variation points. **Tags:** xml.attribute=true

Table 4.6: 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)*

Primitive	**Identifier**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
Note	An Identifier is a string with a number of constraints on its appearance, satisfying the requirements typical programming languages define for their Identifiers. This datatype represents a string, that can be used as a c-Identifier. It needs to start with a letter, may consist of letters, digits and underscore. It shall not have two consecutive underscores (to support subsequent name mangling based on "__"). **Tags:** xml.xsd.customType=IDENTIFIER; xml.xsd.maxLength=128; xml.xsd.pattern=[a-zA-Z]([a-zA-Z0-9]|_[a-zA-Z0-9])*_?; xml.xsd.type=string			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***

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

Table 4.7: Identifier

Primitive	**CIdentifier**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Primitive Types			
Note	This datatype represents a string, that follows the rules of C-identifiers. **Tags:** xml.xsd.customType=C-IDENTIFIER; xml.xsd.pattern=[a-zA-Z_][a-zA-Z0-9_]*; xml.xsd.type=string			
Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
namePatte rn	String	0..1	attr	This attribute represents a pattern which shall be used to define the value of the identifier if the CIdentifier in question is part of a blueprint. For more details refer to TPS_StandardizationTemplate. **Tags:** xml.attribute=true

Table 4.8: CIdentifier

[TPS_STDT_00055] General Syntax for Name Patterns ⌈ The name pattern uses the following syntax defined according to ANTLR [7].

Listing 4.1: 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]|_[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 `shortName`s.

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`.

componentPrototypeTypeName This represents the `shortName` of the dedicated `SwComponentType`.

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)*

Mip This represents the module implementation prefix according to [SWS_BSW_00102].

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

portDir This represents the direction of a port.

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

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

4.5 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 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

⌋*(RS_STDT_00019)*

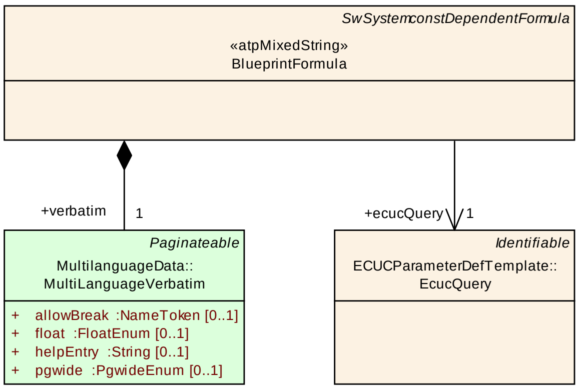

SwSystemconstDependentFormula
«atpMixedString»
BlueprintFormula
+verbatim
1
+ecucQuery
1
Paginateable
MultilanguageData::
MultiLanguageVerbatim
+ allowBreak :NameToken [0..1]
+ float :FloatEnum [0..1]
+ helpEntry :String [0..1]
+ pgwide :PgwideEnum [0..1]
Identifiable
ECUCParameterDefTemplate::
EcucQuery

Figure 4.9: Blueprint Formula

Listing 4.2 illustrates valid expression patterns. Note that `blueprintValue`, `blueprintCondition` etc. denotes a placeholder.

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

Listing 4.2: Blueprint Formula taken from AUTOSAR_MOD_BSWServiceInterfaces_Blueprint.arxml

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

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 [8] 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 [6]. ⌋*(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.

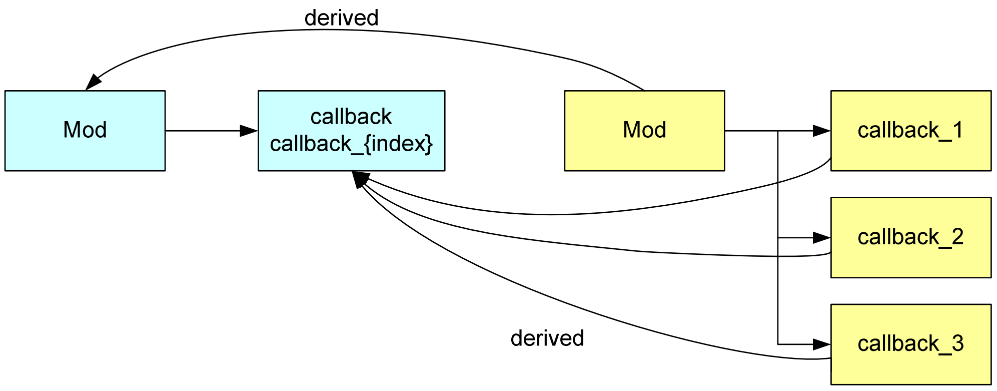


Figure 5.1: Multiply derived Objects

[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 variationPoint at CompuScale. ⌋*(RS_STDT_00002, RS_STDT_00029)*

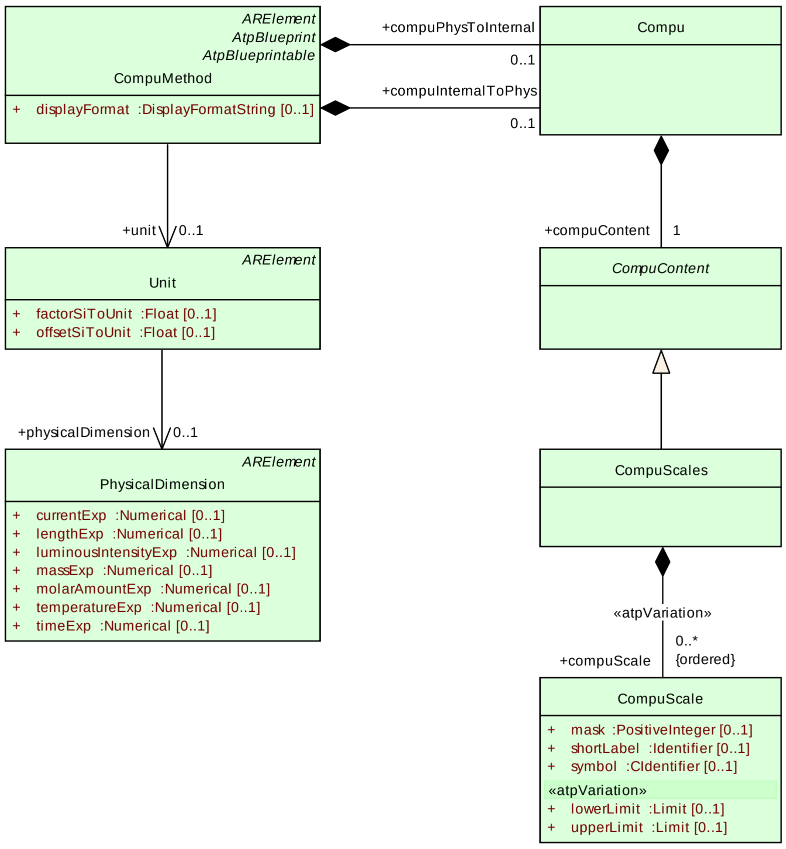

ARElement
AtpBlueprint
AtpBlueprintable
CompuMethod
+ displayFormat :DisplayFormatString [0..1]
+compuPhysToInternal
0..1
+compuInternalToPhys
0..1
Compu
+unit
0..1
ARElement
Unit
+ factorSiToUnit :Float [0..1]
+ offsetSiToUnit :Float [0..1]
+compuContent
1
CompuContent
+physicalDimension
0..1
ARElement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

