Document Title	Specification of Manifest
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	713

Document Status	Final
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	17-03

Document Change History			
Date	**Release**	**Changed by**	**Description**
2017-03-31	17-03	AUTOSAR Release Management	• Initial release

Disclaimer

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

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

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

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

References

[1] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture

[3] Reference Model for Service Oriented Architecture 1.0
https://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[4] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[5] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

[6] ISO/IEC 14882:2011, Information technology – Programming languages – C++
http://www.iso.org

[7] Specification of Communication Management
AUTOSAR_SWS_CommunicationManagement

[8] Diagnostic Extract Template
AUTOSAR_TPS_DiagnosticExtractTemplate

[9] System Template
AUTOSAR_TPS_SystemTemplate

[10] Specification of ECU Resource Template
AUTOSAR_TPS_ECUResourceTemplate

1 Introduction

This document contains the specification of the so-called the *Manifest* on the *AUTOSAR adaptive platform*. A description of the overall modeling approach can be found in section 1.1. A reference to the definition of the term *service* is given in section 1.2.

The term *Manifest* is used in this specification in the meaning of a formal specification of configuration content. Please find a more detailed description of the term and the implications for the *AUTOSAR adaptive platform* in section 2.

Please note that the content of the document (despite the name) extends to the description of design elements necessary to develop software for the *AUTOSAR adaptive platform*.

The design-related modeling mainly is focused on the development of application software on the *AUTOSAR adaptive platform* as well as the connection between application and diagnostics and is described in detail[1] in section 3 and section 4.

Section 5 represents that counterpart to section 3 on deployment level, it describes the content of the so-called *application manifest*.

Section 6 provides a detailed description of how service-oriented communication shall be configured on *manifest* level.

Section 7 describes the options for configuring a machine by means of a *manifest*.

1.1 Modeling Approach

The *AUTOSAR adaptive platform* has been introduced when the *AUTOSAR classic platform* was already a stable and well-established standard in the automotive domain.

And yet, the *AUTOSAR adaptive platform* is no successor of the *AUTOSAR classic platform*. Both platforms complement each other for specific use cases that can be better implemented by one or the other platform.

In this situation, two possible approaches for modeling on the *AUTOSAR adaptive platform* could have been taken:

- The *AUTOSAR adaptive platform* is based on different principles than the *AUTOSAR classic platform*, and hence the modeling approach could also **decouple from the canon of the AUTOSAR classic platform as much as possible** to advertise the fact that the two platforms have different purposes.

[1] The description of the design elements may be moved to other model-related documents in the future.
But for the time being, there is a coexistence of manifest-related and design-related model elements in this document.

Consequentially, even if specific model elements have clear counterparts in the respective other platform, use a different terminology to not confuse the users of both platforms.

- Despite the undeniable differences between the two platforms, there is still a significant number of striking similarities that strongly encourage the **usage of existing modeling concepts** from the *AUTOSAR classic platform*, especially from the specification of the AUTOSAR Software-Component Template [1], as much as possible.

  Consequentially, the conclusion is to use the identical meta-classes for similar purposes on both platforms. It will then be necessary to extend some of the affected meta-classes platform specific where applicable and add constraints that clarify the platform-specific usage of the mentioned extensions.

Without further ado, the modeling approach for the *AUTOSAR adaptive platform* follows the second alternative.

This means, for example, that a piece of application software on the *AUTOSAR adaptive platform* shall be represented by an `SwComponentType`. This includes the definition of `CompositionSwComponentType`s that in turn aggregate `SwComponentPrototype`s typed by e.g. (in case of the *AUTOSAR adaptive platform*) `AdaptiveApplicationSwComponentType`s.

This also means that an `AtomicSwComponentType` used on the *AUTOSAR adaptive platform* shall **not** aggregate `AtomicSwComponentType.internalBehavior` because the latter is reserved for usage on the *AUTOSAR classic platform*.

The reuse of existing model-elements for the definition of the meta-model for the *AUTOSAR adaptive platform* has the side effect that the descriptions of existing model elements may contain references to technical details that only make sense on the *AUTOSAR classic platform*.

After all, the model elements were created when only the *AUTOSAR classic platform* existed.

These references shall be taken with a grain of salt. It is expected that readers can abstract from those details and extract the aspects of these model elements that create relevance for the description of the *AUTOSAR adaptive platform*.

1.2 The Term Service

It is essential to keep in mind that the term *service* is frequently used within this document in particular and the *AUTOSAR adaptive platform* in general.

This usage has its reasons despite the fact that the meaning of the term *service* on the *AUTOSAR adaptive platform* collides with other meanings used within AUTOSAR.

In summary, the following meaning of the term *service* exist in the scope of AUTOSAR:

- The Term *service* is used in the layered software architecture [2] to denote the highest layer of the AUTOSAR software architecture that interacts with the application. In this context, model elements like `ServiceSwComponentType`, `SwcServiceDependency`, `ServiceNeeds`, or `PortInterface.isService` have been created on the *AUTOSAR classic platform*.
- The term *service* is used to express that information is related or required in a workshop where a car is **serviced**. In this context, *service-only diagnostic trouble codes* (DTC) are defined.
- The term *service* is used to describe the handling of **diagnostic services**, e.g. UDS service *ReadDataByIdentifier*, for the communication between a diagnostic tester and a diagnostic stack on an (AUTOSAR) ECU.
- the term *service* is used in the meaning defined by the **service-oriented architecture** (SOA) [3]. This meaning has the strongest relation to the usage of the term *service* on the *AUTOSAR adaptive platform*.

1.3 Abbreviations

The following table contains a list of abbreviations used in the scope of this document along with the spelled-out meaning of each of the abbreviations.

Abbreviation	*Meaning*
API	Application Programming Interface
ATP	AUTOSAR Template Profile
ARXML	AUTOSAR XML
DM	Diagnostic Manager
DTC	Diagnostic Trouble Code
ECU	Electrical Control Unit
ID	Identifier
IO	Input/Output
IP	Internet Protocol
ISO	International Standardization Organization
LAN	Local Area Network
MAC	Media Access Control
NM	Network Management
NV	Non-Volatile
OEM	Original Equipment Manufacturer
OS	Operating System
PDU	Protocol Data Unit
POSIX	Portable Operating System Interface
RAM	Random Access Memory

Abbreviation	*Meaning*
ROM	Read-Only Memory
SD	Service Discovery
SDG	Special Data Group
SOME/IP	Scalable service-Oriented MiddlewarE over IP
SWC	Software Component
TCP	Transport Control Protocol
TTL	Time to Live
UDS	Unified Diagnostic Services
UDP	User datagram Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VFB	Virtual Functional Bus
VLAN	Virtual Local Area Network
VSA	Variable Size Array
XML	Extensible Markup Language
XSD	XML Schema Definition

Table 1.1: Abbreviations used in the scope of this Document

1.4 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	***Type***	***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 1.2: AUTOSAR

The first rows in the table have the following meaning:

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

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

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

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

The headers in the table have the following meaning:

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

Type: The type of an attribute of the class.

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

Kind: Specifies, whether the attribute is aggregated in the class (`aggr` aggregation), an UML attribute in the class (`attr` primitive attribute), or just referenced by it (`ref`

reference). Instance references are also indicated (`iref` instance reference) in this field.

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

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

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([4]).

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([4]).

1.5 Requirements Tracing

Requirements against this document are exclusively stated in the corresponding requirements document.

The following table 1.3 references the requirements specified in the corresponding requirements document and provides information about individual specification items that fulfill a given requirement.

Requirement	Description	Satisfied by
[RS_MANI_00001]	Adaptive AUTOSAR Application	[TPS_MANI_01008] [TPS_MANI_01009]
[RS_MANI_00002]	Declaration of provided and required services in an application	[TPS_MANI_01039] [TPS_MANI_01040] [TPS_MANI_01052] [TPS_MANI_01053] [TPS_MANI_01057]
[RS_MANI_00003]	Specification of service interfaces	[TPS_MANI_01001] [TPS_MANI_01004] [TPS_MANI_01005] [TPS_MANI_01006] [TPS_MANI_01007] [TPS_MANI_01033] [TPS_MANI_01034] [TPS_MANI_01035] [TPS_MANI_01055]
[RS_MANI_00004]	Support of application design	[TPS_MANI_01010]
[RS_MANI_00005]	Configuration of diagnostic capabilities of an application	[TPS_MANI_01037] [TPS_MANI_01038] [TPS_MANI_01048] [TPS_MANI_01049] [TPS_MANI_01050] [TPS_MANI_01051] [TPS_MANI_01060]
[RS_MANI_00006]	Support of application deployment	[TPS_MANI_01011]
[RS_MANI_00007]	Configuration of application startup behavior	[TPS_MANI_01012] [TPS_MANI_01013] [TPS_MANI_01014] [TPS_MANI_01015] [TPS_MANI_01017] [TPS_MANI_01041] [TPS_MANI_01045] [TPS_MANI_01046] [TPS_MANI_01059] [TPS_MANI_01061]

[RS_MANI_00008]	Service interface deployment to a transport layer mechanism	[TPS_MANI_03036] [TPS_MANI_03037] [TPS_MANI_03038] [TPS_MANI_03039] [TPS_MANI_03070] [TPS_MANI_03071] [TPS_MANI_03072] [TPS_MANI_03073] [TPS_MANI_03074] [TPS_MANI_03075] [TPS_MANI_03101] [TPS_MANI_03103] [TPS_MANI_03104] [TPS_MANI_03105] [TPS_MANI_03106] [TPS_MANI_03107] [TPS_MANI_03108]
[RS_MANI_00009]	Service instance configuration on the network-level	[TPS_MANI_03001] [TPS_MANI_03002] [TPS_MANI_03003] [TPS_MANI_03004] [TPS_MANI_03007] [TPS_MANI_03008] [TPS_MANI_03009] [TPS_MANI_03010] [TPS_MANI_03022] [TPS_MANI_03023] [TPS_MANI_03024] [TPS_MANI_03049] [TPS_MANI_03061]
[RS_MANI_00011]	Instantiation of provided and required services in an application	[TPS_MANI_03000] [TPS_MANI_03100]
[RS_MANI_00014]	User defined transport layer mechanisms	[TPS_MANI_03032] [TPS_MANI_03045] [TPS_MANI_03046] [TPS_MANI_03047] [TPS_MANI_03048] [TPS_MANI_03102]
[RS_MANI_00015]	Definition of the nature of a manifest	[TPS_MANI_01000] [TPS_MANI_01019] [TPS_MANI_01020]
[RS_MANI_00016]	Usage of data types specifically on the AUTOSAR adaptive platform	[TPS_MANI_01016] [TPS_MANI_01018] [TPS_MANI_01027] [TPS_MANI_01028] [TPS_MANI_01029] [TPS_MANI_01030] [TPS_MANI_01042] [TPS_MANI_01043] [TPS_MANI_01044] [TPS_MANI_01047] [TPS_MANI_01062] [TPS_MANI_01063]
[RS_MANI_00017]	Specification of the mapping of Service Interfaces	[TPS_MANI_01002] [TPS_MANI_01003] [TPS_MANI_01022] [TPS_MANI_01024] [TPS_MANI_01025] [TPS_MANI_01026] [TPS_MANI_01032] [TPS_MANI_01058]
[RS_MANI_00018]	Network connections of the machine	[TPS_MANI_03035] [TPS_MANI_03052] [TPS_MANI_03053]
[RS_MANI_00019]	Service discovery message exchange configuration	[TPS_MANI_03064]
[RS_MANI_00020]	Hardware resources of the machine	[TPS_MANI_03035] [TPS_MANI_03065]
[RS_MANI_00021]	Description of machine states	[TPS_MANI_03035] [TPS_MANI_03066]
[RS_MANI_00022]	Adaptive Platform configuration	[TPS_MANI_03035]
[RS_MANI_00023]	Adaptive Module configuration	[TPS_MANI_03035] [TPS_MANI_03056] [TPS_MANI_03096] [TPS_MANI_03098]

[RS_MANI_00024]	SOME/IP transport layer mechanisms	[TPS_MANI_03002] [TPS_MANI_03003] [TPS_MANI_03004] [TPS_MANI_03007] [TPS_MANI_03008] [TPS_MANI_03009] [TPS_MANI_03010] [TPS_MANI_03011] [TPS_MANI_03012] [TPS_MANI_03013] [TPS_MANI_03014] [TPS_MANI_03015] [TPS_MANI_03016] [TPS_MANI_03017] [TPS_MANI_03018] [TPS_MANI_03019] [TPS_MANI_03020] [TPS_MANI_03021] [TPS_MANI_03022] [TPS_MANI_03023] [TPS_MANI_03024] [TPS_MANI_03025] [TPS_MANI_03026] [TPS_MANI_03027] [TPS_MANI_03028] [TPS_MANI_03029] [TPS_MANI_03030] [TPS_MANI_03031] [TPS_MANI_03040] [TPS_MANI_03041] [TPS_MANI_03042] [TPS_MANI_03043] [TPS_MANI_03044] [TPS_MANI_03049] [TPS_MANI_03050] [TPS_MANI_03051] [TPS_MANI_03057] [TPS_MANI_03059] [TPS_MANI_03061] [TPS_MANI_03067] [TPS_MANI_03068] [TPS_MANI_03069] [TPS_MANI_03070] [TPS_MANI_03071] [TPS_MANI_03072] [TPS_MANI_03073] [TPS_MANI_03074] [TPS_MANI_03075]
[RS_MANI_00025]	Definition and configuration of serialization	[TPS_MANI_03101] [TPS_MANI_03102] [TPS_MANI_03103] [TPS_MANI_03104] [TPS_MANI_03105] [TPS_MANI_03106] [TPS_MANI_03107] [TPS_MANI_03108]

Table 1.3: RequirementsTracing

2 Big Picture of Manifest Definition

2.1 Design vs. Deployment

Despite the name, this document contains the description of model elements that are clearly bound to a *design* workflow **and** model elements that have a strong relation to the *deployment* aspect.

Model elements discussed in this document are either related to *design* or *deployment*, there is no overlap between the two groups.

Model elements that are related to *deployment* will be used in models that are uploaded to a target platform, see [TPS_MANI_01000]. These model elements are mainly described in sections of this document where the term "Manifest" is part of the section title.

In the absence of a more precise definition, model elements related to *design* can be identified by not being related to *deployment*.

The structure of the document maps to the division between *design* and *deployment* such that the *design* aspect is mostly described in sections 3 and 4. Chapters 5, 6 , and 7 focus on *deployment*-related content.

2.2 About Manifest

This chapter shall clarify the definition of the term `Manifest` in the context of the *AUTOSAR adaptive platform*.

[TPS_MANI_01000] Definition of the term `Manifest` ⌈ A `Manifest` represents a piece of AUTOSAR model description that is created to support the configuration of an *AUTOSAR adaptive platform* product and which is uploaded to the *AUTOSAR adaptive platform* product, potentially in combination with other artifacts (like binary files) that contain executable code to which the `Manifest` applies. ⌋*(RS_MANI_00015)*

It is important to stress the fact that the usage of a `Manifest` is indeed strictly limited to the *AUTOSAR adaptive platform* and that there is no use case to port the concept to the *AUTOSAR classic platform*.

2.3 Serialization Format

One aspect that the definition of a `Manifest` has in common with other AUTOSAR model content is the standardized serialization format.

[TPS_MANI_01020] Serialization format of the `Manifest` in AUTOSAR ⌈ The standardized serialization format of `Manifest` content in AUTOSAR is ARXML.

Consequently, Manifest model content can be validated against the AUTOSAR XML Schema. ⌋(*RS_MANI_00015*)

An important consequence of [TPS_MANI_01020] is that there is no limitation to just one "manifest file" a.k.a. "the manifest".

Content may be distributed among several physical files according to the rules given in the specification of the AUTOSAR Generic Structure Template [5].

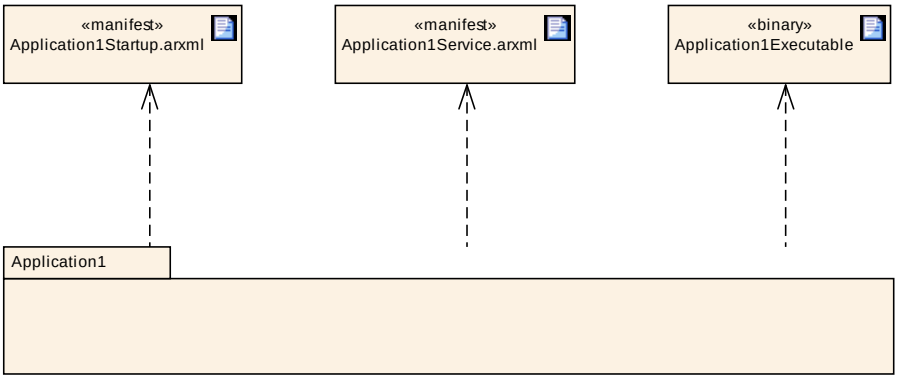

Figure 2.1: Example usage of several manifest files within one software delivery

[TPS_MANI_01021] Serialization format of Manifest content on a machine ⌈ The serialization format used to actually upload a manifest on a machine may be freely chosen by a platform supplier.

However, the content and semantics of the original ARXML Manifest needs to be **fully preserved**. ⌋*()*

It can be expected that in many cases the best option for the upload of the Manifest will still be ARXML because a custom format obviously has to support the full complexity of the Manifest meta-model.

Please note that the meta-model foresees the existence of references from manifest-related meta-classes to design-related meta-classes.

These references are created for the sake of clarity but it is not mandatory that the content of the reference actually needs to be resolvable.

In terms of the AUTOSAR modeling approach, this translates to a decoration of these references with the stereotype ≪atpUriDef≫. More information can be found in [5].

If the referenced meta-classes contain information that is relevant for the manifest level then this information is replicated on the manifest level (such that the manifest-level model does not have to rely on the availability of design-level information).

2.4 Scope

As mentioned before, the usage of a Manifest is limited to the *AUTOSAR adaptive platform*. This does not mean, however, that all ARXML produced in a develop-

ment project that targets the *AUTOSAR adaptive platform* is automatically considered a `Manifest`.

In fact, the *AUTOSAR adaptive platform* is usually not exclusively used in a vehicle project.

A typical vehicle will most likely be also equipped with a number of ECUs developed on the *AUTOSAR classic platform* and the system design for the entire vehicle will therefore have to cover both ECUs built on top of the *AUTOSAR classic platform* and those created on top of the *AUTOSAR adaptive platform*.

[TPS_MANI_01019] `Manifest` content may apply to different aspects of the *AUTOSAR adaptive platform* ⌈ `Manifest` content can apply to different aspects of the model. At the moment, `Manifest` content can roughly be divided into three focus areas:

- Application-related `Manifest` content describes all aspects of the deployment of an application, including - but not limited to - the startup configuration and the configuration of service-oriented communication
- Machine-related `Manifest` content describes the deployment of just a machine, i.e. without any application (including platform modules, see [TPS_MANI_01009]) running on the machine.
- Service instance-related `Manifest` describes how service-oriented communication is bound to endpoints in the application and (in some cases) platform software.

⌋*(RS_MANI_00015)*

2.5 Manifests described in this Document

In principle, the term `Manifest` could be defined such that there is conceptually just one "manifest" and every deployment aspect would be handled in this context.

This does not seem appropriate because it became apparent that manifest-related model-elements exist that are relevant in entirely different phases of a typical development project.

This aspect is taken as the main motivation to subdivide the definition of the term `Manifest` in three different partitions:

Application Manifest This kind of `Manifest` is used to specify the deployment-related information of applications running on the *AUTOSAR adaptive platform*.

A `Application Manifest` is bundled with the actual executable code in order to support the integration of the executable code onto the machine.

Please find more information regarding this topic in section 5.

Service Instance Manifest This kind of `Manifest` is used to specify how service-oriented communication is configured in terms of the requirements of the underlying transport protocols.

A `Service Instance Manifest` is bundled with the actual executable code that implements the respective usage of service-oriented communication.

Please find more information regarding this topic in section 6.

Machine Manifest This kind of `Manifest` is supposed to describe deployment-related content that applies to the configuration of just the underlying machine (i.e. without any applications running on the machine) that runs an *AUTOSAR adaptive platform*.