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)*

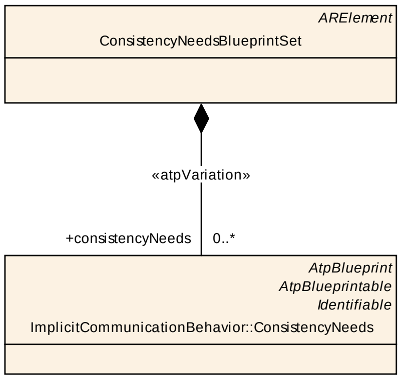


Figure 5.3: Blueprinting ConsistencyNeeds

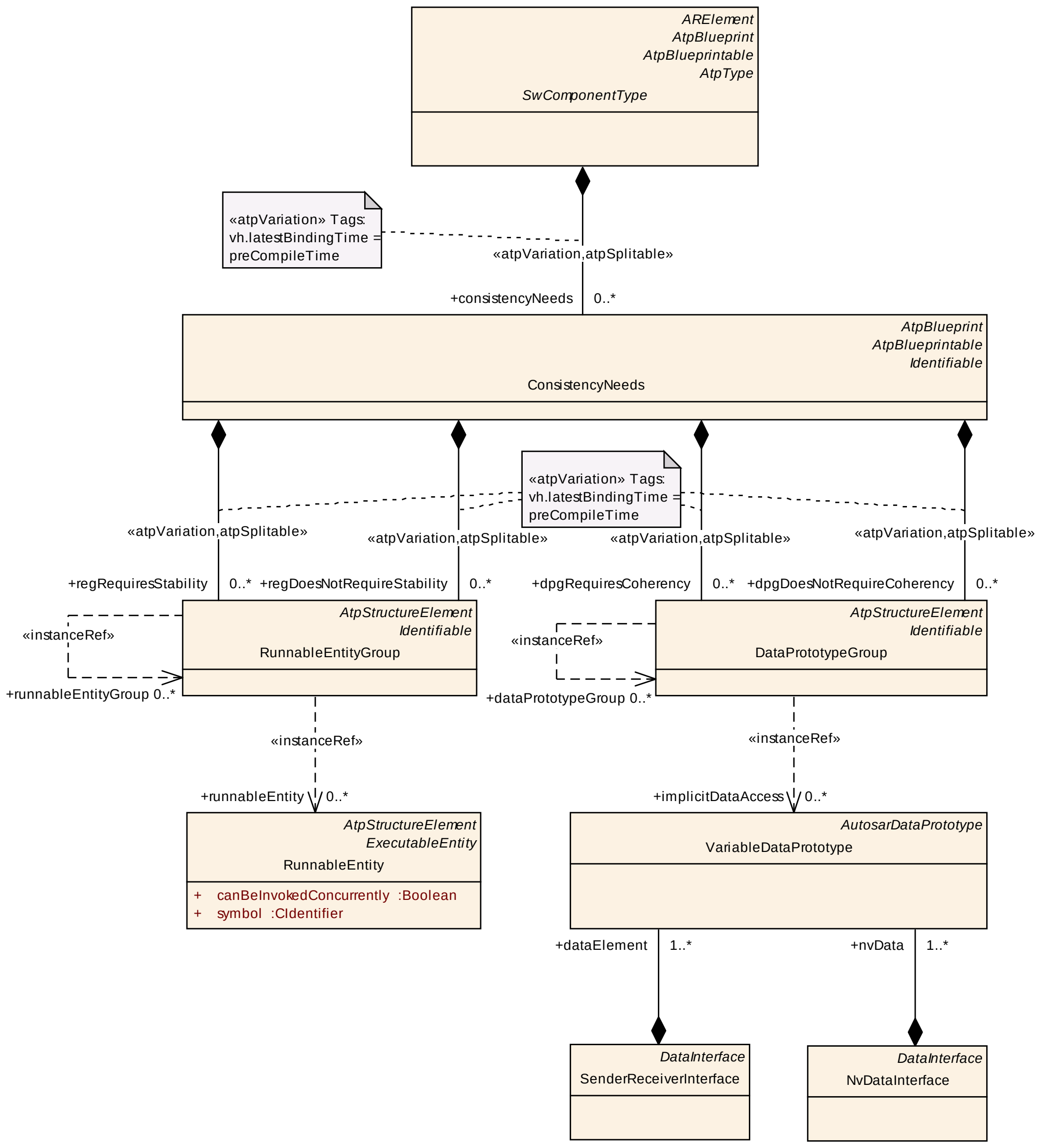


Figure 5.4: ConsistencyNeeds

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

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

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

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

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

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

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

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

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

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

EXPLICIT_DATA_PROTOTYPE_GROUP DataPrototypeGroup of the ConsistencyNeeds shall contain VariableDataPrototypes according functional requirements

⌋*(RS_STDT_00034)*

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

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

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

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

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

EXPLICIT_RUNNABLE_ENTITY_GROUP RunnableEntityGroup of the ConsistencyNeeds shall contain RunnableEntitys according functional requirements

⌋(*RS_STDT_00034*)

5.10 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 [9].

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 Keywordshall 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)*

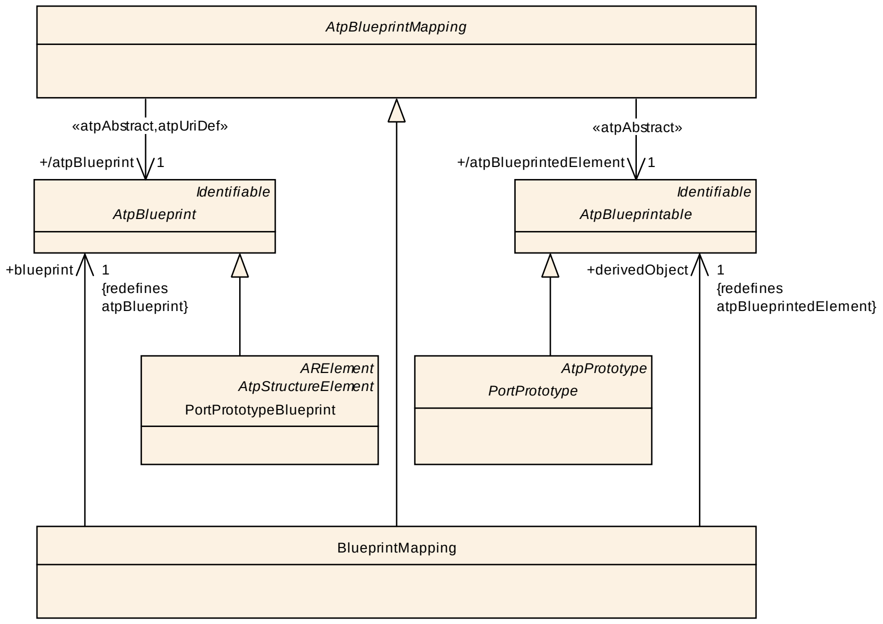


Figure 5.5: Mapping of Port Prototype Blueprints

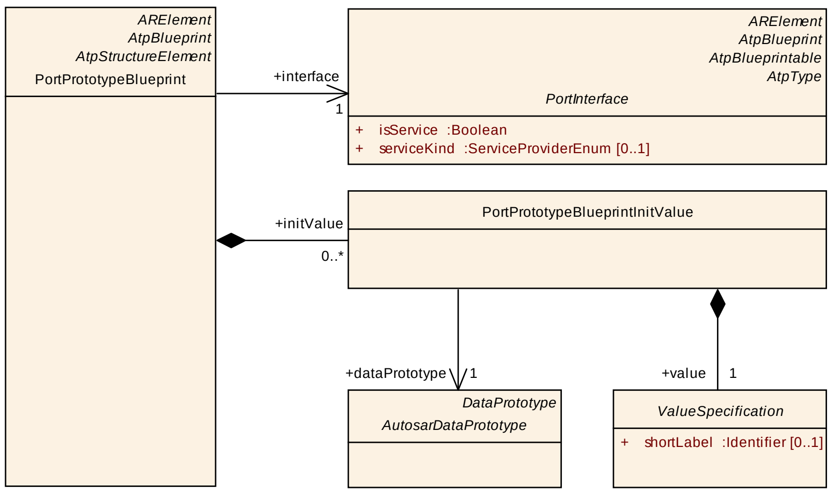


Figure 5.6: Blueprinting Port Prototype

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[1] is part of the definition of a standardized "application interface". Therefore, PortPrototypeBlueprint also supports the definition of an initValue, which needs to be moved to the appropriate ComSpecs.
- It has a reference to the corresponding PortInterface. If the referenced PortInterface is not a blueprint, it can directly be taken over by the PortPrototype created out of the PortPrototypeBlueprint such that the new PortPrototype references the PortInterface. If the referenced PortInterface 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.

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

- 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]).

Class	**PortPrototypeBlueprint**			
Package	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintDedicated::Port ProtoypeBlueprint			
Note	This meta-class represents the ability to express a blueprint of a PortPrototype by referring to a particular PortInterface. This blueprint can then be used as a guidance to create particular PortPrototypes which are defined according to this blueprint. By this it is possible to standardize application interfaces without the need to also standardize software-components with PortPrototypes typed by the standardized PortInterfaces. **Tags:** atp.recommendedPackage=PortPrototypeBlueprints			
Base	ARElement,ARObject,AtpBlueprint,AtpClassifier,AtpFeature,AtpStructure Element,CollectableElement,Identifiable,MultilanguageReferrable,Packageable Element,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
initValue	PortPrototypeBlueprintInitValue	*	aggr	This specifies the init values for the dataElements in the particular PortPrototypeBlueprint.
interface	PortInterface	1	ref	This is the interface for which the blueprint is defined. It may be a blueprint itself or a standardized PortInterface

Table 5.1: PortPrototypeBlueprint

Class	**PortPrototypeBlueprintInitValue**			
Package	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintDedicated::Port ProtoypeBlueprint			
Note	This meta-class represents the ability to express init values in PortPrototypeBlueprints. These init values act as a kind of blueprint from which for example proper ComSpecs can be derived.			
Base	ARObject			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
dataPrototype	AutosarDataPrototype	1	ref	This is the data prototype for which the init value applies **Tags:** xml.sequenceOffset=30
value	ValueSpecification	1	aggr	This is the init value for the particular data prototype. **Tags:** 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 [6], model artifacts (in this case PortPrototypeBlueprint and incompletely specified PortInterfaces) created for the purpose of becoming blueprints shall reside in an ARPackage of category BLUEPRINT. ⌋

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

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

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

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

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

5.20 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 Sender-

ReceiverInterfaces). In other words both interfaces shall be instances of the same meta class. ⌋

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

5.21 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) (*SMMap* and *RMMap*). From this, it is possible to

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

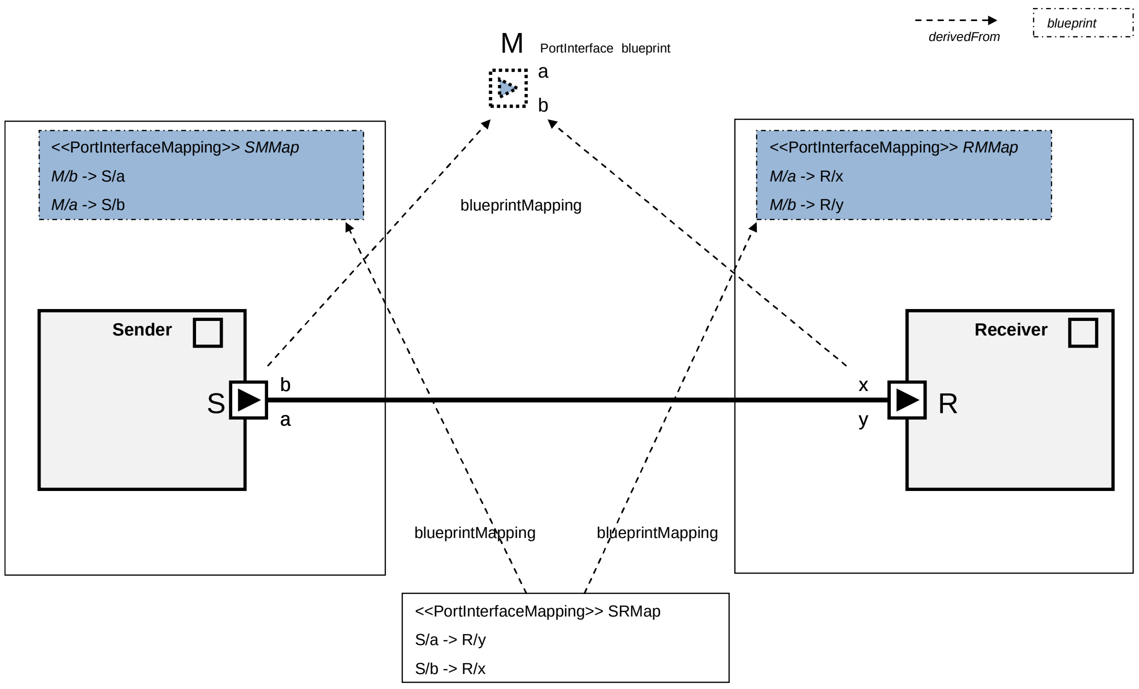


Figure 5.7: Deriving PortInterfaceMapping (1)

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

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

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

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

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

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

   Implicit mappings (i.e. if data element names between PortInterface and PortInterface(blueprint) are identical then no PortInterfaceMapping(blueprint) is needed) have to be considered too (for example by creating "temporary" mappings).
4. Create BlueprintMappings for the created PortInterfaceMapping (*SRMap*) in step 3 to the involved PortInterfaceMapping(blueprints) (*SMMap* and *RMMap*).

The scenario is shown in the now following listings:

- Listing 5.1 shows the definitons eg. given by AUTOSAR.
- Listing 5.2 shows the part of LeftCompany
- Listing 5.3 shows the part of RightCompany
- Listing 5.4 shows the part of the integration in a Project

Listing 5.1: 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 5.2 shows that "LeftCompany" has created the `PortInterface` named *S* derived from the `PortInterface`(Blueprint) *M*. Thereby the description **how** this takes place is given in the blueprint of an appropriate `PortInterfaceMapping` named *SMMap*.

Listing 5.2: Scenario for Blueprints of PortInterfaceMapping (2)

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

Listing 5.3 shows that "RightCompany" has crated the `PortInterface` named *R* derived from the `PortInterface`(Blueprint) *M*. Thereby the description **how** this takes place is given in the blueprint of an appropriate `PortInterfaceMapping` named *RMMap*.

Listing 5.3: 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}">MRMap</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 5.4 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 (*SMMap* and *SRMap*).

Listing 5.4: Scenario for Blueprints of PortInterfaceMapping (4)

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

5.22 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] `SwComponentType`s shall be of same kind ⌈ Both objects (`SwComponentType`s) referenced by a blueprint mapping for port interfaces (represented by `BlueprintMapping`) shall be of the same kind (e.g. both shall be `AtomicSwCom-`

ponentTypes). 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 [10]. 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.

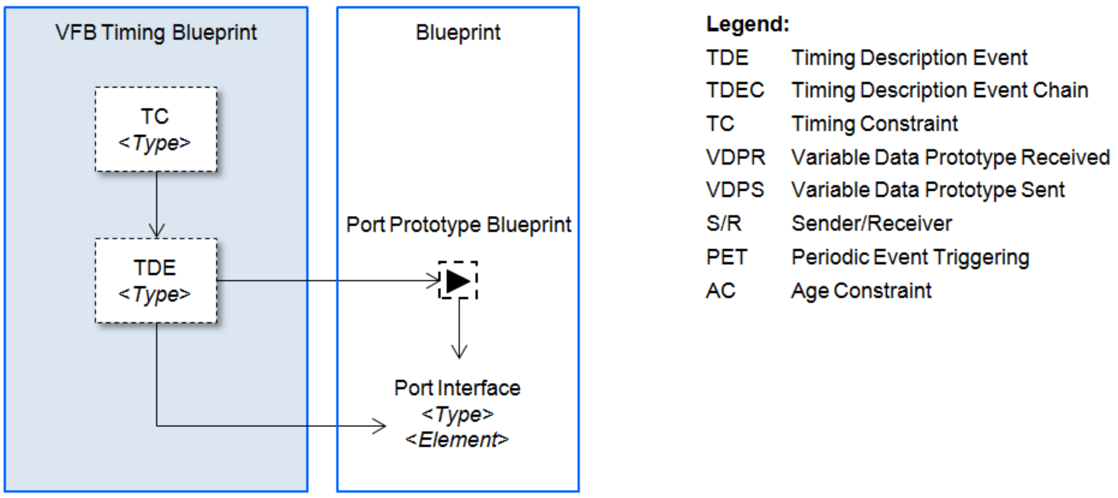

VFB Timing Blueprint
TC
<Type>
TDE
<Type>
Blueprint
Port Prototype Blueprint
Port Interface
<Type>
<Element>
Legend:
TDE Timing Description Event
TDEC Timing Description Event Chain
TC Timing Constraint
VDPR Variable Data Prototype Received
VDPS Variable Data Prototype Sent
S/R Sender/Receiver
PET Periodic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

Figure 5.8: 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” [11].

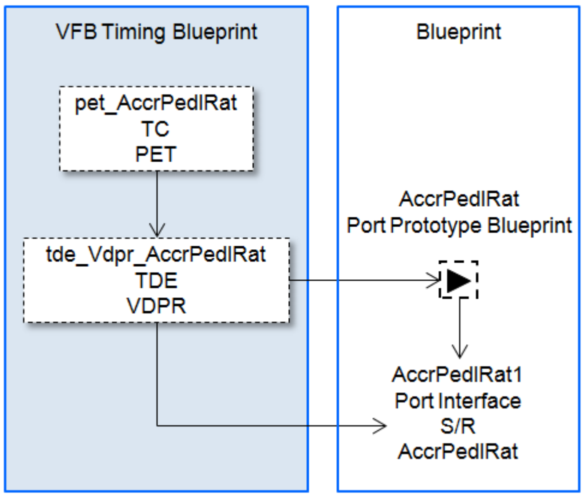
A VFB Timing references the software component it is associated with. In case of a VFB Timing Blueprint this reference need not to be set, but in the derived VFB Timing the `VfbTiming.component` shall be set properly. In addition, any reference to `PortPrototypeBlueprint` shall be replaced by the corresponding 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" [12].



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

Figure 5.9: VFB Timing Blueprint 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 5.5 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 receiption of the specific variable data prototype.

Listing 5.5: 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-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”.