A `Machine Manifest` is bundled with the software taken to establish an instance of the *AUTOSAR adaptive platform*.

Please find more information regarding this topic in section 7.

The temporal division between the definition (and usage) of different kinds of `Manifest` leads to the conclusion that in most cases different physical files will be used to store the content of the three kinds of `Manifest`.

However, as with all kinds of ARXML content, this is not a binding rule.

3 Application Design

3.1 Overview

This chapter describes all design-related modeling that applies to the creation of application software on the *AUTOSAR adaptive platform*.

This also extends to extensions of existing modeling used on the *AUTOSAR classic platform*, e.g. the introduction of new values of the attribute `category`.

In particular, this section of the document focuses on the following aspects:

- Definition of a dedicated subclass of `SwComponentType` for the *AUTOSAR adaptive platform* (section 3.3)
- Definition of data types specifically for the *AUTOSAR adaptive platform* (section 3.3)
- Service interface as the pivotal element for service-oriented communication (section 3.4)
- Service interface mapping as a mediator between internal and external communication (section 3.5)
- Aspects of the service-oriented communication from the inside of a software-component (section 3.7)
- Adaptive AUTOSAR application as a starting point for the transition towards the deployment (section 3.8)

3.2 Software Component

In principle, it would be possible to directly take over the definition of e.g. `ApplicationSwComponentType` for the usage on the *AUTOSAR adaptive platform*.

However, this would complicate the formulation of constraints regarding the existence of model elements (for example: data types, as explained in section 3.3) that are exclusive to the *AUTOSAR adaptive platform*.

Therefore, the `AdaptiveApplicationSwComponentType` is defined as a representation of software-components on the *AUTOSAR adaptive platform*.

The Existence of the `AdaptiveApplicationSwComponentType` allows for a convenient way (see [constr_1492]) to lock out most kinds of software-component defined for the *AUTOSAR classic platform* from the usage on the *AUTOSAR adaptive platform*.

The clarification of the opposite direction (i.e. an erroneous use of an `AdaptiveApplicationSwComponentType`) is less obvious.

In other words, it may be possible to use a AdaptiveApplicationSwComponentType within a System as some sort of overall design model for software on both the *AUTOSAR classic platform* **and** the *AUTOSAR adaptive platform*.

This aspect, however, is not clarified so far nor is a restriction in place that prohibits AdaptiveApplicationSwComponentType to appear in the context of a System.

Later versions of this specification may fix the missing regulation.

Class	**AdaptiveApplicationSwComponentType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign			
Note	This meta-class represents the ability to support the formal modeling of application software on the AUTOSAR adaptive platform. Consequently, it shall only be used on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft; atp.recommendedPackage=AdaptiveApplicationSw ComponentTypes			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
internalBe havior	AdaptiveSwcInt ernalBehavior	0..1	aggr	This aggregation represents the internal behavior of the AdaptiveApplicationSwComponentType for the AUTOSAR adaptive platform. **Tags:** atp.Status=draft

Table 3.1: AdaptiveApplicationSwComponentType

Class	**AdaptiveSwcInternalBehavior**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::AdaptiveInternal Behavior			
Note	This meta-class represents the ability to define an internal behavior of an AtomicSwComponentType used on the AUTOSAR adaptive platform. Please note that the model of internal behavior in this case, in stark contrast to the situation of the AUTOSAR classic platform, is very minimal. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
serviceDep endency	SwcServiceDep endency	*	aggr	This represents the collection of SwcServiceDependencys owned by AdaptiveInternalBehavior. **Tags:** atp.Status=draft

Table 3.2: AdaptiveSwcInternalBehavior

3.3 Data Type

3.3.1 Overview

The specification of data types on the *AUTOSAR adaptive platform* follows the same pattern as the counterpart on the *AUTOSAR classic platform*: data types are defined on different levels of abstraction that complement each other.

In the context of this document, the focus is on the discussion of `ApplicationDataType`s and `ImplementationDataType`s.

In general, most of the concepts regarding the definition of data types can be taken over from the existing specifications on the *AUTOSAR classic platform*.

However, some aspects are specific to the *AUTOSAR adaptive platform* and are consequently discussed in the scope of this document rather than the specification of the AUTOSAR Software Component Template [1].

One of the aspects that could be taken over from the *AUTOSAR classic platform* is the definition of initial values.

Although the utility of initial values is certainly limited on the *AUTOSAR adaptive platform*, there is an opportunity to utilize the definition of initial values in the context of the so-called `Field`s (see [TPS_MANI_01034]).

3.3.2 ApplicationDataType

The full range of the modeling of `ApplicationDataType`s that is supported on the *AUTOSAR classic platform* can directly be used on the *AUTOSAR adaptive platform* as well.

In addition to the `ApplicationDataType`s supported on the *AUTOSAR classic platform*, there are further `ApplicationDataType`s that - while in principle also available on the *AUTOSAR classic platform* - are primarily used on and designed for the *AUTOSAR adaptive platform*.

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, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.3: ApplicationDataType

3.3.2.1 String Data Type

While the handling of data types that represent textual strings is very similar with respect the definition of `ApplicationDataType`s on the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*, special regulations apply on the level of `ImplementationDataType`s on the *AUTOSAR adaptive platform*.

For more information about the modeling of string data types on the level of `ImplementationDataType` please refer to section 3.3.3.1.

For the sake of consistency, this chapter summarizes the modeling of `ApplicationDataType`s for the modeling of data types that represent textual strings as far as the *AUTOSAR adaptive platform* is concerned.

The meta-classes used to define an `ApplicationPrimitiveDataType` of `category STRING` are summarized in Figure 3.1.

Please note that thanks to the usage of programming languages with richer data types than plain C, the implementation of an `ApplicationPrimitiveDataType` of `category STRING` on the *AUTOSAR adaptive platform* is predefined for a given *language binding*.

[TPS_MANI_01047] Existence of `SwRecordLayout` for an `ApplicationPrimitiveDataType` of `category STRING` ⌈ For the usage of an `ApplicationPrimitiveDataType` of `category STRING` on the *AUTOSAR adaptive platform*, the existence of `ApplicationPrimitiveDataType.swDataDefProps.swRecordLayout` shall be ignored. ⌋*(RS_MANI_00016)*

Please note that [TPS_MANI_01047] intentionally does not forbid the existence of `SwRecordLayout` because the same `ApplicationPrimitiveDataType` of `cat-`

egory STRING could rightfully be used **on both** the *AUTOSAR adaptive platform* and the *AUTOSAR classic platform*.

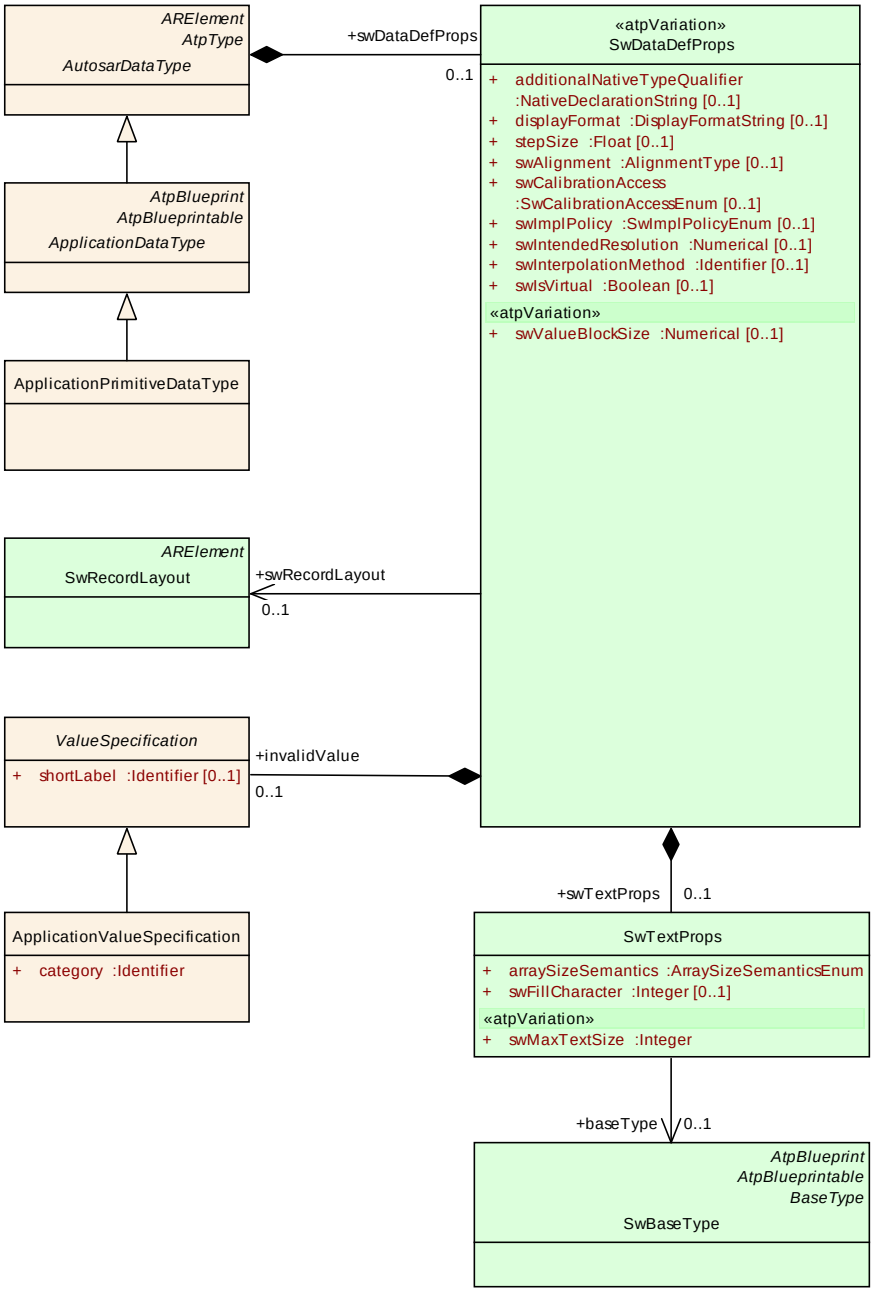

Figure 3.1: Specification of textual strings

Class	**ApplicationPrimitiveDataType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	A primitive data type defines a set of allowed values. **Tags:** atp.recommendedPackage=ApplicationDataTypes			
Base	ARElement, ARObject, ApplicationDataType, AtpBlueprint, AtpBlueprintable, Atp Classifier, AtpType, AutosarDataType, CollectableElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.4: ApplicationPrimitiveDataType

Class	**SwTextProps**			
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	This meta-class expresses particular properties applicable to strings in variables or calibration parameters.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
arraySizeSemantics	ArraySizeSemanticsEnum	1	attr	This attribute controls the semantics of the arraysize for the array representing the string in an ImplementationDataType. It is there to support a safe conversion between ApplicationDatatype and ImplementationDatatype, even for variable length strings as required e.g. for Support of SAE J1939.
baseType	SwBaseType	0..1	ref	This is the base type of one character in the string. In particular this baseType denotes the intended encoding of the characters in the string on level of ApplicationDataType. **Tags:** xml.sequenceOffset=30
swFillCharacter	Integer	0..1	attr	Filler character for text parameter to pad up to the maximum length swMaxTextSize. The value will be interpreted according to the encoding specified in the associated base type of the data object, e.g. 0x30 (hex) represents the ASCII character zero as filler character and 0 (dec) represents an end of string as filler character. The usage of the fill character depends on the arraySizeSemantics. **Tags:** xml.sequenceOffset=40

Attribute	Type	Mul.	Kind	Note
swMaxTextSize	Integer	1	attr	Specifies the maximum text size in characters. Note the size in bytes depends on the encoding in the corresponding baseType. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=20

Table 3.5: SwTextProps

3.3.2.2 Associative Map Data Type

[TPS_MANI_01027] Semantics of `ApplicationAssocMapDataType` ⌈ An `ApplicationAssocMapDataType` represents an associative data structure, i.e. a data structure where so-called *key*s (formalized as `ApplicationAssocMapDataType.key` that are in turn typed by an `ApplicationDataType`) are associated with *value*s (formalized as `ApplicationAssocMapDataType.value` that are also in turn typed by an `ApplicationDataType`). ⌋*(RS_MANI_00016)*

[constr_3349] Usage of `ApplicationAssocMapDataType` is limited ⌈ The usage of an `ApplicationAssocMapDataType` is limited to the context of `AdaptiveApplicationSwComponentType`s and `CompositionSwComponentType`s defined in the context of an `Executable`, i.e. such a data type shall not be used on the *AUTOSAR classic platform*. ⌋*()*

[constr_3349] is a formal approach to express that an `ImplementationDataType` of `category` `VECTOR` shall only be used on the *AUTOSAR adaptive platform*.

[TPS_MANI_01016] Category of `ApplicationAssocMapDataType` ⌈ The value `ApplicationAssocMapDataType.category` shall be set to `ASSOCIATIVE_MAP` for attribute. ⌋*(RS_MANI_00016)*

Figure 3.2 depicts an example of the structure of an `ApplicationAssocMapDataType`.

As can be deduced from looking at Figure 3.2, the concept of an `ApplicationDataType` of `category` `MAP` shall not be confused with an `ApplicationAssocMapDataType`[1].

[1] On the other hand, both concepts of a "map" are justified in their respective "community" and choosing to name one of these very different in order so reduce overall potential confusion would probably not be applicable

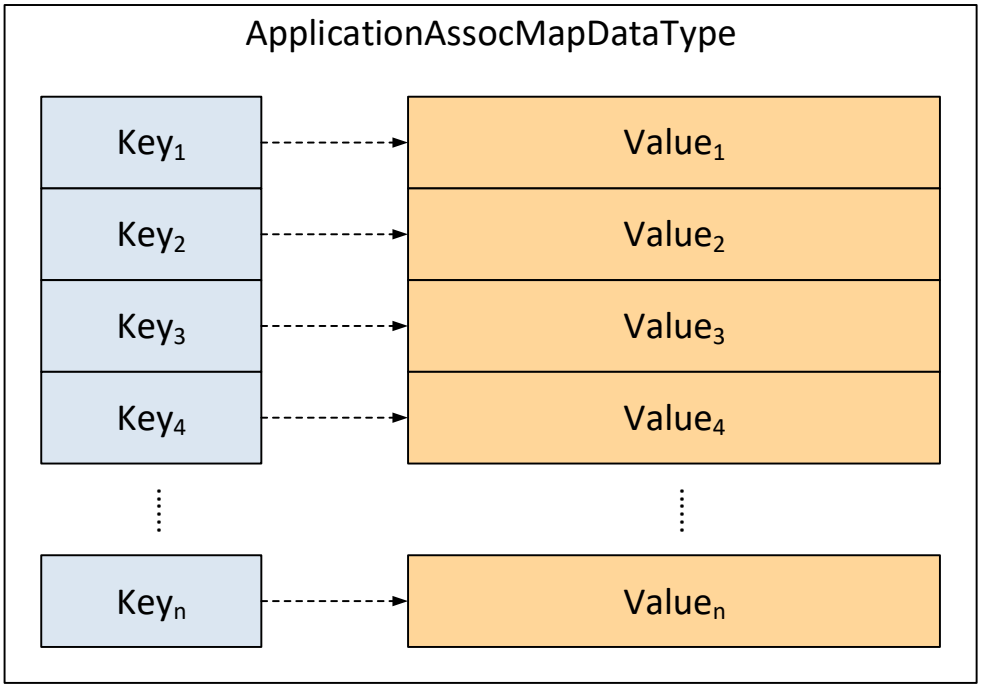

Figure 3.2: Example `ApplicationAssocMapDataType` on the *AUTOSAR adaptive platform*

There are a number of technical implications on the usage of an associative data structure at run-time, e.g. that the content of each *key* shall be unique within the context of the overall data structure.

On the other hand, it is totally no problem if content on the value-side contain duplicates, e.g. two unique `key`s are associated with `value`s that have a completely identical content.

However, these aspects have no implication on the formal model of the `ApplicationAssocMapDataType` and are therefore not considered in this document.

The modeling of the `ApplicationAssocMapDataType` is somewhat minimalistic and motivated mainly be the fact that data types for both key and value need to be defined.

There is no assumption how the structure of an implementation of an associative map may look like. For example, in C++ (which is currently the only supported language binding on the *AUTOSAR adaptive platform*) the straightforward way to use an associative map is to utilize the container `std::map` (where the implementation is opaque to the client programmer).

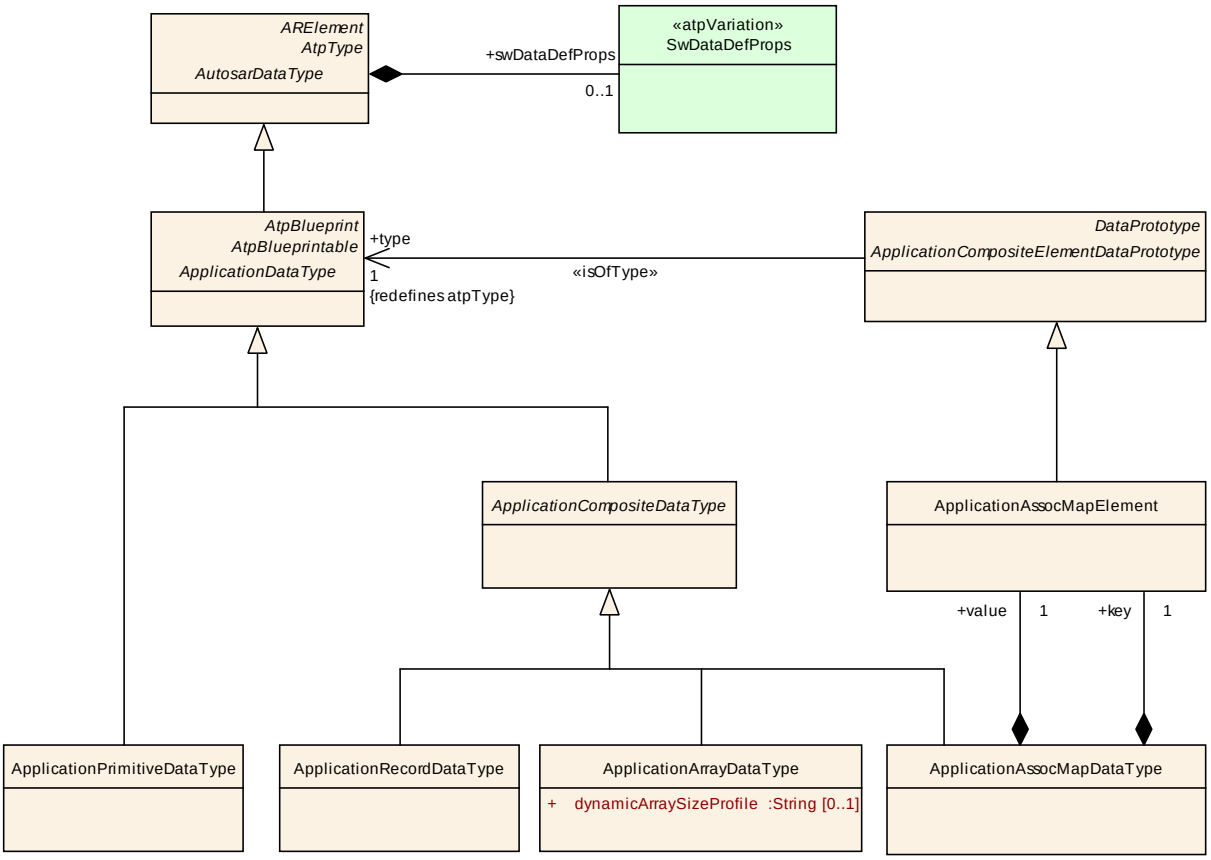

Figure 3.3: Formal model of `ApplicationAssocMapDataType`

Class	**ApplicationAssocMapDataType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DataTypes			
Note	An application data type which is a map and consists of a key and a value **Tags:** atp.Status=draft; atp.recommendedPackage=ApplicationDataTypes			
Base	ARElement, ARObject, ApplicationCompositeDataType, ApplicationDataType, Atp Blueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, Collectable Element, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
key	ApplicationAssocMapElement	1	aggr	Key element of the map that is used to uniquely identify the value of the map. **Tags:** atp.Status=draft
value	ApplicationAssocMapElement	1	aggr	Value element of the map that stores the content associated to a key. **Tags:** atp.Status=draft

Table 3.6: ApplicationAssocMapDataType

Class	**ApplicationAssocMapElement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DataTypes			
Note	Describes the properties of the elements of an application map data type. **Tags:** atp.Status=draft			
Base	ARObject, ApplicationCompositeElementDataPrototype, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.7: ApplicationAssocMapElement

Listing 3.1 provides a sketch of the modeling of an example ApplicationAssocMapDataType.