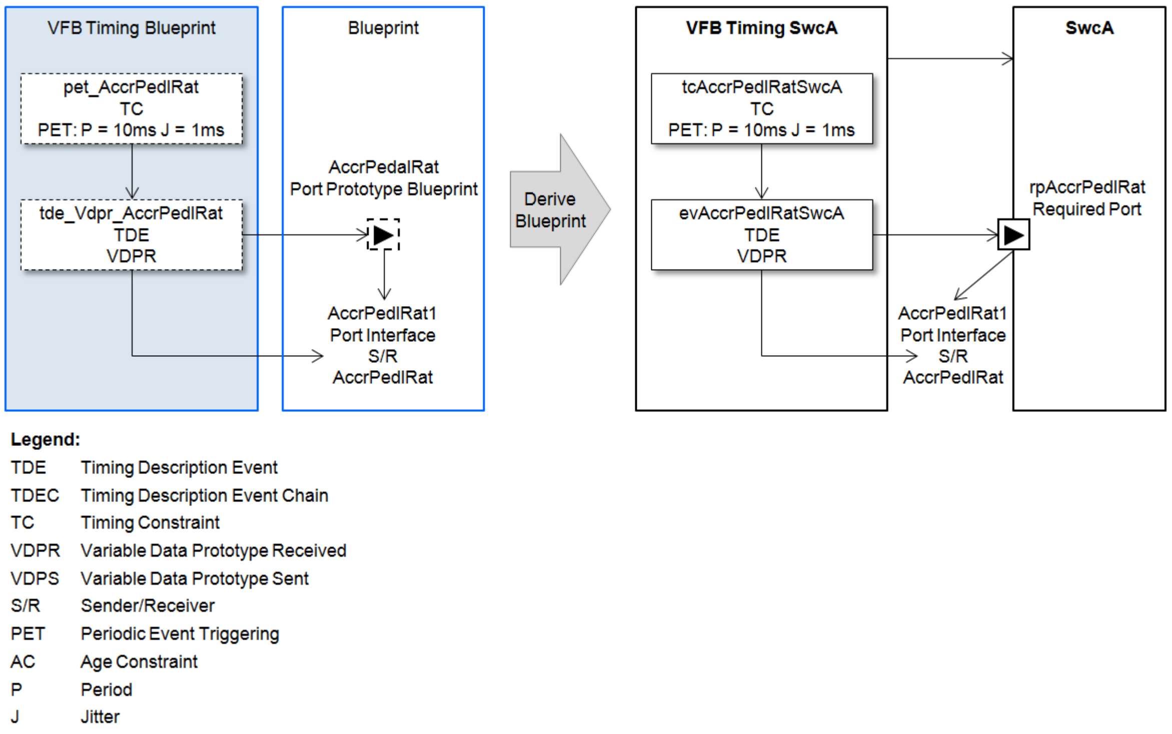


Figure 5.10: Deriving a VFB Timing Blueprint

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)*

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".

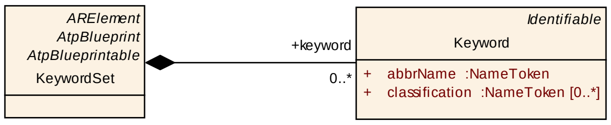


Figure 6.1: Keyword and KeywordSet

[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)*

Class	**KeywordSet**			
Package	M2::AUTOSARTemplates::StandardizationTemplate::Keyword			
Note	This meta–class represents the ability to collect a set of predefined keywords. **Tags:** atp.recommendedPackage=KeywordSets			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,Collectable Element,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***

Attribute	Datatype	Mul.	Kind	Note
keyword	Keyword	*	aggr	This is one particular keyword in the keyword set.

Table 6.1: KeywordSet

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

Table 6.2: Keyword

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

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 [3].

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-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>AISpecification</SHORT-NAME>
          <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-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)*