Figure 3.8 contains the corresponding graphical representation of the model.

The corresponding definition of ImplementationDataTypes can be found in Listing 3.4.

Listing 3.1: Example for the definition of an ApplicationAssocMapDataType

```
<APPLICATION-ASSOC-MAP-DATA-TYPE>
  <SHORT-NAME>MyAssociativeMap</SHORT-NAME>
  <KEY>
    <SHORT-NAME>MyKey</SHORT-NAME>
    <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">keyType</TYPE-TREF>
  </KEY>
  <VALUE>
    <SHORT-NAME>MyValue</SHORT-NAME>
    <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">valueType</TYPE-TREF>
  </VALUE>
</APPLICATION-ASSOC-MAP-DATA-TYPE>

<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>keyType</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
</APPLICATION-PRIMITIVE-DATA-TYPE>

<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>valueType</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
</APPLICATION-PRIMITIVE-DATA-TYPE>
```

The initialization of an ApplicationAssocMapDataType, however, needs to be clarified because it would (using a combination of RecordValueSpecification and ArrayValueSpecification) in general be technically possible to define a number of differently structured ValueSpecifications that are semantically identical.

In order to keep this element of uncertainty out of the AUTOSAR standard, the initialization of a DataPrototype typed by ApplicationAssocMapDataType is clarified by means of [constr_1488].

[constr_1488] Initialization of a `DataPrototype` typed by an `ApplicationAssocMapDataType` ⌈ A `DataPrototype` typed by an `ApplicationAssocMapDataType` shall only be initialized by an `ApplicationAssocMapValueSpecification`. ⌋*()*

As already mentioned, there is a semantic requirement that the *key* elements of an *associative map* need to the unique in the context of one *associative map* container.

Obviously, the model has no influence on what happens at run-time. On the other hand, there is an implication onto the initialization of an `ApplicationAssocMapDataType`, see [constr_1489].

[constr_1489] Uniqueness of `ApplicationAssocMapValueSpecification.mapElementTuple.key` ⌈ The value of all `mapElementTuple.key` elements in the context of a given `ApplicationAssocMapValueSpecification` shall be unique. ⌋*()*

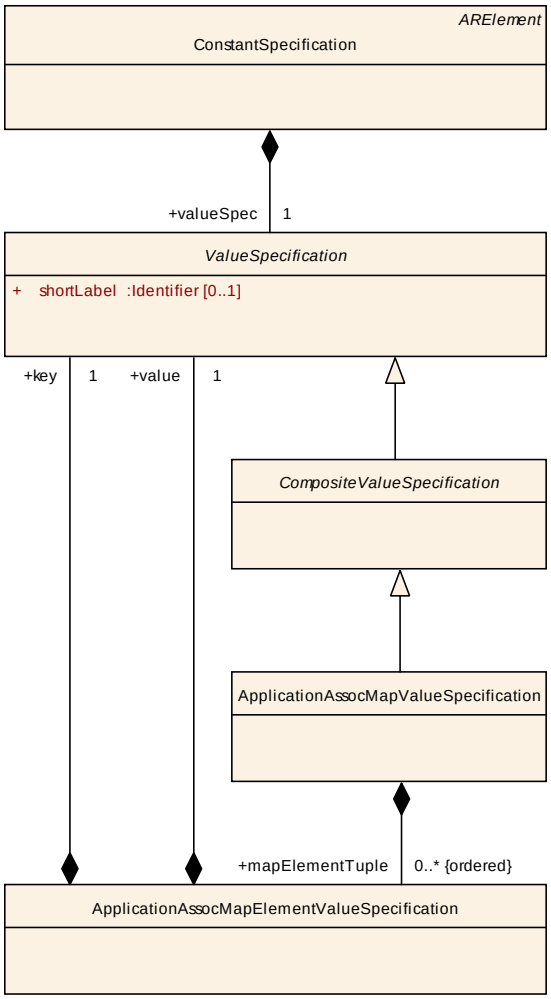

Figure 3.4: Formal model of the initialization of an `ApplicationAssocMapDataType`

Class	**ApplicationAssocMapValueSpecification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DataTypes			
Note	This meta-class represents the ability to define the initialization of an ApplicationAssocMapDataType. **Tags:** atp.Status=draft			
Base	ARObject, CompositeValueSpecification, ValueSpecification			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
mapElementTuple (ordered)	ApplicationAssocMapElementValueSpecification	*	aggr	This aggregation represents the initial values for the elements of the ApplicationAssocMapValueSpecification. **Tags:** atp.Status=draft

Table 3.8: ApplicationAssocMapValueSpecification

Class	**ApplicationAssocMapElementValueSpecification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DataTypes			
Note	This meta-class represents the ability to define the initialization of the elements of an ApplicationAssocMapDataType. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
key	ValueSpecification	1	aggr	This aggregation represents the initialization of the key part of an AssociativeElementValueSpecification. **Tags:** atp.Status=draft
value	ValueSpecification	1	aggr	This aggregation represents the initialization of the value part of an AssociativeElementValueSpecification.

Table 3.9: ApplicationAssocMapElementValueSpecification

3.3.2.3 Attributes of SwDataDefProps

[constr_1478] `SwDataDefProps` applicable to `ApplicationDataType`s exclusive to the *AUTOSAR adaptive platform* ⌈ A complete list of the `SwDataDefProps` and other attributes and their multiplicities which are allowed for a given `category` is shown in table 3.10. ⌋*()*

A consequence of [constr_1478] is that the Table 3.10 shows only the values of `category` that are limited to the *AUTOSAR adaptive platform*. For all other values of `category` that are also supported on the *AUTOSAR classic platform* please refer to a similar table contained in the specification of the Software Component Template [1].

Attributes of SwDataDefProps	Root Elem.		Attribute Existence per Category
	ApplicationAssocMapDataType	ApplicationAssocMapElement	ASSOCIATIVE_MAP
additionalNativeTypeQualifier			
annotation	x	x	*
baseType			
compuMethod			
dataConstr			
displayFormat	x	x	0..1
implementationDataType			
invalidValue			
stepSize			
swAddrMethod			
swAlignment			
swBitRepresentation			
swCalibrationAccess			
swCalprmAxisSet			
swComparisonVariable			
swDataDependency			
swHostVariable			
swImplPolicy			
swIntendedResolution			
swInterpolationMethod			
swIsVirtual			
swPointerTargetProps			
swRecordLayout			
swRefreshTiming			
swTextProps			
swValueBlockSize			
unit			
valueAxisDataType			
Other Attributes below the Root Element			
key: ApplicationAssocMapElement	x		1
value: ApplicationAssocMapElement	x		1

Table 3.10: Allowed Attributes vs. category for ApplicationDataTypes

3.3.3 ImplementationDataType

[TPS_MANI_01029] Usage of `ImplementationDataType` ⌈ A subset of the modeling of `ImplementationDataType`s that is supported on the *AUTOSAR classic platform* can directly be used on the *AUTOSAR adaptive platform* as well.

In addition to the supported values of `category` on the *AUTOSAR classic platform*, it is possible to use further values that are exclusive to the *AUTOSAR adaptive platform*. ⌋*(RS_MANI_00016)*

[constr_1479] No support for certain values of `ImplementationDataType.category` ⌈ On the *AUTOSAR adaptive platform*, the following values of `ImplementationDataType.category` are not supported:

- `DATA_REFERENCE`
- `FUNCTION_REFERENCE`

⌋*()*

For explanation of the existence of [constr_1479], the utilization of formalized data types on the *AUTOSAR adaptive platform* (currently) extends entirely to communication, there is no description of internal values as it is done extensively on the *AUTOSAR classic platform*.

The usage of pointers (which is what the mentioned two values of `category` represent) is not safe for the purpose of communication that extends potentially beyond the scope of a single process or even machine.

It should be noted that the modeling of variable-size arrays on the *AUTOSAR classic platform* has an intrinsic complexity because the programming language C that is used on the *AUTOSAR classic platform* does not provide a **native** support for variable-size arrays.

The *AUTOSAR adaptive platform*, on the other hand, supports the implementation of software using the programming language C++ [6]. This language comes with built-in so-called *container data-types*.

These container data-types are used to type **objects** (as opposed to a plain piece of data, as used in C), and this fact can be taken to significantly simplify the modeling of existing semantics that is more complex on the *AUTOSAR classic platform*, e.g. the already mentioned variable-size array can be much easier modeled with an underlying C++ `vector`.

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	***Type***	***Mul.***	***Kind***	***Note***
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
subElement (ordered)	ImplementationDataTypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of 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 3.11: ImplementationDataType

Class	**ImplementationDataTypeElement**			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. This element either consists of further subElements or it is further defined via its swDataDefProps. There are several use cases within the system of ImplementationDataTypes fur such a local declaration: • It can represent the elements of an array, defining the element type and array size • It can represent an element of a struct, defining its type • It can be the local declaration of a debug element.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
arraySize	PositiveInteger	0..1	attr	The existence of this attributes (if bigger than 0) defines the size of an array and declares that this ImplementationDataTypeElement represents the type of each single array element. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
arraySizeHandling	ArraySizeHandlingEnum	0..1	attr	The way how the size of the array is handled in case of a variable size array.
arraySizeSemantics	ArraySizeSemanticsEnum	0..1	attr	This attribute controls the meaning of the value of the array size.
subElement (ordered)	ImplementationDataTypeElement	*	aggr	Element of an array, struct, or union in case of a nested declaration (i.e. without using "typedefs"). 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
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of this ImplementationDataTypeElementt.

Table 3.12: ImplementationDataTypeElement

3.3.3.1 String Data Type

The new programming language options for implementing software on the *AUTOSAR adaptive platform* open new ways to define a string data type on the level of `Imple-`

mentationDataType that are less complex than the necessary steps that have to be taken on the *AUTOSAR classic platform*.

For more details about how strings could be used on the *AUTOSAR classic platform*, please refer to the specification of the AUTOSAR Software Component Template [1].

In addition to what is supported on the *AUTOSAR classic platform*, the *AUTOSAR adaptive platform* offers a new value of attribute category of ImplementationDataType: STRING.

[TPS_MANI_01030] ImplementationDataType of category STRING ⌈ An ImplementationDataType of category STRING represents a container data type for a sequence of characters.

AUTOSAR demands that the C++ binding of an ImplementationDataType of category STRING is always implemented by a std::string. ⌋*(RS_MANI_00016)*

It is still possible to define an encoding for a string data type according to [TPS_MANI_01030] implemented by a std::string, for any encodings other than ASCII a dedicated library to process the string content would be required.

[constr_1475] ImplementationDataType of category STRING is limited ⌈ The usage of an ImplementationDataType of category STRING is limited to the context of AdaptiveApplicationSwComponentTypes and CompositionSwComponentTypes defined in the context of an Executable. ⌋*()*

[constr_1475] is a formal approach to express that an ImplementationDataType of category STRING shall only be used on the *AUTOSAR adaptive platform*.

The example depicted in Figure 3.5 contains the definition of both an ApplicationDataType as well as the definition of the corresponding ImplementationDataType.

The latter obviously becomes significantly lighter to model thanks to the restriction that, as far as the C++ language binding is concerned, an ImplementationDataType of category STRING shall only be implemented on the basis of a std::string (as expressed by [TPS_MANI_01030]).

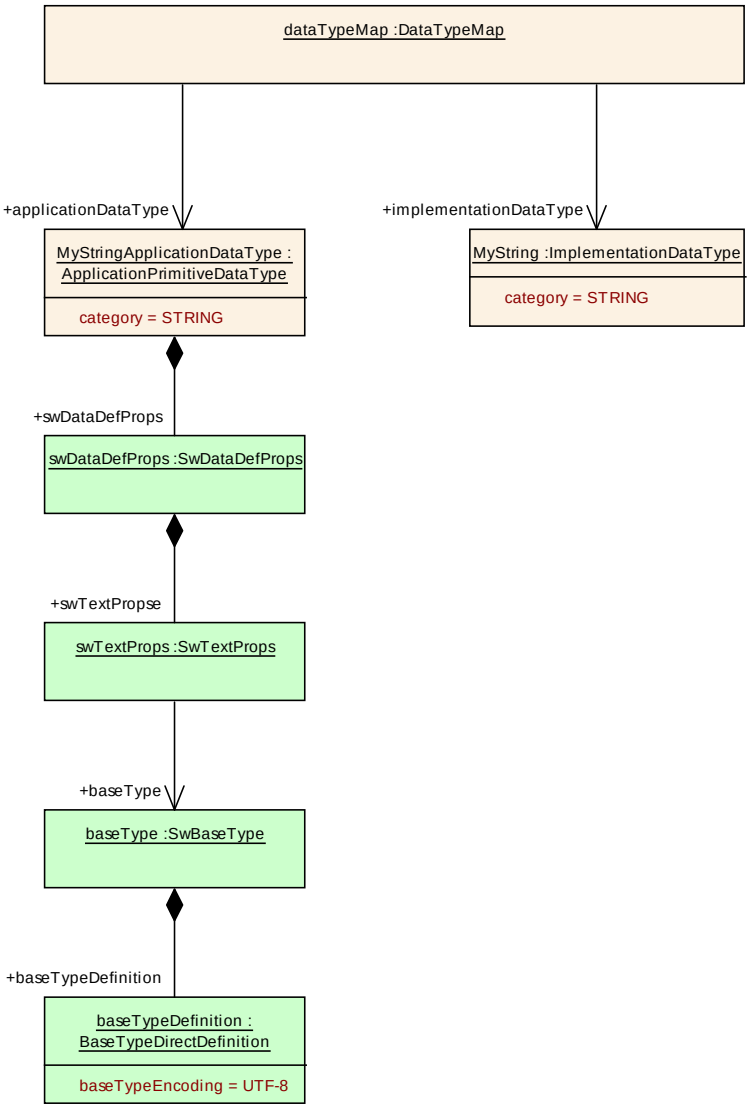

Figure 3.5: Example of the model of a string with ASCII encoding

[constr_1485] No `subElement` for `ImplementationDataType` of `category` `STRING` ⌈ `ImplementationDataType` of `category` `STRING` shall not aggregate an `ImplementationDataTypeElement` in the role `subElement`. ⌋*()*

This is also the reason why the `ImplementationDataType` does not need to refer to an `SwBaseType`. With the restriction to use `std::string` all aspects that could be clarified by the `SwBaseType` are already clarified sufficiently.

[constr_1486] `ImplementationDataType` of `category` `STRING` and `SwBaseType` ⌈ If an `ImplementationDataType` of `category` `STRING` aggregates `SwDataDefProps` in the role `swDataDefProps` then the `SwDataDefProps` shall not refer to an `SwBaseType` in the role `baseType`. ⌋*()*

Another aspect of the example is that it defines the intended encoding of the modeled data type in the scope of the `ApplicationPrimitiveDataType`.

This reflects the plausible intention of the creator of the ApplicationPrimitiveDataType to take control of the underlying encoding and not leave this decision to the corresponding model of an ImplementationDataType.

3.3.3.2 Vector Data Type

There is another case where the language binding to C++ offers new ways of implementing semantics that requires significantly more effort on the *AUTOSAR classic platform*: the so-called variable-size array.

[TPS_MANI_01018] ImplementationDataType of category VECTOR ⌈ For a C++ binding, an ImplementationDataType of category VECTOR (which can be taken as the equivalent of a variable-size array) shall always be implemented as a std::vector. ⌋*(RS_MANI_00016)*

This means that an ImplementationDataType of category VECTOR that holds any data-type other than a further ImplementationDataType of category VECTOR can be taken as the *AUTOSAR adaptive platform* equivalent of an ImplementationDataType of category STRUCT that has attribute dynamicArraySizeProfile set to the value VSA_LINEAR (see [1]).

On a related note, the companion to an ApplicationArrayDataType that does not define attribute dynamicArraySizeProfile (which means that the array data type is supposed to have a fixed size) can still be an ImplementationDataType of category ARRAY that is implemented by means of either a std::array or a C-style array in C++.

Class	**ApplicationArrayDataType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	An application data type which is an array, each element is of the same application data type. **Tags:** atp.recommendedPackage=ApplicationDataTypes			
Base	ARElement, ARObject, ApplicationCompositeDataType, ApplicationDataType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow if it is a variable size array.
element	ApplicationArrayElement	1	aggr	This association implements the concept of an array element. That is, in some cases it is necessary to be able to identify single array elements, e.g. as input values for an interpolation routine.

Table 3.13: ApplicationArrayDataType

[constr_1476] `ImplementationDataType` of `category VECTOR` is limited ⌈ The usage of an `ImplementationDataType` of `category VECTOR` is limited to the context of `AdaptiveApplicationSwComponentType`s and `CompositionSwComponentType`s defined in the context of an `Executable`. ⌋*()*

[constr_1476] is a formal approach to express that an `ImplementationDataType` of `category VECTOR` shall only be used on the *AUTOSAR adaptive platform*.

An `ImplementationDataType` of `category VECTOR` carries the intrinsic semantics that it (bar any limitations set by the used implementation of the C++ runtime) can grow indefinitely.

This technically corresponds to a setting of attribute `dynamicArraySizeProfile` to the value `VSA_FULLY_FLEXIBLE`. In other words, it would not make sense and only lead to confusion if in a concrete model the value of attribute `dynamicArraySizeProfile` would be set to anything else than the value `VSA_FULLY_FLEXIBLE`.