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

Listing 6.2: Example for Stem Relation of Keywords

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

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

```
    </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 [10] 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.

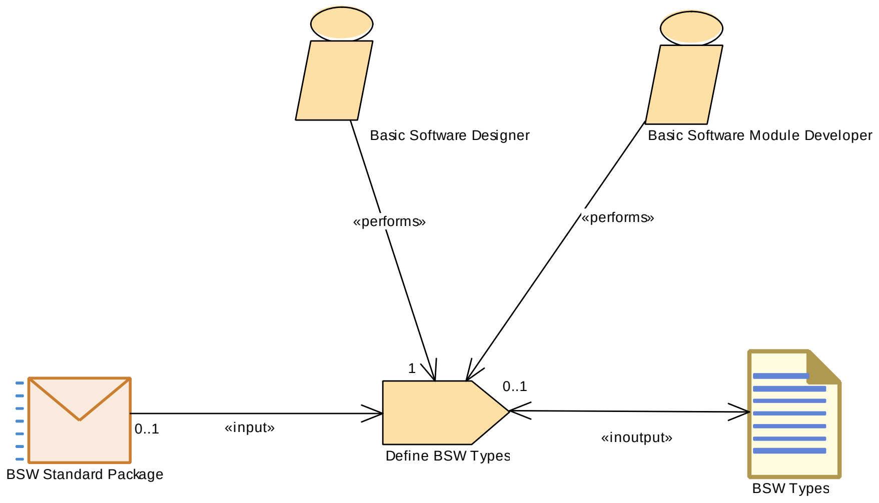


Figure 7.1: Define Bsw Types

[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 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 ≪`atpSplitable`≫. One partial model is represented in an AUTOSAR XML description by one file. The partial model does not need to fulfill all semantic constraints applicable to an AUTOSAR model.

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

AUTOSAR 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 Bluprint 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 TraceableText
[TPS_STDT_00053]	Expression of obligation
[TPS_STDT_00054]	Organisation of TraceableText
[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]	TraceableText
[TPS_STDT_00060]	StructuredReq
[TPS_STDT_00062]	Blueprinting Elements of AccessControl
[TPS_STDT_00063]	Blueprinting BuildActionManifest
[TPS_STDT_00064]	Applied Life Cycle Information Sets on AUTOSAR provided Models (M1)
[TPS_STDT_00065]	Nested Blueprint Can be Used as Blueprint of its own
[TPS_STDT_00066]	Blueprinting PortInterface
[TPS_STDT_00067]	Standardized Path for Standardized Elements
[TPS_STDT_00068]	Expressing "stem"-Relation of Keywords
[TPS_STDT_00069]	Attributes of Keyword
[TPS_STDT_00070]	Classification of Keywords
[TPS_STDT_00071]	Blueprinting ConsistencyNeeds
[TPS_STDT_00072]	Same Meta Class For Blueprints and Derived Objects
[TPS_STDT_00073]	Early definition of ConsistencyNeeds
[TPS_STDT_00074]	Categorization of Blueprints of ConsistencyNeeds
[TPS_STDT_00075]	Categories for DataPrototypeGroup in a Blueprint of ConsistencyNeeds
[TPS_STDT_00076]	Categories for RunnableEntityGroup in a Blueprint of ConsistencyNeeds
[TPS_STDT_00077]	Blueprinting KeywordSet
[TPS_STDT_00078]	Representation of requirements in AUTOSAR documents

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

Table 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 TDEventVfbPort shall reference PortPrototypeBlueprint

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 SwAddrMethod

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

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.

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

Table C.1: ARElement

Class	**ARPackage**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	AUTOSAR package, allowing to create top level packages to structure the contained ARElements. ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package. This is an extended version of MSR's SW-SYSTEM.			
Base	ARObject,AtpBlueprint,AtpBlueprintable,Collectable Element,Identifiable,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
arPackage	ARPackage	*	aggr	This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableEle ment	*	aggr	Elements that are part of this package **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
referenceB ase	ReferenceBase	*	aggr	This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references. **Stereotypes:** atpSplitable **Tags:** atp.splitkey=shortLabel xml.sequenceOffset=10

Table C.2: ARPackage

Class	**AclObjectSet**			
Package	M2::AUTOSARTemplates::GenericStructure::RolesAndRights			
Note	This meta class represents the ability to denote a set of objects for which roles and rights (access control lists) shall be defined. It basically can define the objects based on • the nature of objects • the involved blueprints • the artifact in which the objects are serialized • the definition of the object (in a definition - value pattern) • individual reference objects **Tags:** atp.recommendedPackage=AclObjectSets			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,Collectable Element,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
aclObjectC lass	ReferrableSubty pesEnum	*	attr	This specifies that the considered objects as instances of the denoted meta class.
aclScope	AclScopeEnum	1	attr	this indicates the scope of the referenced objects.
collection	Collection	0..1	ref	This indicates that the relevant objects are specified via a collection.
derivedFro mBlueprint	AtpBlueprint	*	ref	This association indicates that the considered objects are the ones being derived from the associated blueprint. **Stereotypes:** atpUriDef
engineerin gObject	AutosarEnginee ringObject	*	aggr	This indicates an engineering object. The AclPermission relates to all objects in this partial model. This also implies that the other objects in this set shall be placed in the specified engineering object. Note that semantic constraints apply with respect to «atpSplitable»

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
object	Referrable	*	ref	This association applies a particular (usually small) set of objects (e.g. a singular package). Main usage is, if one does not want to create a collection specifically for access control.
objectDefin tion	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. **Stereotypes:** atpUriDef

Table C.3: AclObjectSet

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

Table C.4: AclOperation

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

Attribute	Datatype	Mul.	Kind	Note
aclOperation	AclOperation	*	ref	This denotes an operation which is granted by the given AclPermission.
aclRole	AclRole	*	ref	This denotes the role (individual or even organization) for which the AclPermission. is granted.
aclScope	AclScopeEnum	1	attr	This indicates the scope of applied permissions: explicit, descendant, dependent;

Table C.5: AclPermission

Class	AclRole			
Package	M2::AUTOSARTemplates::GenericStructure::RolesAndRights			
Note	This meta class represents the ability to specify a particular role which is used to grant access rights to AUTOSAR model. The purpose of this meta-class is to support the mutual agreements between the involved parties. **Tags:** atp.recommendedPackage=AclRoles			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,Collectable Element,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	**Datatype**	**Mul.**	**Kind**	**Note**
ldapUrl	UriString	0..1	attr	This is an URL which allows to represent users or organizations taking the particular role.

Table C.6: AclRole

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

Table C.7: AliasNameSet

Class	**ApplicationDataType (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	ApplicationDataType defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake. An ApplicationDataType represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianess, etc. It should be possible to model the application level aspects of a VFB system by using ApplicationDataTypes only.			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,AtpClassifier,AtpType,Autosar DataType,CollectableElement,Identifiable,MultilanguageReferrable,Packageable Element,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.8: ApplicationDataType

Class	**AtomicSwComponentType (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	An atomic software component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs.			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,AtpClassifier,Atp Type,CollectableElement,Identifiable,MultilanguageReferrable,Packageable Element,Referrable,SwComponentType			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
internalBe havior	SwcInternalBeh avior	0..1	aggr	The SwcInternalBehaviors owned by an AtomicSwComponentType can be located in a different physical file. Therefore the aggregation is «atpSplitable». **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalBehavior, variation Point.shortLabel vh.latestBindingTime=preCompileTime
symbolPro ps	SymbolProps	0..1	aggr	This represents the SymbolProps for the AtomicSwComponentType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName

Table C.9: AtomicSwComponentType

Class	**BswModuleDescription**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswOverview			
Note	Root element for the description of a single BSW module or BSW cluster. In case it describes a BSW module, the short name of this element equals the name of the BSW module. **Tags:** atp.recommendedPackage=BswModuleDescriptions			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,AtpClassifier,AtpFeature,Atp StructureElement,CollectableElement,Identifiable,Multilanguage Referrable,PackageableElement,Referrable			
Attribute	**Datatype**	**Mul.**	**Kind**	**Note**
bswModul eDepende ncy	BswModuleDep endency	*	aggr	Describes the dependency to another BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
bswModul eDocumen tation	SwComponentD ocumentation	0..1	aggr	This adds a documentation to the BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswModuleDocumentation, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=6
internalBe havior	BswInternalBeh avior	*	aggr	The various BswInternalBehaviors associated with a BswModuleDescription can be distributed over several physical files. Therefore the aggregation is «atpSplitable». **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName xml.sequenceOffset=65
moduleId	PositiveInteger	0..1	attr	Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen. **Tags:** xml.sequenceOffset=5
outgoingC allback	BswModuleEntr y	*	ref	Specifies a callback, which will be called from this module if required by another module. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=15

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
providedClientServerEntry	BswModuleClientServerEntry	*	aggr	Specifies that this module provides a client server entry which can be called from another parition or core.This entry is declared locally to this context and will be connected to the requiredClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
providedData	VariableDataPrototype	*	aggr	Specifies a data prototype provided by this module in order to be read from another partition or core.The providedData is declared locally to this context and will be connected to the requiredData of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
providedEntry	BswModuleEntry	*	ref	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). **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=10
providedModeGroup	ModeDeclarationGroupPrototype	*	aggr	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. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=25

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
releasedTrigger	Trigger	*	aggr	A Trigger released by this module or cluster. It can be connected to the requiredTriggers of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with Triggers provided via ports by an associated ServiceSwComponentType, EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=35
requiredClientServerEntry	BswModuleClientServerEntry	*	aggr	Specifies that this module requires a client server entry which can be implemented on another parition or core.This entry is declared locally to this context and will be connected to the providedClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
requiredData	VariableDataPrototype	*	aggr	Specifies a data prototype required by this module in oder to be provided from another partition or core.The 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. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
requiredModeGroup	ModeDeclarationGroupPrototype	*	aggr	Specifies that this module or cluster depends on a certain mode group. The requiredModeGroup is local to this context and will be connected to the providedModeGroup of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=30
requiredTrigger	Trigger	*	aggr	Specifies that this module or cluster reacts upon an external trigger.This requiredTrigger is declared locally to this context and will be connected to the providedTrigger of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

Table C.10: BswModuleDescription

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

Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
providedClientServerEntry	BswModuleClientServerEntry	*	aggr	Specifies that this module provides a client server entry which can be called from another parition or core.This entry is declared locally to this context and will be connected to the requiredClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
providedData	VariableDataPrototype	*	aggr	Specifies a data prototype provided by this module in order to be read from another partition or core.The providedData is declared locally to this context and will be connected to the requiredData of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
providedEntry	BswModuleEntry	*	ref	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). **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=10
providedModeGroup	ModeDeclarationGroupPrototype	*	aggr	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. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=25

Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
releasedTrigger	Trigger	*	aggr	A Trigger released by this module or cluster. It can be connected to the requiredTriggers of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with Triggers provided via ports by an associated ServiceSwComponentType, EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=35
requiredClientServerEntry	BswModuleClientServerEntry	*	aggr	Specifies that this module requires a client server entry which can be implemented on another parition or core.This entry is declared locally to this context and will be connected to the providedClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