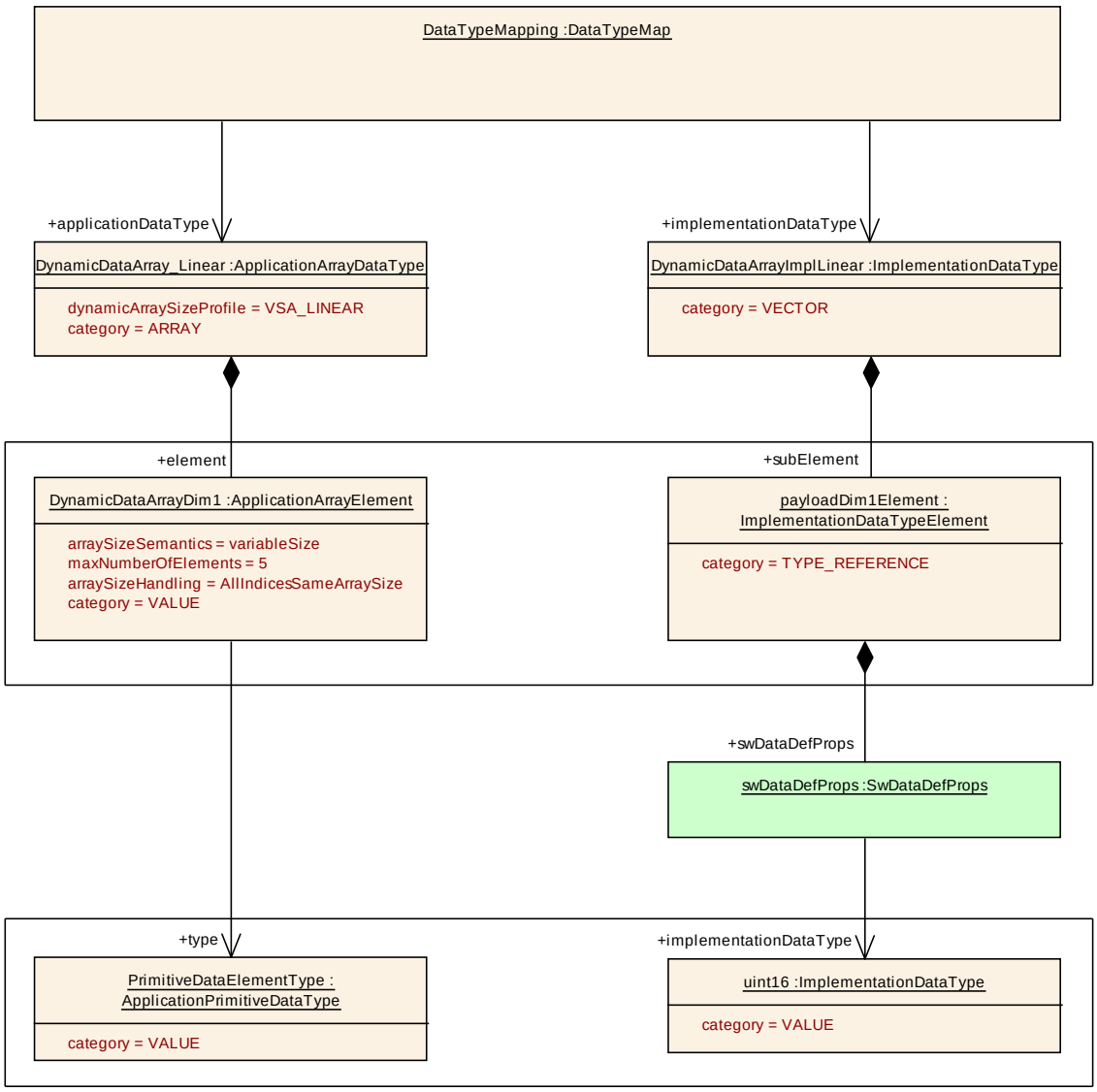

Figure 3.6: A one-dimensional vector

[constr_1506] ImplementationDataType of category VECTOR shall not define dynamicArraySizeProfile ⌈ An ImplementationDataType of category VECTOR shall **not define** attribute dynamicArraySizeProfile. ⌋*()*

In order to channel the definition of ImplementationDataType of category VECTOR the following rules shall apply:

[TPS_MANI_01042] Definition of a linear ImplementationDataType of category VECTOR ⌈ A **linear** ImplementationDataType of category VECTOR shall aggregate one ImplementationDataTypeElement which defines the details of the "payload" of the ImplementationDataType of category VECTOR. ⌋ *(RS_MANI_00016)*

Figure 3.6 contains an example model of a ImplementationDataType of category VECTOR.

For comparison, the diagram also shows the corresponding ApplicationArrayDataType that has attribute dynamicArraySizeProfile set to the value VSA_LINEAR on the left side.

A corresponding ARXML fragment can be found in Listing 3.2.

Note that the fragment represents only a sketch that concentrates on the most important model elements needed to exemplify the definition of a (semantically) **linear** ImplementationDataType of category VECTOR.

As expected, the usage of an ImplementationDataType of category VECTOR is not limited to one dimension. As a matter of fact, the full range of possible values of attribute dynamicArraySizeProfile (as explained in [TPS_SWCT_01607]) can be used.

Listing 3.2: Example for the definition of a linear ImplementationDataType of category VECTOR

```
<IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>DynamicDataArrayImplLinear</SHORT-NAME>
  <CATEGORY>VECTOR</CATEGORY>
  <SUB-ELEMENTS>
    <IMPLEMENTATION-DATA-TYPE-ELEMENT>
      <SHORT-NAME>payloadDim1Element</SHORT-NAME>
      <CATEGORY>TYPE_REEFERENCE</CATEGORY>
      <SW-DATA-DEF-PROPS>
        <SW-DATA-DEF-PROPS-VARIANTS>
          <SW-DATA-DEF-PROPS-CONDITIONAL>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">/
                ArrayExamle_VSA_Linear/uint16</IMPLEMENTATION-DATA-TYPE-REF>
          </SW-DATA-DEF-PROPS-CONDITIONAL>
        </SW-DATA-DEF-PROPS-VARIANTS>
      </SW-DATA-DEF-PROPS>
    </IMPLEMENTATION-DATA-TYPE-ELEMENT>
  </SUB-ELEMENTS>
</IMPLEMENTATION-DATA-TYPE>
```

[TPS_MANI_01043] Definition of a rectangular `ImplementationDataType` of `category VECTOR` ⌈ A (semantically) **rectangular** `ImplementationDataType` of `category VECTOR` shall have the following structure:

- The `ImplementationDataType` of `category VECTOR` shall aggregate one `ImplementationDataTypeElement` where attribute `category` is set to the value `VECTOR`.
- The `ImplementationDataTypeElement` of `category VECTOR` shall aggregate one further `ImplementationDataTypeElement` which defines the details of the "payload" of the `ImplementationDataType` of `category VECTOR`.

⌋*(RS_MANI_00016)*

Figure 3.7 contains an example model of an `ImplementationDataType` of `category VECTOR` that corresponds to [TPS_MANI_01043].

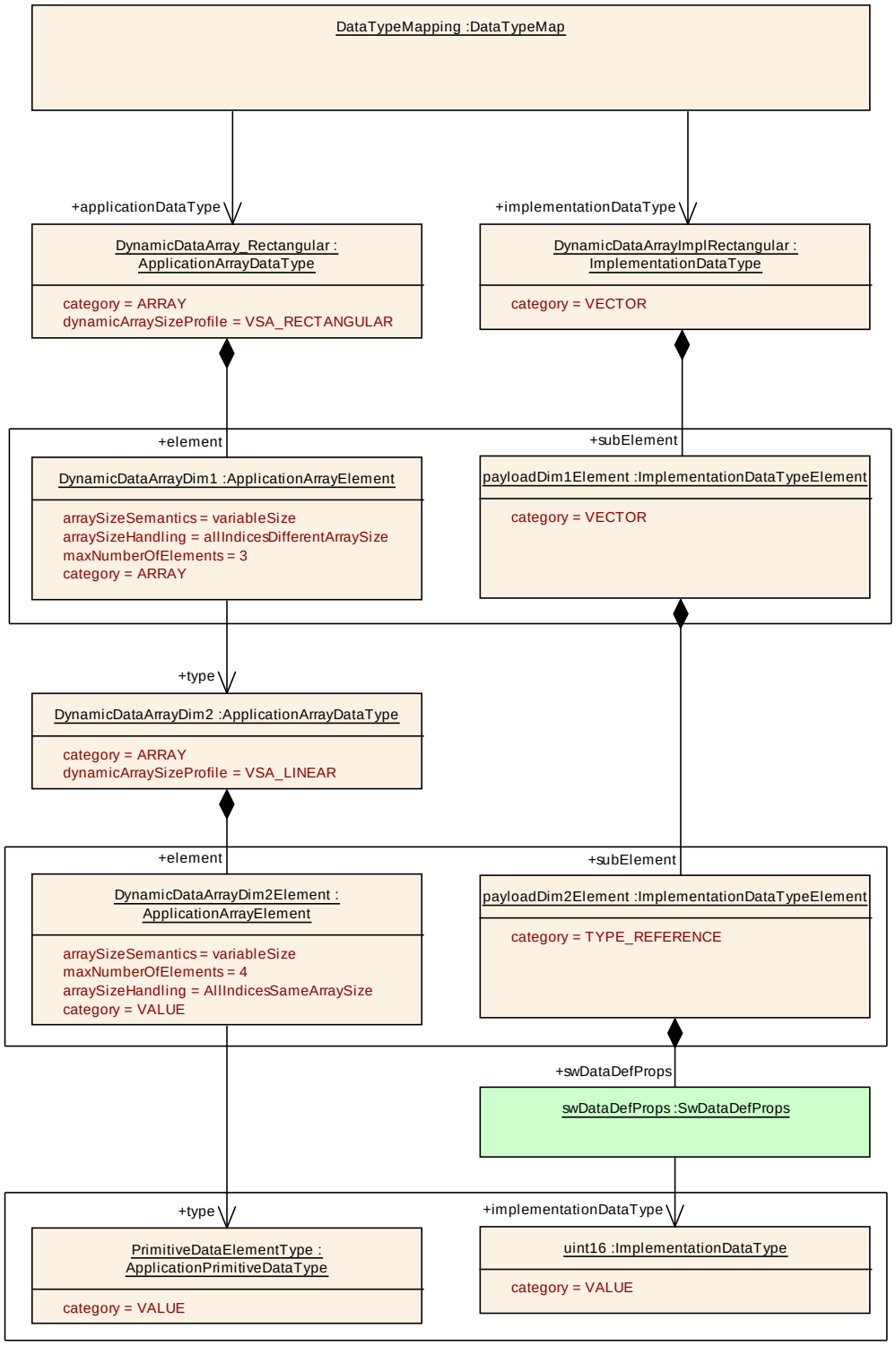

Figure 3.7: A two-dimensional vector with different dimension values

For comparison, the diagram also shows the corresponding `ApplicationArrayDataType` that has attribute `dynamicArraySizeProfile` set to the value `VSA_RECTANGULAR` on the left side.

A corresponding ARXML fragment can be found in Listing 3.3.

Note that the fragment represents only a sketch that concentrates on the most important model elements needed to exemplify the definition of a **rectangular** ImplementationDataType of category VECTOR.

Listing 3.3: Example for the definition of a rectangular ImplementationDataType of category VECTOR

```
<IMPLEMENTATION-DATA-TYPE>
    <SHORT-NAME>DynamicDataArrayImplRectangular</SHORT-NAME>
    <CATEGORY>VECTOR</CATEGORY>
    <SUB-ELEMENTS>
        <IMPLEMENTATION-DATA-TYPE-ELEMENT>
            <SHORT-NAME>payloadDim1Element</SHORT-NAME>
            <CATEGORY>VECTOR</CATEGORY>
            <SUB-ELEMENTS>
                <IMPLEMENTATION-DATA-TYPE-ELEMENT>
                    <SHORT-NAME>payloadDim2Element</SHORT-NAME>
                    <CATEGORY>TYPE_REFERENCE</CATEGORY>
                    <SW-DATA-DEF-PROPS>
                        <SW-DATA-DEF-PROPS-VARIANTS>
                            <SW-DATA-DEF-PROPS-CONDITIONAL>
                                <IMPLEMENTATION-DATA-TYPE-REF DEST="
                                    IMPLEMENTATION-DATA-TYPE">/
                                    ArrayExamle_VSA_Linear/uint16</
                                    IMPLEMENTATION-DATA-TYPE-REF>
                            </SW-DATA-DEF-PROPS-CONDITIONAL>
                        </SW-DATA-DEF-PROPS-VARIANTS>
                    </SW-DATA-DEF-PROPS>
                </IMPLEMENTATION-DATA-TYPE-ELEMENT>
            </SUB-ELEMENTS>
        </IMPLEMENTATION-DATA-TYPE-ELEMENT>
    </SUB-ELEMENTS>
</IMPLEMENTATION-DATA-TYPE>
```

3.3.3.3 Associative Map Data Type

The companion to ApplicationAssocMapDataType on the level of ImplementationDataType could in principle be modeled in various ways.

However, the rules presented in the following paragraphs have been designed to align with an implementation using an std::map on C++[2].

To support this approach a new value of category for ImplementationDataType is necessary.

Since the category value MAP is already taken it is consequently necessary to define a new value that represents the nature of an associative map data type appropriately. This value of category is defined in [TPS_MANI_01028], along with its translation into code.

[2] which is currently the only supported language binding on the *AUTOSAR adaptive platform*

[TPS_MANI_01028] ImplementationDataType of category ASSOCIATIVE_MAP ⌈ An ImplementationDataType of category ASSOCIATIVE_MAP (can be taken as the equivalent of an associative container data structure) shall always be implemented as a std::map for a C++ binding. ⌋*(RS_MANI_00016)*

[constr_1477] ImplementationDataType of category ASSOCIATIVE_MAP is limited ⌈ The usage of an ImplementationDataType of category ASSOCIATIVE_MAP is limited to the context of AdaptiveApplicationSwComponentTypes and CompositionSwComponentTypes defined in the context of an Executable, i.e. such data type shall not be used on the *AUTOSAR adaptive platform*. ⌋*()*

[constr_1477] is a formal approach to express that an ImplementationDataType of category ASSOCIATIVE_MAP shall only be used on the *AUTOSAR adaptive platform*.

The modeling of an ImplementationDataType of category ASSOCIATIVE_MAP needs to be expressive enough to allow for deriving all necessary information for the language binding.

As a design principle, container data types do not reveal their inner structure to the application programmer, and therefore there is no point in trying to regulate the modeling of such an ImplementationDataType with the goal to mock a std::map as closely as possible.

That said, the conclusion of this observation is that the regulation of the modeling of an ImplementationDataType of category ASSOCIATIVE_MAP can be as simple as possible.

Consequently, [constr_1487] as well as [TPS_MANI_01044] implement this approach.

[constr_1487] Number of subElements of an ImplementationDataType of category ASSOCIATIVE_MAP ⌈ An ImplementationDataType of category ASSOCIATIVE_MAP shall have exactly two subElements. Their semantic meaning is defined by [TPS_MANI_01044]. ⌋*()*

[TPS_MANI_01044] Structure of an ImplementationDataType of category ASSOCIATIVE_MAP ⌈ An ImplementationDataType of category ASSOCIATIVE_MAP shall have the following structure:

- The **first** ImplementationDataTypeElement aggregated by ImplementationDataType of category ASSOCIATIVE_MAP shall represent the role that corresponds to ApplicationAssocMapDataType.key and define the respective data type details.
- The **second** ImplementationDataTypeElement aggregated by ImplementationDataType of category ASSOCIATIVE_MAP shall represent the role that corresponds to ApplicationAssocMapDataType.value and define the respective data type details.

⌋*(RS_MANI_00016)*

The regulations made by [TPS_MANI_01044] are implemented in the example modeling of an `ImplementationDataType` of `category ASSOCIATIVE_MAP` that can be found in Figure 3.8.

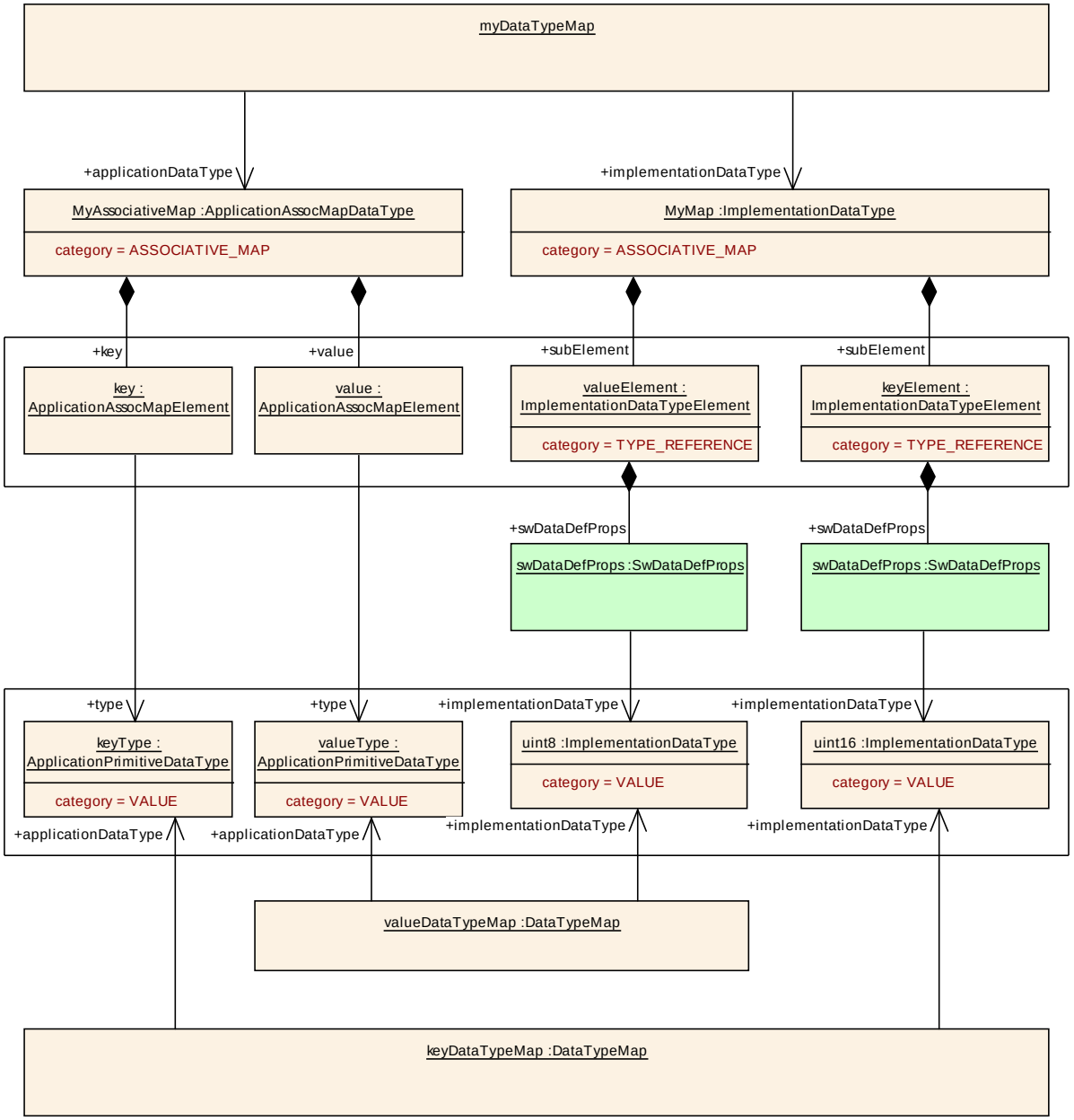

Figure 3.8: Example of the model of an associative map

The ARXML fragment listed in Listing 3.4 corresponds to the model sketched in Figure 3.8. The modeling of the corresponding `ApplicationAssocMapDataType` can be found in Listing 3.1.

Please note the order of definition of `ImplementationDataTypeElement`s in Listing 3.4.

Please note further that the fragments represents only a sketch that concentrates on the most important model elements needed to exemplify the definition of an ImplementationDataType of category ASSOCIATIVE_MAP.

This is significant for the semantics of the overall data type definition, as specified by [TPS_MANI_01044].

Listing 3.4: Example for the definition of an ImplementationDataType of category ASSOCIATIVE_MAP

```
<IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>MyMap</SHORT-NAME>
  <CATEGORY>ASSOCIATIVE_MAP</CATEGORY>
  <SUB-ELEMENTS>
    <IMPLEMENTATION-DATA-TYPE-ELEMENT>
      <SHORT-NAME>keyElement</SHORT-NAME>
      <CATEGORY>TYPE_REEFERENCE</CATEGORY>
      <SW-DATA-DEF-PROPS>
        <SW-DATA-DEF-PROPS-VARIANTS>
          <SW-DATA-DEF-PROPS-CONDITIONAL>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
              uint16</IMPLEMENTATION-DATA-TYPE-REF>
          </SW-DATA-DEF-PROPS-CONDITIONAL>
        </SW-DATA-DEF-PROPS-VARIANTS>
      </SW-DATA-DEF-PROPS>
    </IMPLEMENTATION-DATA-TYPE-ELEMENT>
    <IMPLEMENTATION-DATA-TYPE-ELEMENT>
      <SHORT-NAME>valueElement</SHORT-NAME>
      <CATEGORY>TYPE_REEFERENCE</CATEGORY>
      <SW-DATA-DEF-PROPS>
        <SW-DATA-DEF-PROPS-VARIANTS>
          <SW-DATA-DEF-PROPS-CONDITIONAL>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
              uint8</IMPLEMENTATION-DATA-TYPE-REF>
          </SW-DATA-DEF-PROPS-CONDITIONAL>
        </SW-DATA-DEF-PROPS-VARIANTS>
      </SW-DATA-DEF-PROPS>
    </IMPLEMENTATION-DATA-TYPE-ELEMENT>
  </SUB-ELEMENTS>
</IMPLEMENTATION-DATA-TYPE>

<IMPLEMENTATION-DATA-TYPE>
    <SHORT-NAME>uint16</SHORT-NAME>
    <CATEGORY>VALUE</CATEGORY>
</IMPLEMENTATION-DATA-TYPE>

<IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>uint8</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
</IMPLEMENTATION-DATA-TYPE>
```

Admittedly, the simplistic approach to modeling an ImplementationDataType of category ASSOCIATIVE_MAP also has its drawbacks.

In a clear departure from the situation on the *AUTOSAR classic platform*, the structure of such an ImplementationDataType does not reflect the structure of a Value-

Specification needed to initialize a corresponding DataPrototype, as already described in section 3.3.2.2.

Finally, the DataTypeMaps depicted in Figure 3.8 can be found in Listing 3.5.

Listing 3.5: Example for the definition of DataTypeMaps for the definition of an associative map data type

```
<DATA-TYPE-MAPPING-SET>
    <SHORT-NAME>MyDataTypeMappingSet</SHORT-NAME>
    <DATA-TYPE-MAPS>
        <DATA-TYPE-MAP>
            <APPLICATION-DATA-TYPE-REF DEST="APPLICATION-ASSOC-MAP-DATA-
                TYPE">MyAssociativeMap</APPLICATION-DATA-TYPE-REF>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
                MyMap</IMPLEMENTATION-DATA-TYPE-REF>
        </DATA-TYPE-MAP>
        <DATA-TYPE-MAP>
            <APPLICATION-DATA-TYPE-REF DEST="APPLICATION-PRIMITIVE-DATA-
                TYPE">keyType</APPLICATION-DATA-TYPE-REF>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
                uint16</IMPLEMENTATION-DATA-TYPE-REF>
        </DATA-TYPE-MAP>
        <DATA-TYPE-MAP>
            <APPLICATION-DATA-TYPE-REF DEST="APPLICATION-PRIMITIVE-DATA-
                TYPE">valueType</APPLICATION-DATA-TYPE-REF>
            <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
                uint8</IMPLEMENTATION-DATA-TYPE-REF>
        </DATA-TYPE-MAP>
    </DATA-TYPE-MAPS>
</DATA-TYPE-MAPPING-SET>
```

3.3.3.4 Attributes of SwDataDefProps

[constr_1474] SwDataDefProps applicable to ImplementationDataTypes exclusive to the *AUTOSAR adaptive platform* ⌈ A complete list of the SwDataDefProps and other attributes and their multiplicities which are allowed for a given category is shown in table 3.14. ⌋*()*

A consequence of [constr_1474] is that the Table 3.14 shows only the values of category that are limited to the *AUTOSAR adaptive platform*. For all other values of category that are also supported on the *AUTOSAR classic platform* please refer to a similar table contained in the specification of the Software Component Template [1].

Attributes of SwDataDefProps	Root Element		Attribute Existence per Category		
	ImplementationDataType	ImplementationDataTypeElement	STRING	VECTOR	ASSOCIATIVE_MAP
additionalNativeTypeQualifier					
annotation	x	x	*	*	*
baseType	x	x			
compuMethod					
dataConstr	x	x		0..1	
displayFormat	x	x	0..1	0..1	0..1
implementationDataType					
invalidValue	x	x	0..1		
stepSize					
swAddrMethod					
swAlignment					
swBitRepresentation					
swCalibrationAccess					
swCalprmAxisSet					
swComparisonVariable					
swDataDependency					
swHostVariable					
swImplPolicy					
swIntendedResolution					
swInterpolationMethod					
swIsVirtual					
swPointerTargetProps					
swPointerTargetProps.swDataDefProps					
swPointerTargetProps.functionPointerSignature					
swRecordLayout					
swRefreshTiming	x	x	0..1	0..1	0..1
swTextProps					
swValueBlockSize					
unit					
valueAxisDataType					
Other Attributes					
subElement: ImplementationDataTypeElement	x	x		1	2
subElement.arraySizeSemantics	x	x			
subElement.arraySize	x	x			

Table 3.14: Allowed Attributes vs. category for ImplementationDataType

3.3.4 BaseType

Some implications on the usage of data types only occur in the context of the SwBaseType resp. the BaseTypeDirectDefinition.

In other words, there are cases where the data types on the level of ApplicationDataType and ImplementationDataType are identical on both the *AUTOSAR adaptive platform* and the *AUTOSAR classic platform*.

Nevertheless, a different modeling is indicated on the level of the SwBaseType/BaseTypeDirectDefinition (i.e. in the binding to the actual programming language used to implement one of the respective platforms).

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

Table 3.15: SwBaseType

Class	**BaseTypeDirectDefinition**			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	ARObject, BaseTypeDefinition			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
baseTypeEncoding	BaseTypeEncodingString	1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. **Tags:** xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. **Tags:** xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. **Tags:** xml.sequenceOffset=110
maxBaseTypeSize	PositiveInteger	0..1	attr	Describes the maximum length of the BaseType in bits. **Tags:** xml.sequenceOffset=80

memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". **Tags:** xml.sequenceOffset=100

nativeDeclaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example
				
				BaseType with
				`shortName: "MyUnsignedInt"`
				`nativeDeclaration: "unsigned short"`
				
				Results in
				`typedef unsigned short MyUnsignedInt;`
				
				If the attribute is not defined the referring ImplementationDataTypes will not be generated as a typedef by RTE.
				
				If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseTypeSize.
				
				This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems.
				
				Tags: xml.sequenceOffset=120

Table 3.16: BaseTypeDirectDefinition

3.3.4.1 Bitfield

A prominent example for this kind of implication is the definition of a `nativeDeclaration` for a primitive type that implements an enumeration or a bitfield.

On the *AUTOSAR classic platform*, support for bitfields and enumeration is possible by using specific kinds of `CompuMethod`s.

However, the language C does not provide portable implementations enumerations or bitfields and thus any bitfields and enumerations can only be implemented by means of plain integer data objects.

This changes on the *AUTOSAR adaptive platform*, here it is possible to use native ways for the implementation of a bitfield, i.e. it is possible to set the value of nativeDeclaration to std::bitfield<8> for this purpose.

3.3.4.2 Enumeration

[TPS_MANI_01062] ImplementationDataType to generate a C++ enum ⌈ On the *AUTOSAR adaptive platform*, it is possible to define an ImplementationDataType that refers to a CompuMethod of category TEXTTABLE and use this ImplementationDataType to generate a native C++ enum out of it. ⌋*(RS_MANI_00016)*

[TPS_MANI_01063] Sharing of ImplementationDataType with enumeration semantics ⌈ It is possible to share an ImplementationDataType according to [TPS_MANI_01062] between the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform* if the ImplementationDataType (via SwDataDefProps) does not refer to a SwBaseType where attribute nativeDeclaration exists.

In other words, the ImplementationDataType shall be of category TYPE_REFERENCE and (via SwDataDefProps) refer to another ImplementationDataType that has a shortName that is identical with a shortName of a platform data type. ⌋*(RS_MANI_00016)*

[constr_1508] BaseTypeDirectDefinition.nativeDeclaration shall not be set to the value enum ⌈ For any given ImplementationDataType, the actual value of the attribute swDataDefProps.baseType.baseTypeDefinition.nativeDeclaration shall **not** be set to the value enum. ⌋*()*

Rationale for the existence of [constr_1508]: the attribute nativeDeclaration is needed for the specification of the integral C++ data type used for the specification of the enumeration.

Note that the usage of attribute SwDataDefProps.additionalNativeTypeQualifier is not required for achieving “native” enum semantics in the generated data type.

On the contrary, the usage of this attribute may potentially complicate the sharing of ImplementationDataTypes between the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*.

Please note further that the definition of an enum is only possible for CompuMethods that represent “pure” enumeration semantics. In cased of a “mixed” semantics (e.g. CompuMethod of category SCALE_LINEAR_AND_TEXTTABLE) it will be necessary to fall back to the generation of symbols in the source code that represent the enumerators.

The details of how an enum data type shall be generated out of the formal definition of an ImplementationDataType are explained in [7].

3.4 Service Interface

3.4.1 Overview

[TPS_MANI_01001] Meaning of `ServiceInterface` ⌈ Meta-class `ServiceInterface` inherits from `PortInterface` and allows for a heterogeneous aggregation of elements, i.e. it is possible to mix

- aggregation of `VariableDataPrototype` in the role `event` with
- aggregation of meta-class `Field` in the role `field` with
- aggregation of `ClientServerOperation` in the role `method` (with
- aggregation of `ApplicationError` in the role `possibleError`)

within the same `ServiceInterface`. ⌋*(RS_MANI_00003)*

The purpose of this modeling is to embrace the concept of service-oriented communication [3] and better support this paradigm for communication on the *AUTOSAR adaptive platform*.

Please note that, in terms of semantics, the `ApplicationError` represents sort of a second-class citizen (that only makes sense in the presence of `ClientServerOperation` in the role `method`) in the scope of the `ServiceInterface`.

More information can be found in section 3.4.4.

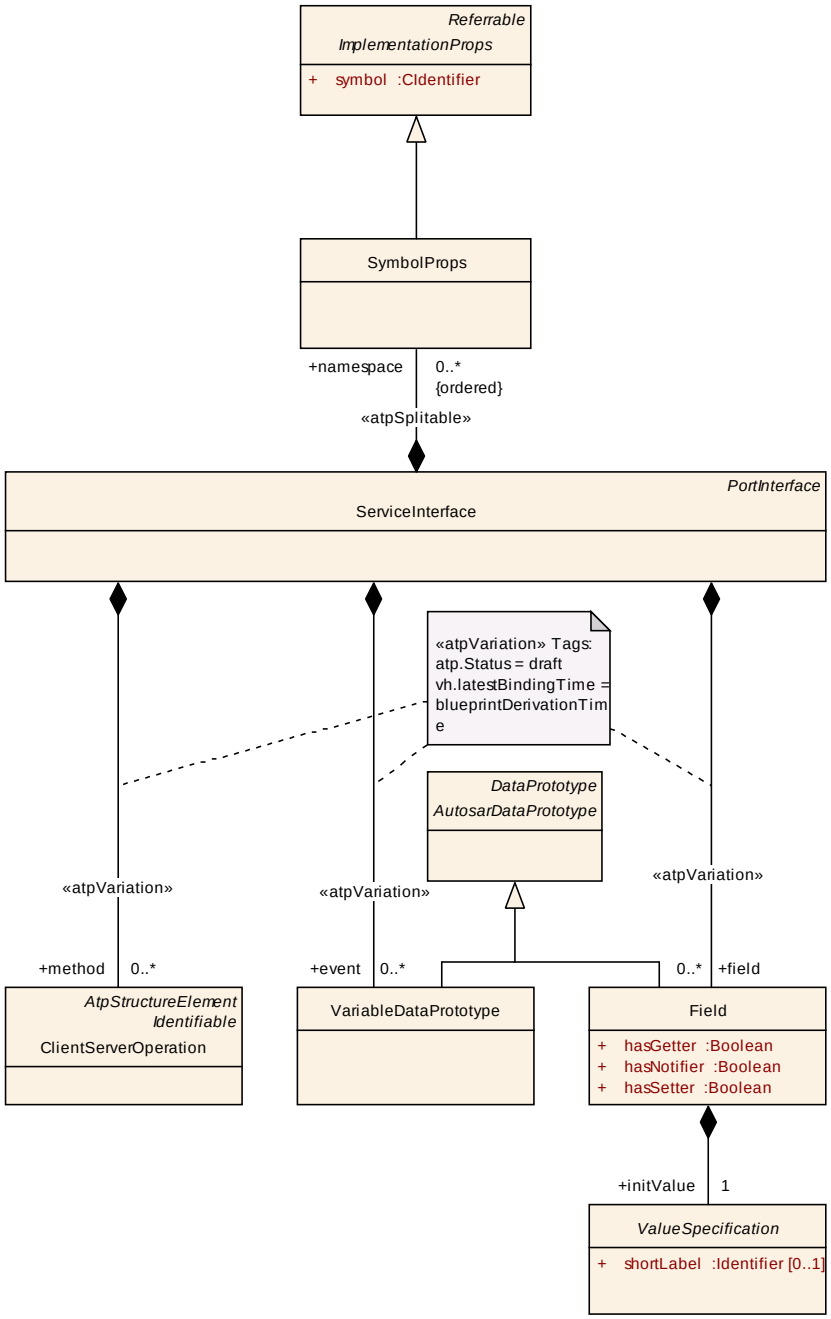

Figure 3.9: Modeling of the `ServiceInterface`

[constr_1483] Applicability of a `ServiceInterface` ⌈ The applicability of a `ServiceInterface` shall be limited to the *AUTOSAR adaptive platform*, i.e. a `ServiceInterface` shall only be taken to type a `PortPrototype` if the latter is aggregated by an `AdaptiveApplicationSwComponentType` or by a `CompositionSwComponentType` defined in the context of an `Executable`. ⌋*()*

Please note that on the *AUTOSAR adaptive platform* there are use-cases for the utilization of a `ServiceInterface` **without** the existence of a corresponding `PortPrototype`. For more explanation, please refer to [TPS_MANI_01032].

Class	**ServiceInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign			
Note	This represents the ability to define a PortInterface that consists of a heterogeneous collection of methods, events and fields. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Port Interface, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
event	VariableDataPr ototype	*	aggr	This represents the collection of events defined in the context of a ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=blueprintDerivationTime
field	Field	*	aggr	This represents the collection of fields defined in the context of a ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=blueprintDerivationTime
method	ClientServerOp eration	*	aggr	This represents the collection of methods defined in the context of a ServiceInterface. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=blueprintDerivationTime
namespac e (ordered)	SymbolProps	*	aggr	This represents the SymbolProps used for the definition of a hierarchical namespace applicable for the generation of code artifacts out of the definition of a ServiceInterface. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName; atp.Status=draft
possibleErr or	ApplicationError	*	aggr	This represents the collection of ApplicationErrors defined in the context of the enclosing ServiceInterface.

Table 3.17: ServiceInterface

[TPS_MANI_01033] Semantics of `ServiceInterface.event` ⌈ An `event` represents an update to a piece of data. The server decides when to send this update and makes sure that the `event` has full control over the value.

The occurrence of an `event` is transmitted from a server to one or more client(s). ⌋ *(RS_MANI_00003)*

[constr_1494] Initial value for `event` ⌈ An `ServiceInterface.event` shall **not** have an `initValue`. ⌋*()*

For the client, the only way to get access to the value of an `event` is to receive an update of the `event` from the server.

As mentioned in [constr_1494], the Server always has full control over the value of the `event` and when it is sent to clients. Therefore, the definition of an `initValue` is not necessary.

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	***Type***	***Mul.***	***Kind***	***Note***
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

Table 3.18: VariableDataPrototype

[TPS_MANI_01034] Semantics of `ServiceInterface.field` ⌈ A `field` represents a piece of data hosted by a server that is accessible to one or more client(s) via *get* and/or *set* accessors.

Clients can optionally receive notifications of changes of the `field`'s value. ⌋ *(RS_MANI_00003)*

[constr_1495] Initial value for `field` ⌈ A `field` shall have an `initValue`. ⌋*()*

If a `field` defines `hasGetter` = `True` then the client may access the value of the `Field` at any time and at its own discretion. It is therefore necessary that the `Field` always has a valid value because the client would have no way to distinguish an undefined from a defined value.

Class	**Field**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign			
Note	This meta-class represents the ability to define a piece of data that can be accessed with read and/or write semantics. It is also possible to generate a notification if the value of the data changes. **Tags:** atp.Status=draft			
Base	ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hasGetter	Boolean	1	attr	This attribute controls whether read access is foreseen to this field.
hasNotifier	Boolean	1	attr	This attribute controls whether a notification semantics is foreseen to this field.
hasSetter	Boolean	1	attr	This attribute controls whether write access is foreseen to this field.

initValue	ValueSpecification	1	aggr	Specifies initial value(s) of the Field.

Table 3.19: Field

[TPS_MANI_01035] Semantics of `ServiceInterface.method` ⌈ A `method` represents a function that is executed by and in the scope of a server on request of one or more client(s). ⌋*(RS_MANI_00003)*

Class	**ClientServerOperation**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An operation declared within the scope of a client/server interface.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this ClientServerOperation **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime
possibleError	ApplicationError	*	ref	Possible errors that may by raised by the referring operation.

Table 3.20: ClientServerOperation

[TPS_MANI_01007] Atomic unit of service discovery ⌈ As far as the application level is concerned, the atomic unit for **service discovery** on the *AUTOSAR adaptive platform* is the `ServiceInterface`. ⌋*(RS_MANI_00003)*

3.4.2 Compatibility of Service Interfaces

The question of compatibility of `PortInterface`s is discussed in full detail in the specification of the Software Component Template [1].

However, this compatibility consideration is based on the assumption that communication ends are associated with each other up-front (by means of `SwConnector`s) and can be analyzed offline with respect to compatibility.

On the *AUTOSAR adaptive platform*, the situation is entirely different. The actual counterparts of required and provided services are identified based on information that is contained **exclusively in the deployment** description (i.e. by means of the assignment of service instance identifiers).

Each transport layer mechanism (e.g. SOME/IP) may define own compatibility rules. Therefore for each individual transport layer an own impact assessment on the compatibility needs to be performed whether the changed service interface has an incompatible representation on this transport layer.

3.4.3 Namespace

The definition of a ServiceInterface has a direct impact on the code of an application on the *AUTOSAR adaptive platform*.

Without going into too much detail at this point, it is necessary to support the definition of a *namespace* in the context of a ServiceInterface.

The namespace shall be used to encapsulate source code related to the ServiceInterface and thus avoid name clashes with the content of other definitions of ServiceInterfaces.

In principle, the definition of the namespace around a concrete ServiceInterface could be derived from the structure of ARPackages in which the definition of the ServiceInterface is contained. However, this approach puts some constraints of the package structure.

The same ServiceInterface may be used in different projects that may or may not demand the usage of a specific *different* package structure.

This placement of the same ServiceInterface in potentially different package hierarchies would lead to the definition of different namespaces, and thus the necessity to create or generate the code representing the ServiceInterface **plus** the code that uses this definition again and again.

One way to overcome this potential issue is to attach a dedicated namespace definition to the definition of the ServiceInterface itself.

This approach is documented in Figure 3.9.

[TPS_MANI_01004] Semantics of ServiceInterface.namespace ⌈ The aggregation ServiceInterface.namespace shall be used to define the namespace to be used for the source code that corresponds to the given ServiceInterface. ⌋ *(RS_MANI_00003)*

[TPS_MANI_01005] The definition of the namespace of a ServiceInterface may follow a hierarchical pattern ⌈ The namespace of a ServiceInterface may follow a hierarchical pattern, as supported by many modern programming languages.

The separator between the elements of the hierarchical namespace definition depends on the used programming language and is not explicitly defined in the model.

The model only defines the elements of the hierarchical namespace pattern. ⌋ *(RS_MANI_00003)*

As the consequence of the ability to define a hierarchical namespace, the aggregation ServiceInterface.namespace is qualified as being ordered. This means that the order of individual elements to the collection of namespaces has a semantical relevance[3].

[3]This means that the definition of a namespace a::b is semantically different from the definition of a namespace b::a.

[TPS_MANI_01006] Ordered definition of `ServiceInterface.namespace` ⌈ In a hierarchical definition of `ServiceInterface.namespace` the order of `namespace` fragments shall be maintained in the translation of the namespace to source code.