requiredData	VariableDataPrototype	*	aggr	Specifies a data prototype required by this module in oder to be provided from another partition or core.The 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. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
requiredModeGroup	ModeDeclarationGroupPrototype	*	aggr	Specifies that this module or cluster depends on a certain mode group. The requiredModeGroup is local to this context and will be connected to the providedModeGroup of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=30
requiredTrigger	Trigger	*	aggr	Specifies that this module or cluster reacts upon an external trigger.This requiredTrigger is declared locally to this context and will be connected to the providedTrigger of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

Table C.11: BswModuleDescription

Enumeration	**BindingTimeEnum**
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling
Note	This enumerator specifies the applicable binding times for the pre build variation points.
Literal	***Description***
codeGenerationTime	• Coding by hand, based on requirements document. • Tool based code generation, e.g. from a model. • The model may contain variants. • Only code for the selected variant(s) is actually generated.
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)
preCompile Time	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	• Designing the VFB. • Software Component types (PortInterfaces). • SWC Prototypes and the Connections between SWCprototypes. • Designing the Topology • ECUs and interconnecting Networks • Designing the Communication Matrix and Data Mapping

Table C.12: BindingTimeEnum

Class	≪`atpMixedString`≫ **BlueprintFormula**			
Package	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintFormula			
Note	This class express the extension of the Formula Language to provide formalized blueprint-Value resp. blueprintCondition.			
Base	ARObject,FormulaExpression,SwSystemconstDependentFormula			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
ecucQuery	EcucQuery	1	ref	The EcucQuery serves as a argument for the formula.
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.13: BlueprintFormula

Class	**BlueprintMapping**			
Package	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintDedicated::Generic Blueprint			
Note	This meta-class represents the ability to map two an object and its blueprint.			
Base	ARObject,AtpBlueprintMapping			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
blueprint	AtpBlueprint	1	ref	This represents the mapped blueprint.
derivedObject	AtpBlueprintable	1	ref	This represents the object which was derived from the blueprint.

Table C.14: BlueprintMapping

Class	**BswInternalBehavior**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	Specifies the behavior of a BSW module or a BSW cluster w.r.t. the code entities visible by the BSW Scheduler. It is possible to have several different BswInternalBehaviors referring to the same BswModuleDescription.			
Base	ARObject,AtpClassifier,AtpFeature,AtpStructureElement,Identifiable,Internal Behavior,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
distinguishedPartition	BswDistinguishedPartition	*	aggr	Indicates an abstract partition context in which the enclosing BswModuleEntity can be executed. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
entity	BswModuleEntity	1..*	aggr	A code entity for which the behavior is described **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=5
event	BswEvent	*	aggr	An event required by this module behavior. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=10
internalTriggeringPoint	BswInternalTriggeringPoint	*	aggr	An internal triggering point. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=2
modeReceiverPolicy	BswModeReceiverPolicy	*	aggr	Implementation policy for the reception of mode switches. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=25
modeSenderPolicy	BswModeSenderPolicy	*	aggr	Implementation policy for providing a mode group. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=20

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
perInstanceParameter	ParameterDataPrototype	*	aggr	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. In contrast to constantMemory, this object is not allocated locally by the module's code, but by the BSW Scheduler and it is accessed from the BSW module via the BSW Scheduler API. The main use case is the support of software emulation of calibration data. The aggregation is subject to variability with the purpose to support implementation variants. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
receptionPolicy	BswDataReceptionPolicy	*	aggr	Data reception policy for inter-partition and/or inter-core communication. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
schedulerNamePrefix	BswSchedulerNamePrefix	*	aggr	Optional definition of one or more prefixes to be used for the BswScheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceDependency, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
triggerDirectImplementation	BswTriggerDirectImplementation	*	aggr	Specifies a trigger to be directly implemented via OS calls. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=15

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***

Table C.15: BswInternalBehavior

Class	**BswModuleDependency**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces			
Note	This class collects the dependencies of a BSW module or cluster on a certain other BSW module.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
expectedCallback	BswModuleEntry	*	ref	Indicates a callback expected to be called from another module and implemented by this module. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=15
requiredEntry	BswModuleEntry	*	ref	Indicates an entry into another modules which is required by this module. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=10
serviceItem	ServiceNeeds	*	aggr	A single item (example: Nv block) for which the quality of a service is defined. The aggregation is marked as «atpSplitable» to allow for extension during the ECU configuration process. This association is deprecated since R4.0.3, since ServiceNeeds shall be associated with the new element BswServiceDependency within the BswInternalBehavior. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=obsolete xml.sequenceOffset=20
targetModuleId	PositiveInteger	0..1	attr	AUTOSAR identifier of the target module of which the dependencies are defined. This information is optional, because the target module may also be identified by targetModuleRef. **Tags:** xml.sequenceOffset=5
targetModuleRef	BswModuleDescription	0..1	ref	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. **Stereotypes:** atpUriDef; atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=7

Attribute	*Datatype*	*Mul.*	*Kind*	*Note*

Table C.16: BswModuleDependency

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

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
providedClientServerEntry	BswModuleClientServerEntry	*	aggr	Specifies that this module provides a client server entry which can be called from another parition or core.This entry is declared locally to this context and will be connected to the requiredClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
providedData	VariableDataPrototype	*	aggr	Specifies a data prototype provided by this module in order to be read from another partition or core.The providedData is declared locally to this context and will be connected to the requiredData of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
providedEntry	BswModuleEntry	*	ref	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). **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=10
providedModeGroup	ModeDeclarationGroupPrototype	*	aggr	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. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=25

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
releasedTrigger	Trigger	*	aggr	A Trigger released by this module or cluster. It can be connected to the requiredTriggers of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with Triggers provided via ports by an associated ServiceSwComponentType, EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=35
requiredClientServerEntry	BswModuleClientServerEntry	*	aggr	Specifies that this module requires a client server entry which can be implemented on another parition or core.This entry is declared locally to this context and will be connected to the providedClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
requiredData	VariableDataPrototype	*	aggr	Specifies a data prototype required by this module in oder to be provided from another partition or core.The 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. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
requiredModeGroup	ModeDeclarationGroupPrototype	*	aggr	Specifies that this module or cluster depends on a certain mode group. The requiredModeGroup is local to this context and will be connected to the providedModeGroup of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=30
requiredTrigger	Trigger	*	aggr	Specifies that this module or cluster reacts upon an external trigger.This requiredTrigger is declared locally to this context and will be connected to the providedTrigger of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

Table C.17: BswModuleDescription

Class	**BswModuleEntry**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces			
Note	This class represents a single API entry (C-function prototype) into the BSW module or cluster. The name of the C-function is equal to the short name of this element with one exception: In case of multiple instances of a module on the same CPU, special rules for "infixes" apply, see description of class BswImplementation. **Tags:** atp.recommendedPackage=BswModuleEntrys			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,Collectable Element,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
argument (ordered)	SwServiceArg	*	aggr	An argument belonging to this BswModuleEntry. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivation Time xml.sequenceOffset=45
callType	BswCallType	1	attr	The type of call associated with this service. **Tags:** xml.sequenceOffset=25
executionC ontext	BswExecutionC ontext	1	attr	Specifies the execution context which is required (in case of entries into this module) or guaranteed (in case of entries called from this module) for this service. **Tags:** xml.sequenceOffset=30
isReentran t	Boolean	1	attr	Reentrancy from the viewpoint of function callers: • True: Enables the service to be invoked again, before the service has finished. • False: It is prohibited to invoke the service again before is has finished. **Tags:** xml.sequenceOffset=15
isSynchron ous	Boolean	1	attr	Synchronicity from the viewpoint of function callers: • True: This calls a synchronous service, i.e. the service is completed when the call returns. • False: The service (on semantical level) may not be complete when the call returns. **Tags:** xml.sequenceOffset=20
returnType	SwServiceArg	0..1	aggr	The return type belonging to this bswModuleEntry. **Tags:** xml.sequenceOffset=40

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
role	Identifier	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). **Tags:** xml.sequenceOffset=10
serviceId	PositiveInteger	0..1	attr	Refers to the service identifier of the Standardized Interfaces of AUTOSAR basic software. For non-standardized interfaces, it can optionally be used for proprietary identification. **Tags:** xml.sequenceOffset=5
swServiceImplPolicy	SwServiceImplPolicyEnum	1	attr	Denotes the implementation policy as a standard function call, inline function or macro. This has to be specified on interface level because it determines the signature of the call. **Tags:** xml.sequenceOffset=35

Table C.18: BswModuleEntry

Class	**BuildAction**			
Package	M2::AUTOSARTemplates::GenericStructure::BuildActionManifest			
Note	This meta-class represents the ability to specify a build action.			
Base	ARObject,AtpBlueprint,AtpBlueprintable,BuildActionEntity,Identifiable,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
createdData	BuildActionIoElement	*	aggr	This represents the artifacts which are cated by the processor.
followUpAction	BuildAction	*	ref	This association specifies a set of follow up actions. **Tags:** xml.sequenceOffset=-80
inputData	BuildActionIoElement	*	aggr	This represents the artifacts which are read by the processor.
modifiedData	BuildActionIoElement	*	aggr	This denotes the data which are modifed by the action.
predecessorAction	BuildAction	*	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. **Tags:** xml.sequenceOffset=-90
requiredEnvironment	BuildActionEnvironment	1	ref	This represents the environment which is required to use the specified Processor.

Attribute	Datatype	Mul.	Kind	Note

Table C.19: BuildAction

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

Table C.20: BuildActionEnvironment

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

Table C.21: BuildActionManifest

Class	**Collection**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Element Collection			
Note	This meta-class specifies a collection of elements. A collection can be utilized to express additional aspects for a set of elements. Note that Collection is an ARElement. Therefore it is applicable e.g. for EvaluatedVariant, even if this is not obvious. Usually the category of a Collection is "SET". On the other hand, a Collection can also express an arbitrary relationship between elements. This is denoted by the category "RELATION" (see also [TPS_GST_00347]). In this case the collection represents an association from "sourceElement" to "targetElement" in the role "role". **Tags:** atp.recommendedPackage=Collections			
Base	ARElement,ARObject,CollectableElement,Identifiable,Multilanguage Referrable,PackageableElement,Referrable			
Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