In other words, the first `namespace` fragment shall appear first, followed by the second `namespace` fragment, and so on. ⌋*(RS_MANI_00003)*

Listing 3.6: Example for the definition of a namespace for a given `ServiceInterface`

```
<SERVICE-INTERFACE>
  <SHORT-NAME>MyServiceInterface</SHORT-NAME>
  <NAMESPACES>
    <SYMBOL-PROPS>
      <SHORT-NAME>first</SHORT-NAME>
      <SYMBOL>com</SYMBOL>
    </SYMBOL-PROPS>
    <SYMBOL-PROPS>
      <SHORT-NAME>second</SHORT-NAME>
      <SYMBOL>myCompany</SYMBOL>
    </SYMBOL-PROPS>
    <SYMBOL-PROPS>
      <SHORT-NAME>third</SHORT-NAME>
      <SYMBOL>software</SYMBOL>
    </SYMBOL-PROPS>
  </NAMESPACES>
</SERVICE-INTERFACE>
```

Class	**SymbolProps**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This meta-class represents the ability to attach with the symbol attribute a symbolic name that is conform to C language requirements to another meta-class, e.g. AtomicSwComponentType, that is a potential subject to a name clash on the level of RTE source code.			
Base	ARObject, ImplementationProps, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.21: SymbolProps

The Listing 3.6 exemplifies the statement made by [TPS_MANI_01006], i.e. the resulting name space in e.g. C++ would look like sketched in Listing 3.7.

```
1 namespace com {
2     namespace myCompany {
3         namespace software {
4
5         }
6     }
7 }
```

Listing 3.7: Resulting namespace for the example `ServiceInterface`

3.4.4 Error Handling

[TPS_MANI_01055] Semantics of `ServiceInterface.possibleError` ⌈ The `ServiceInterface` aggregates `ApplicationError` in the role `possibleError` in order to allow for the definition of application-level errors. ⌋*(RS_MANI_00003)*

Please note that [constr_1108] also applies for the possible values of `ApplicationError.errorCode` on the *AUTOSAR adaptive platform*.

[constr_1491] Reference to `ApplicationError` ⌈ A `ServiceInterface.possibleError` referenced by a given `ClientServerOperation` shall be owned by the same `ServiceInterface` that also owns the `ClientServerOperation`. ⌋*()*

One problem that the definition of `ApplicationError` by itself doesn't really solve is that the information returned back to the caller in case of an error is extremely limited.

By definition, the **caller cannot rely on the value** of `out`-arguments if an error occurs.

It is, however, considered crucial that the caller has the ability to obtain further information about the nature of an error from the call of a given `ClientServerOperation`. The existence of `ApplicationError.errorContext` fixes this problem.

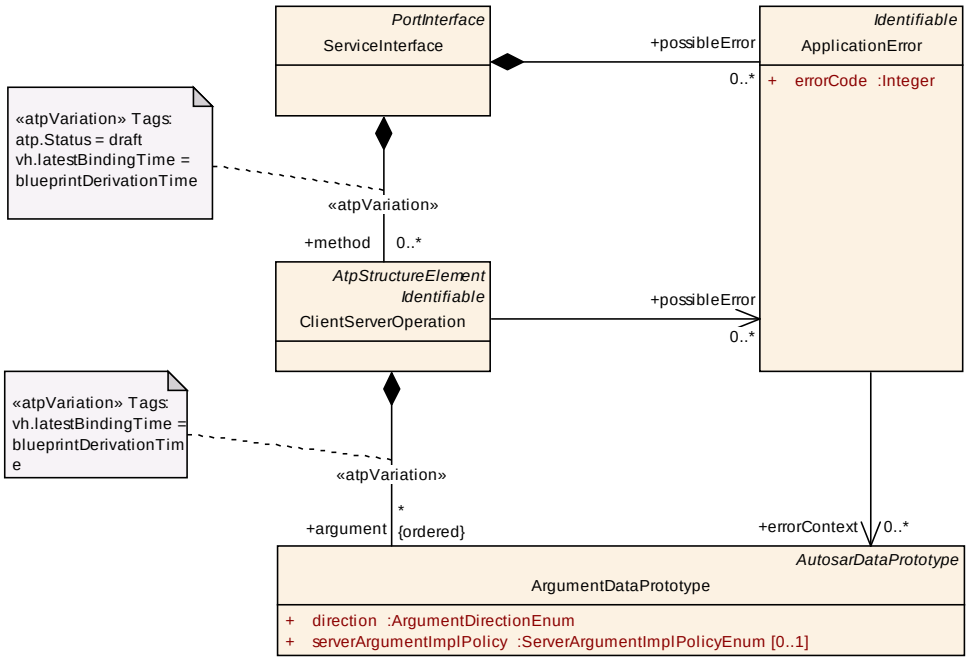

Figure 3.10: Modeling of `ApplicationError` on the *AUTOSAR adaptive platform*

By this means it is **possible to formally identify operation arguments that will have a valid value** if the call to the respective `ClientServerOperation` returns with an error indication.

[TPS_MANI_01056] Semantics of `ApplicationError.errorContext` ⌈ `ArgumentDataPrototype`s referenced in the role `ApplicationError.errorContext` are used to convey context information about a given error scenario back to the caller.

Class	**ApplicationError**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This is a user-defined error that is associated with an element of an AUTOSAR interface. It is specific for the particular functionality or service provided by the AUTOSAR software component.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
errorCode	Integer	1	attr	The RTE generator is forced to assign this value to the corresponding error symbol. Note that for error codes certain ranges are predefined (see RTE specification).
errorContext	ArgumentDataPrototype	*	ref	This reference identifies out arguments that shall have a meaning (even) if an error occurs. **Tags:** atp.Status=draft; atp.StatusComment=Reserved for AUTOSAR adaptive platform

Table 3.22: ApplicationError

Therefore, if an error occurs then `ArgumentDataPrototype`s referenced in the role `ApplicationError.errorContext` shall (in contrast to `ArgumentDataPrototype`s **not** referenced in this role) have a valid value upon termination of the execution of the associated `ClientServerOperation`. ⌋()

[constr_1493] `ArgumentDataPrototype` referenced in the role `ApplicationError.errorContext` ⌈ The reference to `ArgumentDataPrototype` in the role `ApplicationError.errorContext` is **only** supported for `ArgumentDataPrototype`s where attribute `direction` is set to `out`. ⌋()

Class	**ArgumentDataPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.			
Base	ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
direction	ArgumentDirectionEnum	1	attr	This attribute specifies the direction of the argument prototype.
serverArgumentImplPolicy	ServerArgumentImplPolicyEnum	0..1	attr	This defines how the argument type of the servers RunnableEntity is implemented. If the attribute is not defined this has the same semantics as if the attribute is set to the value useArgumentType for primitive arguments and structures and to the value useArrayBaseType for arrays.

Table 3.23: ArgumentDataPrototype

Enumeration	**ArgumentDirectionEnum**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	Use cases: • Arguments in ClientServerOperation can have different directions that need to be formally indicated because they have an impact on how the function signature looks like eventually. • Arguments in BswModuleEntry already determine a function signature, but the direction is used to specify the semantics, especially of pointer arguments.
Literal	***Description***
in	The argument value is passed to the callee. **Tags:** atp.EnumerationValue=0
inout	The argument value is passed to the callee but also passed back from the callee to the caller. **Tags:** atp.EnumerationValue=1
out	The argument value is passed from the callee to the caller. **Tags:** atp.EnumerationValue=2

Table 3.24: ArgumentDirectionEnum

3.4.5 Service Interface Data Type Mapping

An important step in the workflow of implementing software on the *AUTOSAR adaptive platform* is the creation of a code-based representation of a `ServiceInterface` to make it accessible for the application code.

This creation of a code-based representation is usually automatized and will be executed by a code generator. This code generator needs an input from the model. The main input for this purpose is obviously the definition of the `ServiceInterface` itself.

However, this is not sufficient. The designer of a `ServiceInterface` is free to use `ApplicationDataType`s for the specification of the details of the `ServiceInterface`.

It is therefore necessary to provide the definition of an `ImplementationDataType` for each of the used `ApplicationDataType`. In the meta-model, this correspondence is implemented by means of the meta-class `DataTypeMappingSet`[4].

However, from the methodological point of view it is considered inappropriate to let `ServiceInterface` directly refer to one or more `DataTypeMappingSet`(s).

[4]For more background regarding the definition and use of meta-class `DataTypeMappingSet` please refer to [1].

For clarification, this would mean that the mapping of ApplicationDataType to ImplementationDataType becomes an integral part of the definition of the ServiceInterface although the mapping itself does not really contribute to the actual semantics of the ServiceInterface.

As a consequence, the ServiceInterface would have to be updated whenever the mapping between data types changes.

But since the definition of ServiceInterfaces are usually considered very stable a frequent update for the mere purpose of acknowledging a change in the data type mapping is not acceptable.

In this concrete case, the described problem can be circumvented by the definition of a mapping class that refers to both a ServiceInterface and a DataTypeMappingSet and therefore create the correspondence without the need to update the ServiceInterface.

Although the prelude into this chapter suggests the existence of a meta-class that maps a ServiceInterface to one or more DataTypeMappingSet(s) the actual meta-model is designed with a broader focus.

In the future, there could be further kinds of PortInterfaces beside the ServiceInterface that need to fulfill the same use case.

Consequently, the name of the meta-class created for this purpose is PortInterfaceToDataTypeMapping.

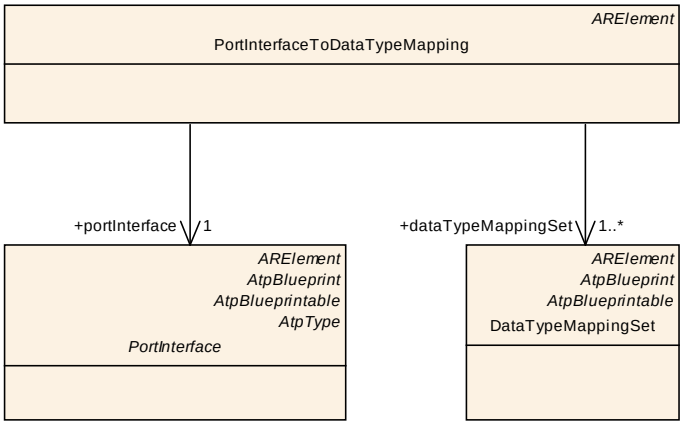

Figure 3.11: Modeling of PortInterfaceToDataTypeMapping

[constr_1507] PortInterfaceToDatatypeMapping is only applicable to ServiceInterface ⌈ PortInterfaceToDataTypeMapping.portInterface shall only refer to a ServiceInterface. ⌋*()*

Class	**PortInterfaceToDataTypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign			
Note	This meta-class represents the ability to associate a PortInterface with a DataTypeMappingSet. This association is needed for the generation of header files in the scope of a single PortInterface. The association is intentionally made outside the scope of the PortInterface itself because the designers of a PortInterface most likely will not want to add details about the level of ImplementationDataType. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInterfaceToDataType Mappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataTypeMappingSet	DataTypeMappingSet	1..*	ref	This represents the reference to the applicable dataTypemappingSet **Tags:** atp.Status=draft; atp.Status Comment=Reserved for adaptive platform
portInterface	PortInterface	1	ref	This represents the reference to the applicable PortInterface **Tags:** atp.Status=draft; atp.Status Comment=Reserved for adaptive platform

Table 3.25: PortInterfaceToDataTypeMapping

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	***Type***	***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 3.26: DataTypeMappingSet

Class	**DataTypeMap**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	This class represents the relationship between ApplicationDataType and its implementing ImplementationDataType.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
applicationDataType	ApplicationDataType	1	ref	This is the corresponding ApplicationDataType
implementationDataType	ImplementationDataType	1	ref	This is the corresponding ImplementationDataType.

Table 3.27: DataTypeMap

3.5 Service Interface Mapping

Please note that, according to [TPS_MANI_01007], the `ServiceInterface` becomes the single basis for both VFB-based and *external* (i.e. using communication networks) communication.

This concept is in stark contrast to the approach on the *AUTOSAR classic platform* where different model elements are used for the VFB-level (`PortInterface`) and the network-level (`SystemSignal`, `ISignal`, and `ISignalIPdu`).

The usage of different model elements optimally supports the existence of different granularity for VFB-based vs. network-based communication.

In other words, design of communication on the network level may be subject to different design restrictions, e.g. keep the bus load caused by service discovery manageable by defining coarse-grained communication packages.

Opposed to that, designers on the VFB level may want to define interface granularity to achieve maximum reusability.

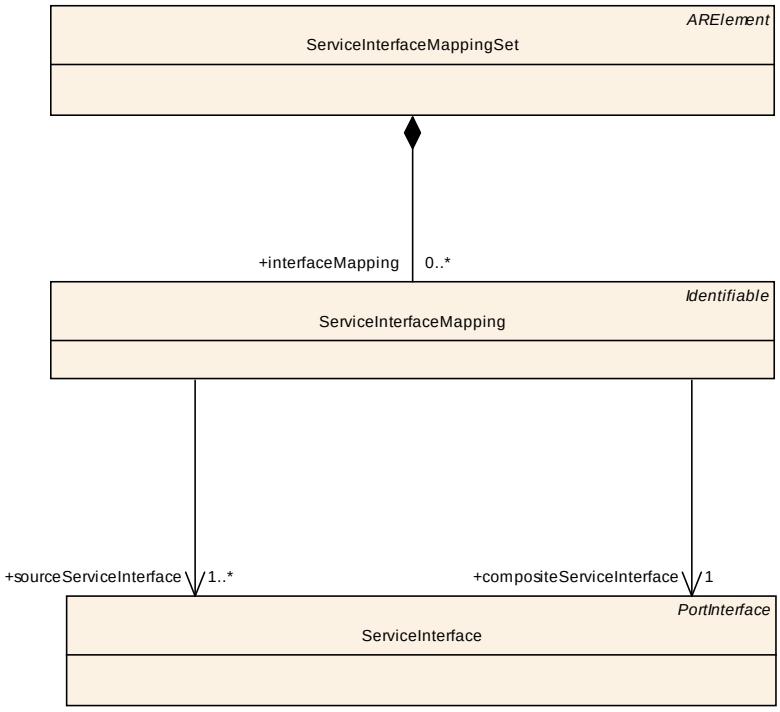

Figure 3.12: Modeling of the `ServiceInterfaceMapping`

[TPS_MANI_01002] Semantics of meta-class `ServiceInterfaceMapping` ⌈ In order to sort out a potentially different motivation between the definition of

- `ServiceInterface`s explicitly designed for VFB-based communication and
- `ServiceInterface`s explicitly designed for network-based communication

meta-class `ServiceInterfaceMapping` is available to map

- (fine-grained) `ServiceInterface`s for the VFB-communication to
- (coarse-grained) `ServiceInterface`s for network communication.

⌋*(RS_MANI_00017)*

[TPS_MANI_01032] Usage of `ServiceInterfaceMapping` ⌈ The ability to apply a `ServiceInterfaceMapping` can be used in two different ways:

- It is possible to derive a dedicated `AdaptiveApplicationSwComponentType` that implements the mapping functionality. A `SwComponentPrototype` derived from this so-called *facade* software-component would expose `PortPrototype`s for each of the `ServiceInterface`s.

  Other `SwComponentPrototype`s could then "connect" to the `PortPrototype`s typed by `ServiceInterface`s referenced in the role `sourceServiceInterface`.

  The `PortPrototype` typed by the `ServiceInterface` referenced in the role `compositeServiceInterface` is used for external communication.

- It is also possible to configure the communication middleware to offer or require a service typed by the `ServiceInterface` referenced in the role `composite-ServiceInterface`.

  A configuration of the relevant ids for this scenario is possible as part of the Application Manifest.

⌋(*RS_MANI_00017*)

Figure 3.13 summarizes the idea behind the creation of a *facade* software-component. The latter is able to "bundle" the communication of different `PortPrototype`s owned by potentially different `SwComponentType`s for external communication.

In other words, elements `event1` owned by `SWC1` and `event2` owned by `SWC1` are combined into one `ServiceInterface` used to type one `PortPrototype` of the *facade* software-component.

From the communication-related outside point-of-view, `SWC3` acts like a facade to the "inner structure" created by `SWC1` and `SWC2` that is, by way of the existence of `SWC3`, abstracted away.

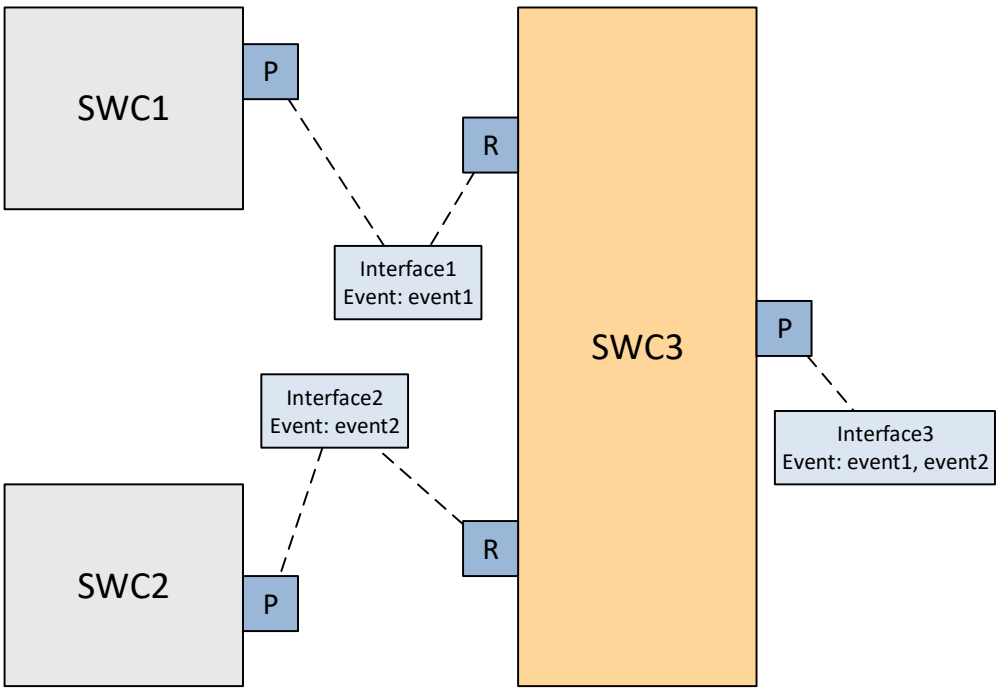

Figure 3.13: Concept of a facade software-component

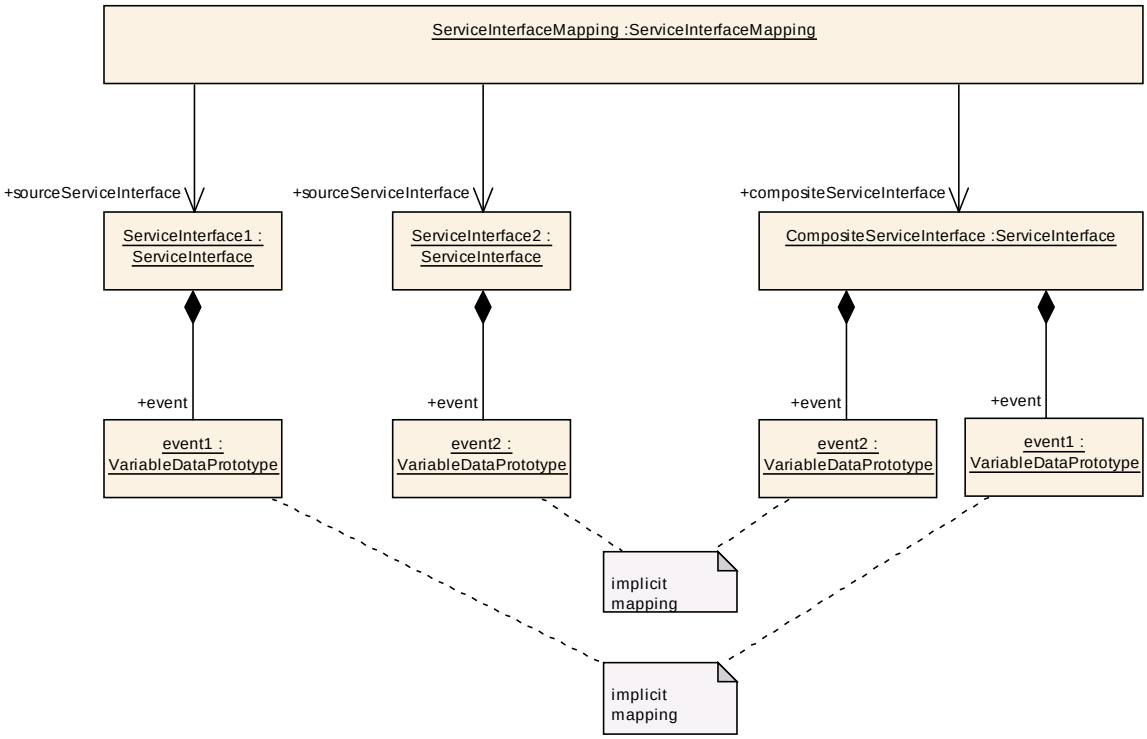

Figure 3.14: Example for the application of a `ServiceInterfaceMapping`

[TPS_MANI_01022] Concept behind `ServiceInterfaceMapping` ⌈ The concept behind the definition of a `ServiceInterfaceMapping` is that **all elements** of the `sourceServiceInterface` are required to have a **counterpart of the same kind** (`ServiceInterface.event`, `ServiceInterface.field`, or `ServiceInterface.method`) and with the identical `shortName`. ⌋*(RS_MANI_00017)*

The regulation stated in [TPS_MANI_01022] is exemplified in Figure 3.14.

Please note that the creation of a `ServiceInterfaceMapping` is considered an atomic step, it is unlikely that such a `ServiceInterfaceMapping` is partially created and then later finished by a different party.

After all, there are mutually exclusive ways to specify the mapping, and any creator of a partial mapping of `ServiceInterface`s could not be sure which of the alternatives apply for a specific pairing of one `ServiceInterface` with another without already knowing the other `ServiceInterface` (in which case the mapping can already be completed).

Therefore, there is no need to set the lower multiplicity of the references to `ServiceInterface` to 0.

Class	**ServiceInterfaceMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterface Mapping			
Note	Specifies one ServiceInterfaceMapping that allows to define that a ServiceInterface is composite of several other ServiceInterfaces. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
compositeServiceInterface	ServiceInterface	1	ref	This represents the composite ServiceInterface. **Tags:** atp.Status=draft
sourceServiceInterface	ServiceInterface	1..*	ref	ServiceInterface that is mapped into the composite ServiceInterface. **Tags:** atp.Status=draft

Table 3.28: ServiceInterfaceMapping

Class	**ServiceInterfaceMappingSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterface Mapping			
Note	This meta-class represents the ability to aggregate a collection of ServiceInterfaceElementMappings. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInterfaceMappingSets			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
elementMapping	ServiceInterfaceElementMapping	*	aggr	This represents the collection of ServiceInterfaceElementMappings aggregated at the ServiceInterfaceElementMappingSet **Tags:** atp.Status=draft
interfaceMapping	ServiceInterfaceMapping	*	aggr	This represents the collection of ServiceInterfaceMappings owned by the ServiceInterfaceMappingSet. **Tags:** atp.Status=draft

Table 3.29: ServiceInterfaceMappingSet

[TPS_MANI_01003] Limitation of the applicability of `ServiceInterfaceMapping` ⌈ The applicability of the `ServiceInterfaceMapping` is limited to cases where the `shortName`s of the elements of the `compositeServiceInterface` are **unique** in the context of the `compositeServiceInterface`. ⌋*(RS_MANI_00017)*

As already indicated, the meta-class `ServiceInterfaceMappingSet` has been defined as a container for both `ServiceInterfaceMapping`s as well as the `ServiceInterfaceElementMapping` introduced in section 3.6.

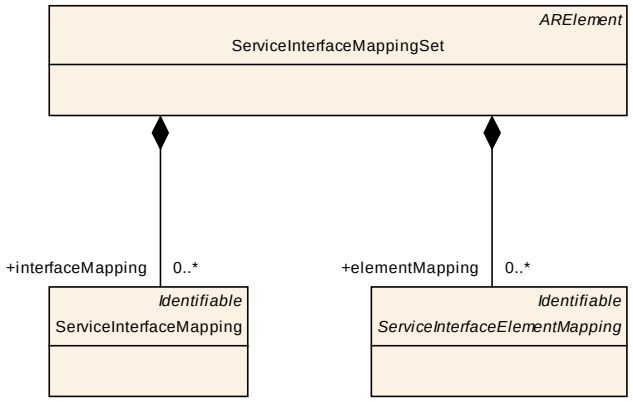

Figure 3.15: Modeling of the `ServiceInterfaceMappingSet`

Note that the `ServiceInterfaceMapping` is not an up-front association (by means of `SwConnector`s) between communication ends in the sense of section 3.4.2.

As stated in [TPS_MANI_01032], the `ServiceInterfaceMapping` allows for the derivation of a facade software-component or a proper configuration of the communication middleware.

The compatibility between the `sourceServiceInterface`s and the `compositeServiceInterface` is achieved by an adequate transformation implemented in the facade software-component or the configuration of the middleware.

Thus, connecting `ServiceInterface`s (or parts of them) via `ServiceInterfaceMapping`s is not constrained by any compatibility rules apart from the ones stated in [TPS_MANI_01022].

3.6 Service Interface Element Mapping

3.6.1 Overview

The existence of the `ServiceInterfaceMapping` leaves the question about how `ServiceInterface`s where elements have non-matching `shortName` can be mapped.

The answer to this question is provided by the ability to create an element-wise mapping of elements of the same kind.

Figure 3.16 provides an example of how such a mapping on element basis looks like. Note that, in this example, both `ServiceInterface1` and `ServiceInterface2` aggregate a `field` with the `shortName` `field1`.

This configurations disqualifies the scenario from the application of the `ServiceInterfaceMapping`, as of [TPS_MANI_01003]. The element-wise mapping, however, is able to work around the existence of the `shortName` `field1` in both "source" `ServiceInterface`s quite nicely:

- `ServiceInterface1.field1` is mapped to `CompositeServiceInterface.leftField`
- `ServiceInterface2.field1` is mapped to `CompositeServiceInterface.rightField`

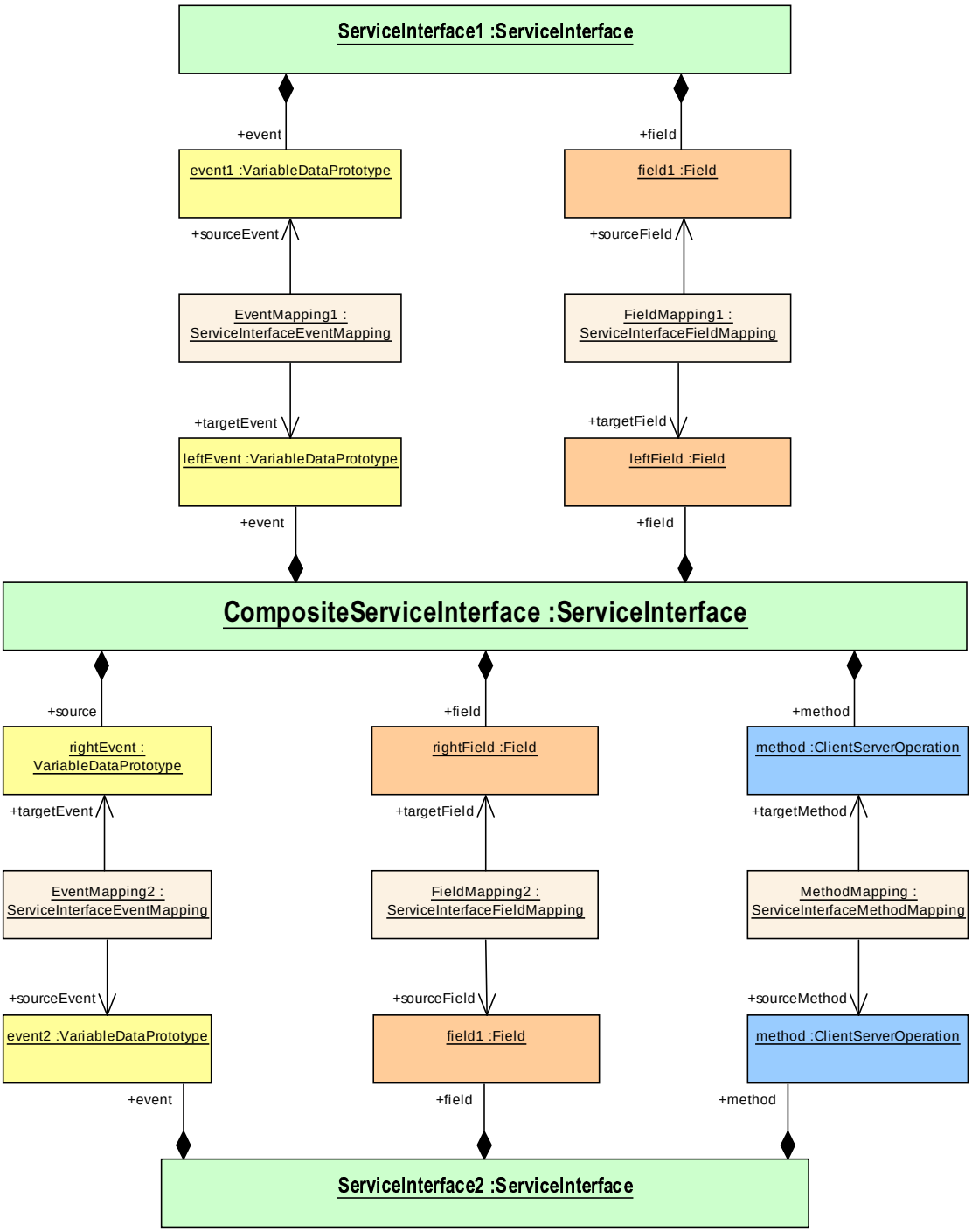

Figure 3.16: Example for a mapping of elements of `ServiceInterface`

The formal modeling of the individual mappings is described in section 3.6.

Please note that it is **not intended** to mix a mapping of ServiceInterfaces with a mapping of elements of a ServiceInterface.

In other words, as soon as a mapping between two ServiceInterfaces exists, it is not supported that a mapping between elements of the same pair of ServiceInterfaces exists. This important restriction is formalized by [constr_1482].

[constr_1482] Mapping of service interfaces vs. mapping of service interface elements ⌈ In order to establish a mapping between a given pair of ServiceInterfaces, at most **one of** the following alternatives can exist:

- the given pair of ServiceInterfaces is referenced by a ServiceInterfaceMapping, where one ServiceInterface is referenced in the role sourceServiceInterface and the other ServiceInterface is referenced in the role compositeServiceInterface.
- an arbitrary mixture of the following options exists:
  - an event aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceEventMapping in the role sourceEvent and one events aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceEventMapping in the role targetEvent.
  - a field aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceFieldMapping in the role sourceField and one fields aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceFieldMapping in the role targetField.
  - a method aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceMethodMapping in the role sourceMethod and one methods aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceMethodMapping in the role targetMethod.

⌋()

Of course, it is possible that the same ServiceInterface is referenced by mappings to elements and mappings to entire ServiceInterfaces. The limitation formalized in [constr_1482] always applies to a **pair** of ServiceInterfaces.

A mapping between elements of ServiceInterfaces is modeled by means of a subclass of the abstract meta-class ServiceInterfaceElementMapping.

Class	**ServiceInterfaceElementMapping (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterface Mapping			
Note	This abstract meta-class acts as base class for the mapping of specific elements of a ServiceInterface. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.30: ServiceInterfaceElementMapping

ServiceInterfaceElementMappings are aggregated by a ServiceInterfaceMappingSet that – in principle – allows for an arbitrary grouping of ServiceInterfaceElementMappings.

Please note that the creation of a ServiceInterfaceElementMapping is considered an atomic step, i.e. it is unlikely that such a ServiceInterfaceElementMapping is partially created, handed over to a different party and then later finished by that different party.

After all, there are mutually exclusive ways to specify the mapping, and any creator of a partial mapping of ServiceInterfaces could not be sure which of the alternatives apply for a specific pairing of one ServiceInterface with another without already knowing the other ServiceInterface (in which case the mapping can already be completed).

Therefore, there is no need to set the lower multiplicity of the references to elements of the ServiceInterface to 0.

3.6.2 Service Interface Event Mapping

[TPS_MANI_01024] Semantics of ServiceInterfaceEventMapping ⌈ Metaclass ServiceInterfaceEventMapping has the ability to map a ServiceInterface.event referenced in the role sourceEvent explicitly to another ServiceInterface.event referenced in the role targetEvent. ⌋*(RS_MANI_00017)*

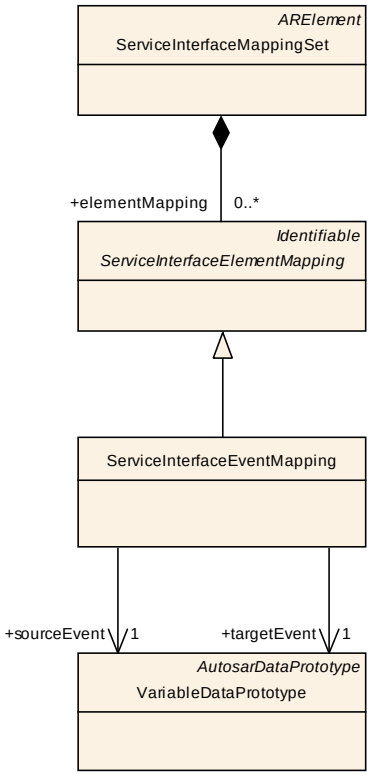

Figure 3.17: Modeling of the `ServiceInterfaceEventMapping`

Class	**ServiceInterfaceEventMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterface Mapping			
Note	This meta-class allows to define a mapping between events of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceInterfaceElement Mapping			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
sourceEvent	VariableDataPrototype	1	ref	Reference to an event that is contained in the source ServiceInterface. **Tags:** atp.Status=draft
targetEvent	VariableDataPrototype	1	ref	Reference to an event that is contained in the composite ServiceInterface. **Tags:** atp.Status=draft

Table 3.31: ServiceInterfaceEventMapping

The explicit mapping implemented by `ServiceInterfaceEventMapping` does **not** require equal `shortName`s on both sides of the mapping.

It is also possible to map a given `event` of a given `ServiceInterface` multiple times in different roles to the `ServiceInterface` that aggregates the `targetEvent`, as exemplified by Figure 3.18.

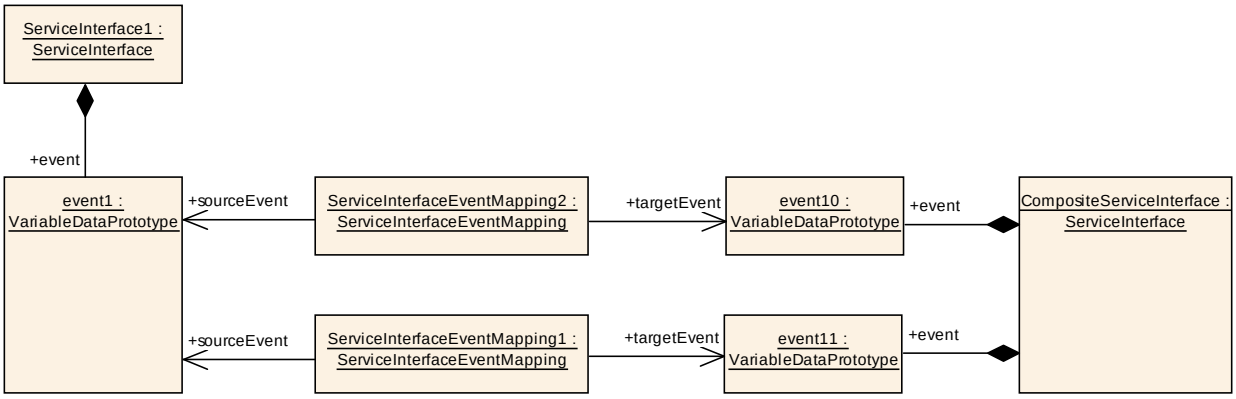

Figure 3.18: Example for the application of a `ServiceInterfaceEventMapping`

Please note that the mapping of one `sourceEvent` to different `targetEvent`s does **not** represent a *fan-out* of any kind.

It only means that the `sourceEvent` will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of `ServiceInterface`s works in Figure A.5.

3.6.3 Service Interface Field Mapping

[TPS_MANI_01025] Semantics of `ServiceInterfaceFieldMapping` ⌈ Meta-class `ServiceInterfaceFieldMapping` has the ability to map a `ServiceInterface.field` referenced in the role `sourceField` explicitly to another `ServiceInterface.field` referenced in the role `targetField`. ⌋*(RS_MANI_00017)*

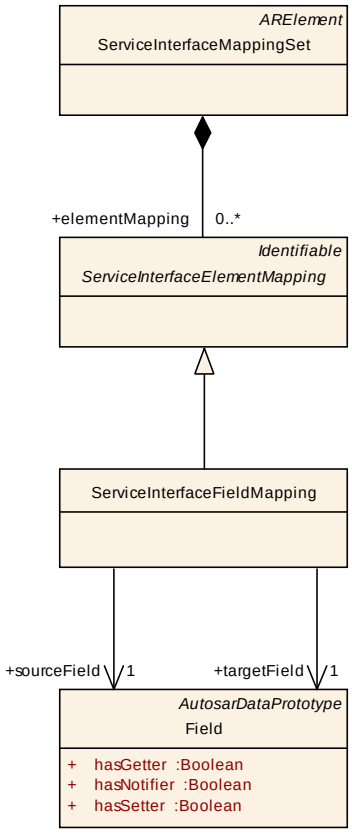

Figure 3.19: Modeling of the `ServiceInterfaceFieldMapping`

Class	**ServiceInterfaceFieldMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterface Mapping			
Note	This meta-class allows to define a mapping between fields of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceInterfaceElement Mapping			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
sourceFiel d	Field	1	ref	Reference to a field that is contained in the source ServiceInterface. **Tags:** atp.Status=draft
targetField	Field	1	ref	Reference to a field that is contained in the composite ServiceInterface. **Tags:** atp.Status=draft

Table 3.32: ServiceInterfaceFieldMapping

The explicit mapping implemented by `ServiceInterfaceFieldMapping` does **not** require equal `shortName`s on both sides of the mapping.

It is also possible to map a given `field` of a given `ServiceInterface` multiple times in different roles to the `ServiceInterface` that aggregates the `targetField`, as exemplified by Figure 3.20.

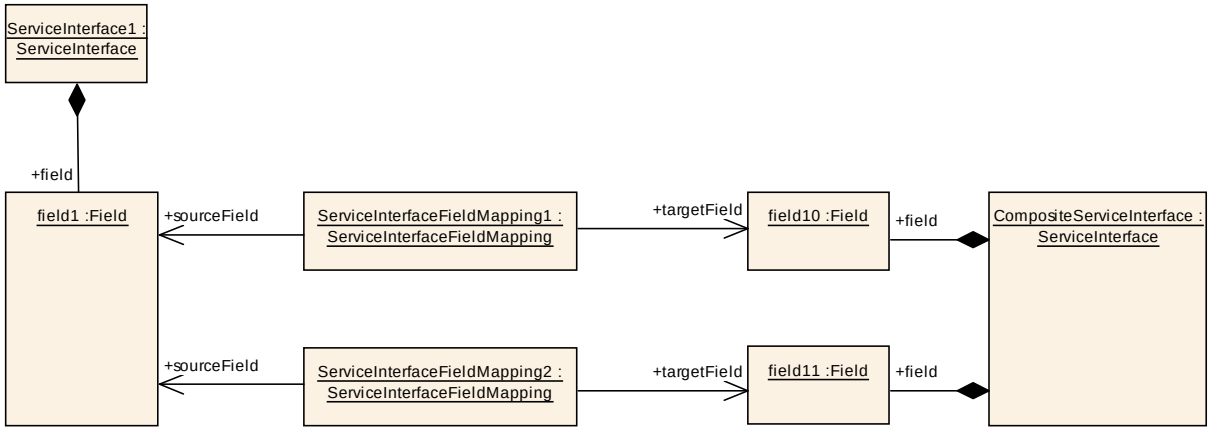

Figure 3.20: Example for the application of a `ServiceInterfaceFieldMapping`

Please note that the mapping of one `sourceField` to different `targetField`s does **not** represent a *fan-out* of any kind.

It only means that the `sourceField` will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of `ServiceInterface`s works in Figure A.5.