autoCollect	AutoCollectEnum	0..1	attr	This attribute reflects how far the referenced objects are part of the collection. **Tags:** xml.sequenceOffset=20
collectedInstance	AtpFeature	*	iref	This instance ref supports the use case that a particular instance is part of the collection. **Tags:** xml.sequenceOffset=60
element	Identifiable	*	ref	This is an element in the collection. Note that Collection itself is collectable. Therefore collections can be nested. In case of category="RELATION" this represents the target end of the relation. **Tags:** xml.sequenceOffset=40
elementRole	Identifier	0..1	ref	This attribute allows to denote a particular role of the collection. Note that the applicable semantics shall be mutually agreed between the two parties. In particular it denotes the role of element in the context of sourceElement. **Tags:** xml.sequenceOffset=30
sourceElement	Identifiable	*	ref	Only if Category = "RELATION". This represents the source of a relation. **Tags:** xml.sequenceOffset=50
sourceInstance	AtpFeature	*	iref	Only if Category = "RELATION". This represents the source instance of a relation. **Tags:** xml.sequenceOffset=70

Table C.22: Collection

Class	**CompuMethod**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:** atp.recommendedPackage=CompuMethods			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,Collectable Element,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
compuInter nalToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Tags:** xml.sequenceOffset=80
compuPhy sToInternal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Tags:** xml.sequenceOffset=90
displayFor mat	DisplayFormatS tring	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:** xml.sequenceOffset=20
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:** xml.sequenceOffset=30

Table C.23: CompuMethod

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

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
compuScaleContents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. **Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=70; xml.typeElement=false; xml.typeWrapperElement=false
lowerLimit	Limit	0..1	ref	This specifies the lower limit of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
mask	PositiveInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the COMPU-SCALE elements. **Tags:** xml.sequenceOffset=35
shortLabel	Identifier	0..1	ref	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. **Tags:** xml.sequenceOffset=20
symbol	CIdentifier	0..1	ref	The symbol, if provided, is used by code generators to get a C identifier for the CompuScale. The name will be used as is for the code generation, therefore it needs to be unique within the generation context. **Tags:** xml.sequenceOffset=25
upperLimit	Limit	0..1	ref	This specifies the upper limit of a of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

Table C.24: CompuScale

Class	**ConsistencyNeeds**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
Note	This meta-class represents the ability to define requirements on the implicit communication behavior.			
Base	ARObject,AtpBlueprint,AtpBlueprintable,Identifiable,Multilanguage Referrable,Referrable			
Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
dpgDoesNotRequireCoherency	DataPrototypeGroup	*	aggr	This group of VariableDataPrototypes does not require coherency with respect to the implicit communication behavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dpgRequiresCoherency	DataPrototypeGroup	*	aggr	This group of VariableDataPrototypes requires coherency with respect to the implicit communication behavior, i.e. all read and write access to VariableDataPrototypes in the DataPrototypeGroup by the RunnableEntitys of the RunnableEntityGroup need to be handled in a coherent manner. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
regDoesNotRequireStability	RunnableEntityGroup	*	aggr	This group of RunnableEntities does not require stability with respect to the implicit communication behavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
regRequiresStability	RunnableEntityGroup	*	aggr	This group of RunnableEntities requires stability with respect to the implicit communication behavior, i.e. all read and write access to VariableDataPrototypes in the DataPrototypeGroup by the RunnableEntitys of the RunnableEntityGroup need to be handled in a stable manner. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table C.25: ConsistencyNeeds

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

Table C.26: ConsistencyNeedsBlueprintSet

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

Table C.27: DataConstr

Class	**DataPrototypeGroup**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
Note	This meta-class represents the ability to define a collection of DataPrototypes that are subject to the formal definition of implicit communication behavior. The definition of the collection can be nested.			
Base	ARObject,AtpClassifier,AtpFeature,AtpStructureElement,Identifiable,MultilanguageReferrable,Referrable			
Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
dataPrototypeGroup	DataPrototypeGroup	*	iref	This represents the ability to define nested groups of VariableDataPrototypes. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

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

Table C.28: DataPrototypeGroup

Class	**DataTypeMappingSet**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	This class represents a list of mappings between ApplicationDataTypes and ImplementationDataTypes. In addition, it can contain mappings between ImplementationDataTypes and ModeDeclarationGroups. **Tags:** atp.recommendedPackage=DataTypeMappingSets			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
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.29: DataTypeMappingSet

Class	**Documentation**			
Package	M2::AUTOSARTemplates::GenericStructure::DocumentationOnM1			
Note	This meta-class represents the ability to handle a so called standalone documentation. Standalone means, that such a documentation is not embedded in another ARElement or identifiable object. The standalone documentation is an entity of its own which denotes its context by reference to other objects and instances. **Tags:** atp.recommendedPackage=Documentations			
Base	ARElement,ARObject,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
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. **Tags:** xml.sequenceOffset=200

Table C.30: Documentation

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

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

Table C.31: DocumentationBlock

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

Table C.32: EcucDefinitionCollection

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

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
container	EcucContainerDef	1..*	aggr	Aggregates the top-level container definitions of this specific module definition. **Tags:** xml.sequenceOffset=11
refinedModuleDef	EcucModuleDef	0..1	ref	Optional reference from the Vendor Specific Module Definition to the Standardized Module Definition it refines. In case this EcucModuleDef has the category STANDARDIZED_MODULE_DEFINITION this reference shall not be provided. In case this EcucModuleDef has the category VENDOR_SPECIFIC_MODULE_DEFINITION this reference is mandatory. **Stereotypes:** atpUriDef
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.33: EcucModuleDef

Class	**FlatMap**			
Package	M2::AUTOSARTemplates::CommonStructure::FlatMap			
Note	Contains a flat list of references to software objects. This list is used to identify instances and to resolve name conflicts. The scope is given by the RootSwCompositionPrototype for which it is used, i.e. it can be applied to a system, system extract or ECU-extract. An instance of FlatMap may also be used in a preliminary context, e.g. in the scope of a software component before integration into a system. In this case it is not referred by a RootSwCompositionPrototype. **Tags:** atp.recommendedPackage=FlatMaps			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
instance	FlatInstanceDescriptor	1..*	aggr	A descriptor instance aggregated in the flat map. The variation point accounts for the fact, that the system in scope can be subject to variability, and thus the existence of some instances is variable. The aggregation has been made splitable because the content might be contributed by different stakeholders at different times in the workflow. Plus, the overall size might be so big that eventually it becomes more manageable if it is distributed over several files. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild

Table C.34: FlatMap

Class	**Identifiable (abstract)**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.			
Base	ARObject,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
desc	MultiLanguageOverviewParagraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:** xml.sequenceOffset=-60
category	CategoryString	0..1	attr	This element assigns a category to the parent element. The category is intended to specialize the usage and/or the content identifiable object. Such a specialization may also impose particular semantic constraints on the entire substructure (not only the identifiable itself). **Tags:** xml.sequenceOffset=-50
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Tags:** xml.sequenceOffset=-40

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** xml.sequenceOffset=-25
introduction	Documentation Block	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:** xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". **Tags:** xml.attribute=true

Table C.35: Identifiable

Class	**ImplementationDataType**			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:** atp.recommendedPackage=ImplementationDataTypes			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,AtpClassifier,AtpType,AutosarDataType,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
subElement (ordered)	ImplementationDataTypeElement	*	aggr	Specifies an element of an arrray, struct, or union data type. The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a ImplementationDataType representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the ImplementationDataType. **Stereotypes:** atpSplitable **Tags:** 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.36: ImplementationDataType

Class	**LifeCycleInfo**			
Package	M2::AUTOSARTemplates::GenericStructure::LifeCycles			
Note	LifeCycleInfo describes the life cycle state of an element together with additional information like what to use instead			
Base	ARObject			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
lcObject	Referrable	1	ref	Element(s) have the life cycle as described in lcState.
lcState	LifeCycleState	0..1	ref	This denotes the particular state assigned to the object. If no lcState is given then the default life cycle state of 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	DocumentationBlock	0..1	aggr	Remark describing for example • why the element was given the specified life cycle • the semantics of useInstead

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

Table C.37: LifeCycleInfo

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

Table C.38: LifeCycleInfoSet

Class	**LifeCycleState**			
Package	M2::AUTOSARTemplates::GenericStructure::LifeCycles			
Note	This meta class represents one particular state in the LifeCycle.			
Base	ARObject,AtpBlueprint,AtpBlueprintable,Identifiable,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.39: LifeCycleState

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

Table C.40: LifeCycleStateDefinitionGroup

Class	**ModeDeclarationGroup**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:** atp.recommendedPackage=ModeDeclarationGroups			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,AtpClassifier,AtpType,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
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. **Stereotypes:** atpVariation **Tags:** 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 avaliable ModeTransitions of the ModeDeclarationGroup

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
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.41: ModeDeclarationGroup

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

Table C.42: PPortPrototype

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

Table C.43: PackageableElement

Class	**PortInterface (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,AtpClassifier,AtpType,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
isService	Boolean	1	attr	This flag is set if the PortInterface is to be used for communication between an • ApplicationSwComponentType or • ServiceProxySwComponentType or • SensorActuatorSwComponentType or • ComplexDeviceDriverSwComponentType or • EcuAbstractionSwComponentType and a ServiceSwComponentType (namely an AUTOSAR Service) located on the same ECU. Otherwise the flag is not set.
serviceKind	ServiceProviderEnum	0..1	attr	This attribute provides further details about the nature of the applied service.

Table C.44: PortInterface

Class	**PortInterfaceBlueprintMapping**			
Package	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintDedicated::PortInterfaceBlueprint			
Note	This meta-class represents the ability to map two PortInterfaces of which one acts as the blueprint for the other. **Tags:** atp.Status=obsolete			
Base	ARObject,AtpBlueprintMapping			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
derivedPortInterface	PortInterface	1	ref	This represents the derived interface. **Tags:** xml.sequenceOffset=30
portInterfaceBlueprint	PortInterface	1	ref	This represents the interface blueprint. Note that this interface needs to live in a package of category BLUEPRINT. **Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=20

Table C.45: PortInterfaceBlueprintMapping

Class	**PortInterfaceMapping (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	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).			
Base	ARObject,AtpBlueprint,AtpBlueprintable,Identifiable,Multilanguage Referrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.46: PortInterfaceMapping

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

Table C.47: PortInterfaceMappingSet

Class	**PortPrototype (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	ARObject,AtpBlueprintable,AtpFeature,AtpPrototype,Identifiable,Multilanguage Referrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
clientServerAnnotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/server communication.
delegatedPortAnnotation	DelegatedPortAnnotation	0..1	aggr	Annotations on this delegated port.

Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
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 voilatile 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.48: PortPrototype

Class	**PortPrototypeBlueprintMapping**			
Package	M2::AUTOSARTemplates::StandardizationTemplate::BlueprintDedicated::PortProtoypeBlueprint			
Note	This meta-class represents the ability to map a PortPrototypeBlueprint to a PortProtoype of which one acts as the blueprint for the other. **Tags:** atp.Status=obsolete			
Base	ARObject,AtpBlueprintMapping			
Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
derivedPortPrototype	PortPrototype	1	ref	The PortPrototype in the context of the mapping. **Tags:** xml.sequenceOffset=30
portPrototypeBlueprint	PortPrototypeBlueprint	1	ref	The PortPrototypeBlueprint in the context of the mapping. **Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=20

Table C.49: PortPrototypeBlueprintMapping

Class	**RPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	ARObject,AbstractRequiredPortPrototype,AtpBlueprintable,AtpFeature,AtpPrototype,Identifiable,MultilanguageReferrable,PortPrototype,Referrable			
Attribute	*Datatype*	*Mul.*	*Kind*	*Note*

Attribute	Datatype	Mul.	Kind	Note
requiredInterface	PortInterface	1	tref	The interface that this port requires, i.e. the port depends on another port providing the specified interface. **Stereotypes:** isOfType

Table C.50: RPortPrototype

Class	Referrable (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
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. **Tags:** xml.enforceMinMultiplicity=true; xml.sequenceOffset=-100

Table C.51: Referrable

Class	RunnableEntity			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
Note	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.			
Base	ARObject,AtpClassifier,AtpFeature,AtpStructureElement,ExecutableEntity,Identifiable,MultilanguageReferrable,Referrable			
Attribute	Datatype	Mul.	Kind	Note
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. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
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".
dataReadAccess	VariableAccess	*	aggr	RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataReadAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataReadAccess in the implementation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
dataReceivePointByArgument	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of an argument in the function signature. The aggregation of dataReceivePointByArgument is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data receive points in the implementation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
dataReceivePointByValue	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of the return value. The aggregation of 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. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
dataSendPoint	VariableAccess	*	aggr	RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataSendPoint is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data send points in the implementation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
dataWriteAccess	VariableAccess	*	aggr	RunnableEntity has implicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataWriteAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataWriteAccess in the implementation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
externalTriggeringPoint	ExternalTriggeringPoint	*	aggr	The aggregation of ExternalTriggeringPoint is subject to variability with the purpose to support the conditional existence of trigger ports or the variant existence of external triggering points in the implementation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
internalTriggeringPoint	InternalTriggeringPoint	*	aggr	The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
modeAccessPoint	ModeAccessPoint	*	aggr	The runnable has a mode access point. The aggregation of ModeAccessPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode access points in the implementation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
modeSwitchPoint	ModeSwitchPoint	*	aggr	The runnable has a mode switch point. The aggregation of ModeSwitchPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode switch points in the implementation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
parameterAccess	ParameterAccess	*	aggr	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. 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. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
readLocalVariable	VariableAccess	*	aggr	The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableDataPrototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicitInterRunnableVariable or the variant existence of readLocalVariable (points) in the implementation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
serverCallPoint	ServerCallPoint	*	aggr	The RunnableEntity has a ServerCallPoint. The aggregation of ServerCallPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes or the variant existence of server call points in the implementation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
symbol	CIdentifier	1	ref	The symbol describing this RunnableEntity's entry point. This is considered the API of the RunnableEntity and is required during the RTE contract phase.
waitPoint	WaitPoint	*	aggr	The WaitPoint associated with the RunnableEntity.

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

Table C.52: RunnableEntity

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

Table C.53: RunnableEntityGroup

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

Note	SectionInitializationPolicyType describes the intended initialization of MemorySections. The following values are standardized in AUTOSAR Methodology: • **NO-INIT**: No initialization and no clearing is performed. Such data elements shall not be read before one has written a value into it. • **INIT**: To be used for data that are initialized by every reset to the specified value (initValue). • **POWER-ON-INIT**: To be used for data that are initialized by "Power On" to the specified value (initValue). Note: there might be several resets between power on resets. • **CLEARED**: To be used for data that are initialized by every reset to zero. • **POWER-ON-CLEARED**: To be used for data that are initialized by "Power On" to zero. Note: there might be several resets between power on resets. Please note that the values are defined similar to the representation of enumeration types in the XML schema to ensure backward compatibility. **Tags:** xml.xsd.customType=SECTION-INITIALIZATION-POLICY-TYPE; xml.xsd.type=NMTOKEN

Table C.54: SectionInitializationPolicyType

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

Table C.55: SenderReceiverInterface

Class	**SwAddrMethod**			
Package	M2::AUTOSARTemplates::CommonStructure::AuxillaryObjects			
Note	Used to assign a common addressing method, e.g. common memory section, to data or code objects. These objects could actually live in different modules or components. **Tags:** atp.recommendedPackage=SwAddrMethods			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,Collectable Element,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
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 addresssing method.

Table C.56: SwAddrMethod

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

Table C.57: SwBaseType

Class	**SwComponentType (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for AUTOSAR software components.			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,AtpClassifier,AtpType,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
consistencyNeeds	ConsistencyNeeds	*	aggr	This represents the colelction of ConsistencyNeeds owned by the enclosing SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
port	PortPrototype	*	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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
swComponentDocumentation	SwComponentDocumentation	0..1	aggr	This adds a documentation to the SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** 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.58: SwComponentType

Class	**SwServiceArg**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceProcessTask			
Note	Specifies the properties of a data object exchanged during the call of an SwService, e.g. an argument or a return value.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
direction	ArgumentDirectionEnum	0..1	attr	Specifies the direction of the data transfer. The direction shall indicate the direction of the actual information that is being consumed by the caller and/or the callee, not the direction of formal arguments in C. The attribute is optional for backwards compatibility reasons. For example, if a pointer is used to pass a memory address for the expected result, the direction shall be "out". If a pointer is used to pass a memory address with content to be read by the callee, its direction shall be "in". **Tags:** xml.sequenceOffset=10
swArraysize	ValueList	0..1	aggr	This turns the argument of the service to an array. **Tags:** xml.sequenceOffset=20
swDataDefProps	SwDataDefProps	0..1	aggr	Data properties of this SwServiceArg. **Tags:** xml.sequenceOffset=30

Table C.59: SwServiceArg

Class	**TDEventVfbPort (abstract)**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventVfb			
Note	This is the abstract parent class to describe specific timing event types at Virtual Function Bus (VFB) level.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TDEventVfb,TimingDescription,TimingDescriptionEvent			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
isExternal	Boolean	1	attr	This attribute is used to refer to external events that are related to hardware I/O, like physical sensors and actuators, at Virtual Function Bus (VFB) level.
port	PortPrototype	0..1	ref	The port scope of the timing event.
portPrototypeBlueprint	PortPrototypeBlueprint	0..1	ref	The PortPrototypeBlueprint is the scope of the timing event.

Table C.60: TDEventVfbPort

Class	**VariableDataPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	A VariableDataPrototype is used to contain values in an ECU application. This means that most likely a VariableDataPrototype allocates "static" memory on the ECU. In some cases optimization strategies might lead to a situation where the memory allocation can be avoided. In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes.			
Base	ARObject,AtpFeature,AtpPrototype,AutosarDataPrototype,DataPrototype,Identifiable,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

Table C.61: VariableDataPrototype

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

Table C.62: VfbTiming

Class	**VariationPoint**			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
Note	This meta-class represents the ability to express a "structural variation point". The container of the variation point is part of the selected variant if swSyscond evaluates to true and each postBuildVariantCriterion is fulfilled.			
Base	ARObject			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
desc	MultiLanguageOverviewParagraph	0..1	aggr	This allows to describe shortly the purpose of the variation point. **Tags:** xml.sequenceOffset=20

Attribute	*Datatype*	*Mul.*	*Kind*	*Note*
blueprintCondition	DocumentationBlock	0..1	aggr	This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint. Note that variationPoints are not allowed within a blueprintCondition. **Tags:** xml.sequenceOffset=28
formalBlueprintCondition	BlueprintFormula	0..1	aggr	This denotes a formal blueprintCondition. This shall be not in contradiction with blueprintCondition. It is recommanded only to use one of the two. **Tags:** xml.sequenceOffset=29
postBuildVariantCondition	PostBuildVariantCondition	*	aggr	This is the set of post build variant conditions which all shall be fulfilled in order to (postbuild) bind the variation point. **Tags:** xml.sequenceOffset=40
sdg	Sdg	0..1	aggr	An optional special data group is attached to every variation point. These data can be used by external software systems to attach application specific data. For example, a variant management system might add an identifier, an URL or a specific classifier. **Tags:** xml.sequenceOffset=50
shortLabel	Identifier	0..1	ref	This provides a name to the particular variation point to support the RTE generator. It is necessary for supporting splitable aggregations and if binding time is later than codeGenerationTime, as well as some RTE conditions. It needs to be unique with in the enclosing Identifiables with the same ShortName. **Tags:** xml.sequenceOffset=10
swSyscond	ConditionByFormula	0..1	aggr	This condition acts as Binding Function for the VariationPoint. Note that the mulitplicity is 0..1 in order to support pure postBuild variants. **Tags:** xml.sequenceOffset=30

Table C.63: VariationPoint

D Variation Points in this Template

Variation Point	**Latest Binding Time**
`ConsistencyNeedsBlueprintSet.consistencyNeeds`	(`preCompileTime`)

Table D.1: Usage of variation points