3.6.4 Service Interface Method Mapping

[TPS_MANI_01026] Semantics of `ServiceInterfaceMethodMapping` ⌈ Meta-class `ServiceInterfaceMethodMapping` has the ability to map a `ServiceInterface.method` referenced in the role `sourceMethod` explicitly to another `ServiceInterface.method` referenced in the role `targetMethod`. ⌋*(RS_MANI_00017)*

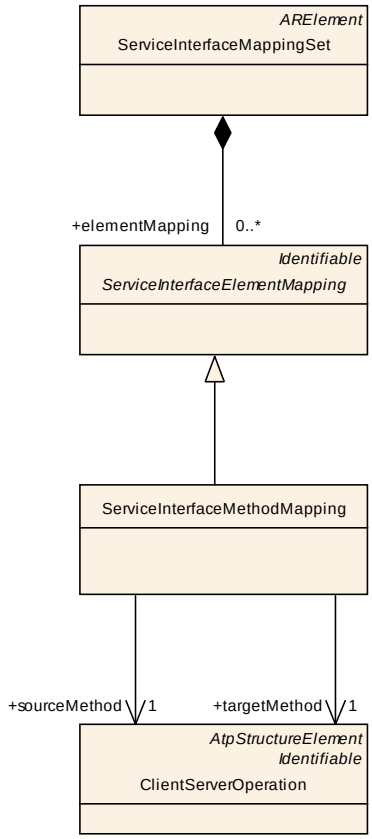

Figure 3.21: Modeling of the `ServiceInterfaceMethodMapping`

Class	**ServiceInterfaceMethodMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterface Mapping			
Note	This meta-class allows to define a mapping between methods of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceInterfaceElement Mapping			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
sourceMethod	ClientServerOperation	1	ref	Reference to a method that is contained in the source ServiceInterface. **Tags:** atp.Status=draft
targetMethod	ClientServerOperation	1	ref	Reference to a method that is contained in the composite ServiceInterface. **Tags:** atp.Status=draft

Table 3.33: ServiceInterfaceMethodMapping

The explicit mapping implemented by `ServiceInterfaceMethodMapping` does **not** require equal `shortName`s on both sides of the mapping.

It is also possible to map a given `method` of a given `ServiceInterface` multiple times in different roles to the `ServiceInterface` that aggregates the `targetMethod`, as exemplified by Figure 3.22.

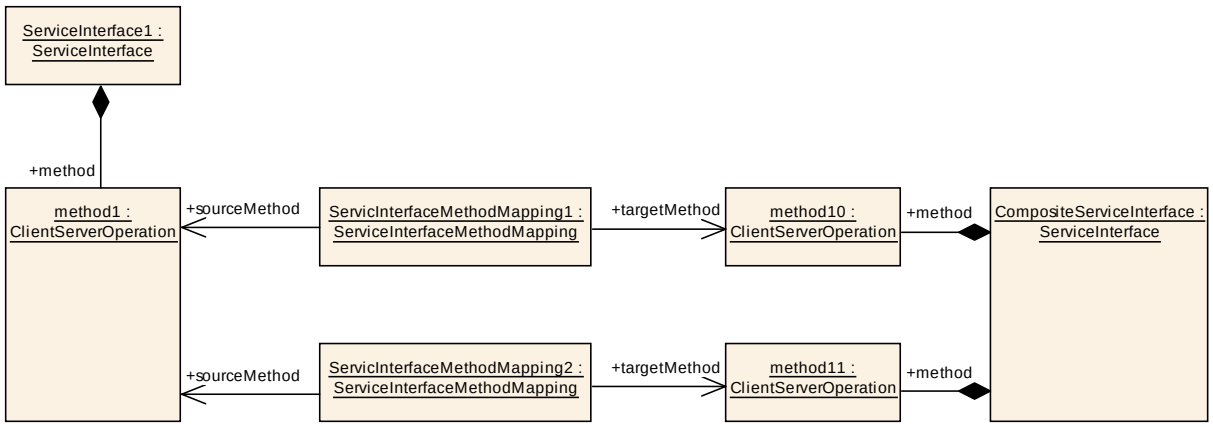

Figure 3.22: Example for the application of a `ServiceInterfaceMethodMapping`

Please note that the mapping of one `sourceMethod` to different `targetMethod`s does **not** represent a *fan-out* of any kind.

It only means that the `sourceMethod` will be used in different roles, as specified in the deployment. For more explanation, please find an example of how the role-based mapping of elements of `ServiceInterface`s works in Figure A.5.

3.6.5 Service Interface Application Error Mapping

[TPS_MANI_01058] Ability to create a mapping of `ApplicationError`s aggregated in the role `possibleError` ⌈ Apart from the “first-class citizen” of a `ServiceInterface`, i.e. `event`, `method`, and `field`, there is also the ability to create a mapping of `ApplicationError`s aggregated in the role `possibleError`. ⌋ *(RS_MANI_00017)*

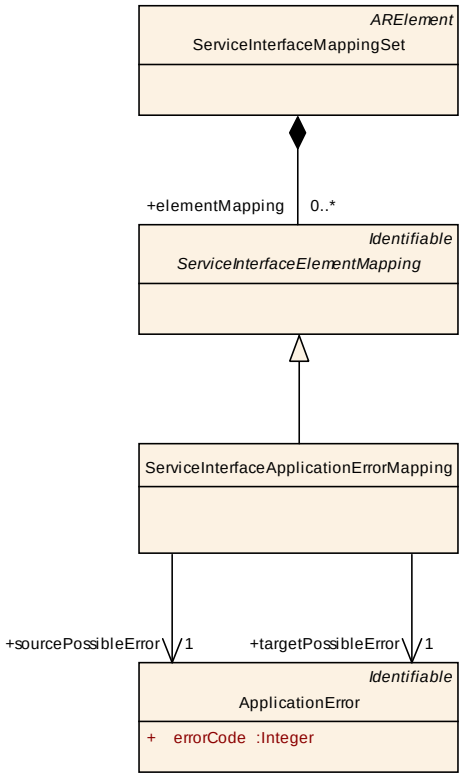

Figure 3.23: Modeling of the `ServiceInterfaceApplicationErrorMapping`

Class	**ServiceInterfaceApplicationErrorMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterface Mapping			
Note	This meta-class allows to define a mapping between possibleErrors of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceInterfaceElement Mapping			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
sourcePossibleError	ApplicationError	1	ref	This reference represents the source end of the ApplicationError mapping. **Tags:** atp.Status=draft; atp.Status Comment=Reserved for adaptive platform
targetPossibleError	ApplicationError	1	ref	This reference represents the target end of the ApplicationError mapping **Tags:** atp.Status=draft; atp.Status Comment=Reserved for adaptive platform

Table 3.34: ServiceInterfaceApplicationErrorMapping

3.7 Communication Endpoint for Application

3.7.1 Overview

As mentioned in [TPS_MANI_01007] the ServiceInterface is a model element related to service communication.

More specifically, it is used to define a *class* of service communication that can be used as a *type* for the creation of concrete service communication endpoints.

In other words, it is possible to define model-elements that represent a concrete service communication endpoint for service communication, typed by a ServiceInterface.

This concrete model-element can directly be taken over from the *AUTOSAR classic platform*: the PortPrototype. Specifically, the *AUTOSAR adaptive platform* utilizes PPortPrototype as well as the corresponding RPortPrototype.

The service-oriented communication does **not** support the concept of a communication endpoint that is both required and provided. This motivates the existence of [constr_1473].

[constr_1473] No support for PRPortPrototype ⌈ A ServiceInterface shall not be referenced by a PRPortPrototype in the role providedRequiredInterface. ⌋*()*

[TPS_MANI_01039] Representation of provided service ⌈ A **provided service** shall be modeled by means of an PPortPrototype that is typed by a ServiceInterface. ⌋*(RS_MANI_00002)*

[TPS_MANI_01040] Representation of required service ⌈ A **required service** shall be modeled by means of an RPortPrototype that is typed by a ServiceInterface. ⌋*(RS_MANI_00002)*

For more background regarding the rationale of [constr_1473], please refer to [1].

Please note that the utilization of service discovery on the *AUTOSAR adaptive platform* means that opposite communication ends **are by design not known upfront**.

As a consequence, it is in general not possible to use AssemblySwConnectors to model a pre-defined relation between two communication endpoints modeled as PortPrototypes.

Independent of the issue described above, it is still necessary to provide means for configuration of a given PortPrototype on different levels:

- The PortPrototype itself (i.e. as a whole) may need to be customized, independently of the kind or number of elements aggregated by the corresponding ServiceInterface. This aspect is discussed in section 3.7.2.
- The usage of elements of the corresponding ServiceInterface may need to be configured for a given PortPrototype. This aspect is discussed in section 3.7.3.

3.7.2 Port Prototype Props

As mentioned before, in some cases a qualification of the semantics of PortPrototypes is necessary. For this purpose, AUTOSAR typically defines a *props* class of some kind. The same approach applies in this situation as well.

In particular, PortPrototype aggregates the abstract meta-class PortPrototypeProps, that in turn starts an inheritance tree of derived meta-classes that have the ability to qualify sub-classes of PortPrototype accordingly.

One example for this approach is the definition of the meta-class RPortPrototypeProps, sketched in Figure 3.24.

[constr_3359] RPortPrototypeProps are related only to RPortPrototypes ⌈ The RPortPrototypeProps shall be aggregated only by a RPortPrototype in the role portPrototypeProps. ⌋*()*

[TPS_MANI_01052] Semantics of RPortPrototypeProps.portInstantiationBehavior ⌈ The attribute RPortPrototypeProps.portInstantiationBehavior adds the ability to define whether a given RPortPrototype can have a "multiple-instantiation semantics".

This means that the RPortPrototype exists only as a single model-element but can have a collection-semantics in the implementation of the software-component. ⌋ *(RS_MANI_00002)*

[TPS_MANI_01057] Semantics of RPortPrototypeProps.searchBehavior ⌈ The value of the attribute RPortPrototypeProps.searchBehavior clarifies whether the search for a corresponding offer shall be done as a search for “any” or else as a search for a specific ID.

Typically, a search for “any” results in a collection of offers while the search for a given id results in just a single offer. ⌋*(RS_MANI_00002)*

Please note that a search for “any” does not necessarily mean that [TPS_MANI_01052] applies, i.e. that the RPortPrototype is supposed to assume array semantics.

Even if a search for “any” is executed it may still be intended to select just a **single offer** from the result of the search. Therefore, the simultaneous existence of RPortPrototypeProps.searchBehavior and RPortPrototypeProps.portInstantiationBehavior is warranted.

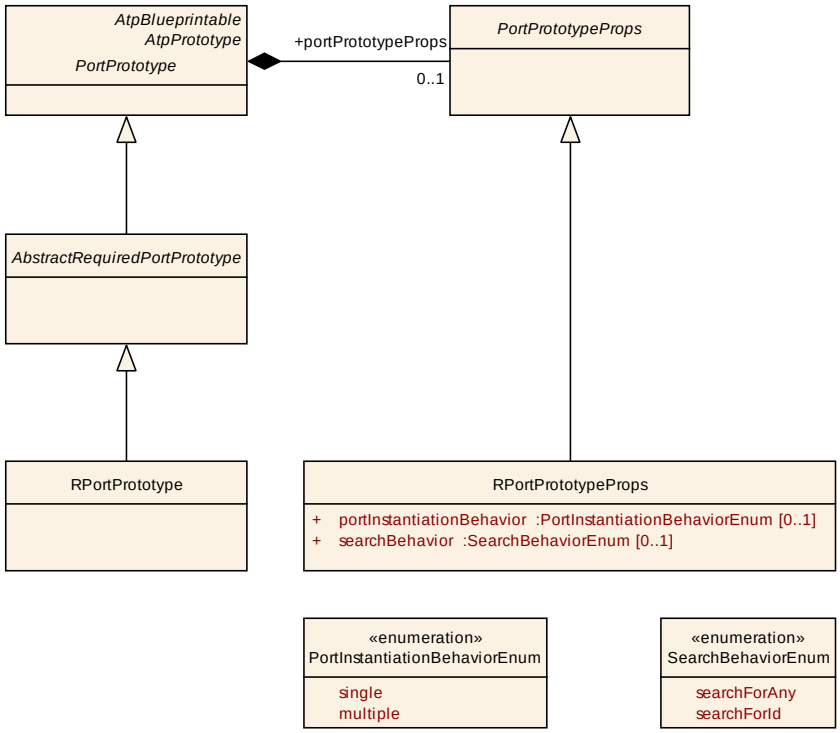

Figure 3.24: Modeling of the `RPortPrototypeProps` for `RPortPrototype`

Class	**PortPrototypeProps (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign			
Note	This meta-class represents the ability to define a further qualification of semantics of sub-classes of PortPrototype. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.35: PortPrototypeProps

Class	**RPortPrototypeProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign			
Note	PortPrototypeProps for a RPort. **Tags:** atp.Status=draft			
Base	ARObject, PortPrototypeProps			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
portInstantiationBehavior	PortInstantiationBehaviorEnum	0..1	attr	This attribute specifies how many proxy instances may be created at this RPort.
searchBehavior	SearchBehaviorEnum	0..1	attr	This attribute is used to specify the search behavior.

Table 3.36: RPortPrototypeProps

Enumeration	**PortInstantiationBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign
Note	This enumeration describes different option for the instantiation behavior of a PortPrototype. **Tags:** atp.Status=draft
Literal	***Description***
multiple	Multiple proxy instances may be created at this port. **Tags:** atp.EnumerationValue=1
single	A single proxy instance is created at this port **Tags:** atp.EnumerationValue=0

Table 3.37: PortInstantiationBehaviorEnum

Enumeration	**SearchBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign
Note	This meta-class allows for the definition of a dedicated search behavior from the application's point of view. **Tags:** atp.Status=draft
Literal	***Description***
searchFor Any	This value represents the intention to search for "any" **Tags:** atp.EnumerationValue=0
searchForId	This value represents the intention to search for a dedicated Id. **Tags:** atp.EnumerationValue=1

Table 3.38: SearchBehaviorEnum

3.7.3 Port Prototype ComSpec

[TPS_MANI_01053] Usage of `ComSpec`s on the *AUTOSAR adaptive platform* ⌈ The aspect of further qualification of elements of the `ServiceInterface` used to type given `PortPrototype` is implemented by means of `ComSpec`s, i.e. specific sub-classes of the abstract meta-classes `RPortComSpec` and `PPortComSpec`.

However, the support for `ComSpec`s on the *AUTOSAR adaptive platform* only covers a **limited selection** of attributes of a specific `ComSpec`. ⌋*(RS_MANI_00002)*

The details about supported attributes of either a `RPortComSpec` or `PPortComSpec` are described in this chapter.

3.7.3.1 Queue Length

It is necessary to provide means to configure the queue length of the reception of an event on a case-by-case basis. In other words, even two “adjacent” events within the same RPortPrototype may need a different handling of the queue length.

[TPS_MANI_01054] Definition of the queue length of an event ⌈ The definition of the queue length of an event shall be modeled by means of the attribute QueuedReceiverComSpec.queueLength. ⌋()

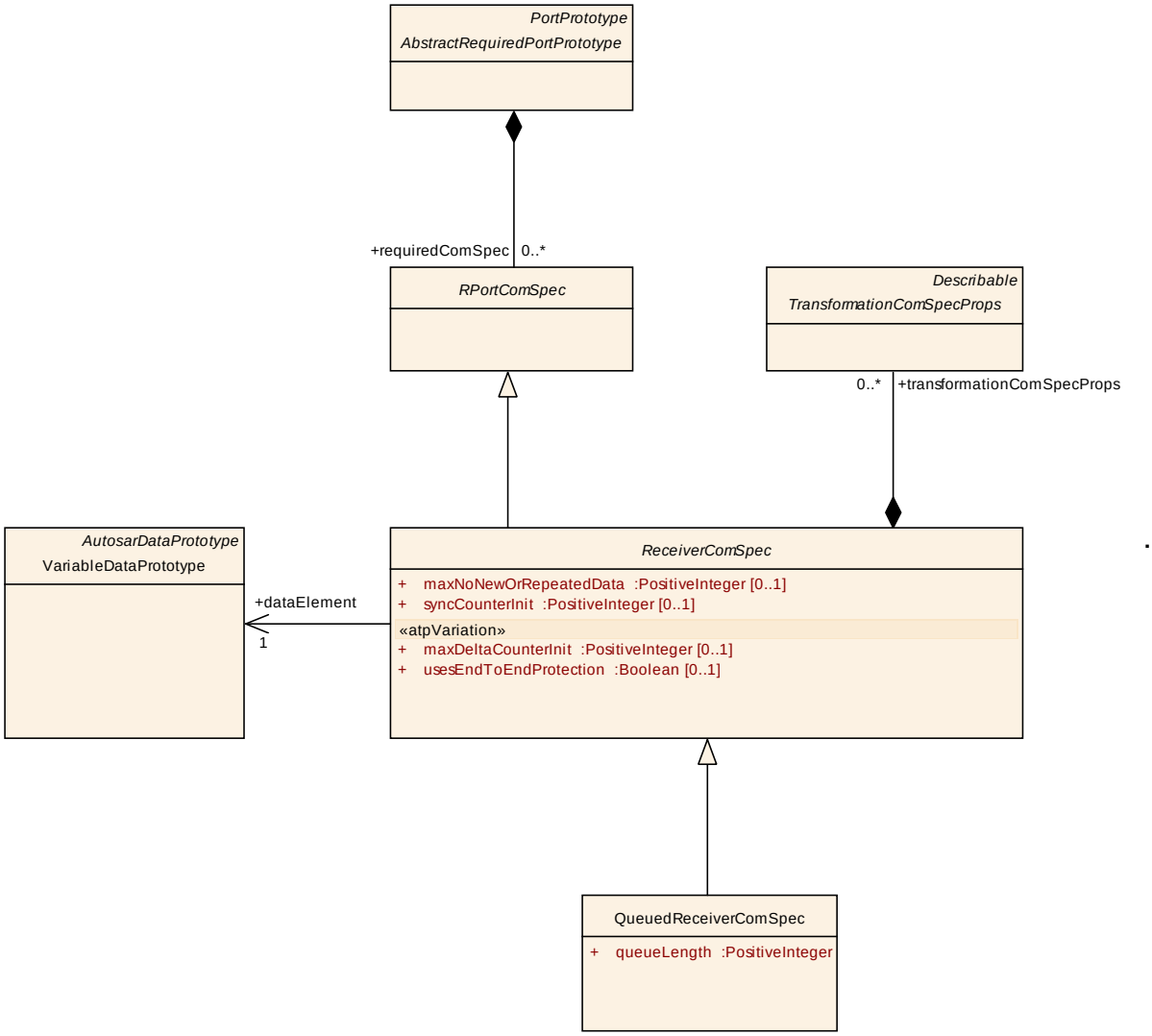

Figure 3.25: Modeling of the queueLength on the *AUTOSAR adaptive platform*

Class	**ReceiverComSpec (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Receiver-specific communication attributes (RPortPrototype typed by SenderReceiverInterface).			
Base	ARObject, RPortComSpec			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

dataElement	VariableDataPrototype	1	ref	Data element these attributes belong to.
maxDeltaCounterInit	PositiveInteger	0..1	attr	Initial maximum allowed gap between two counter values of two consecutively received valid Data, i.e. how many subsequent lost data is accepted. For example, if the receiver gets Data with counter 1 and MaxDeltaCounterInit is 1, then at the next reception the receiver can accept Counters with values 2 and 3, but not 4. Note that if the receiver does not receive new Data at a consecutive read, then the receiver increments the tolerance by 1. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
maxNoNewOrRepeatedData	PositiveInteger	0..1	attr	The maximum amount of missing or repeated Data which the receiver does not expect to exceed under normal communication conditions.
syncCounterInit	PositiveInteger	0..1	attr	Number of Data required for validating the consistency of the counter that shall be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter.
transformationComSpecProps	TransformationComSpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.
usesEndToEndProtection	Boolean	0..1	attr	This indicates whether the corresponding dataElement shall be transmitted using end-to-end protection. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table 3.39: ReceiverComSpec

Class	**QueuedReceiverComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to queued receiving.			
Base	ARObject, RPortComSpec, ReceiverComSpec			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
queueLength	PositiveInteger	1	attr	Length of queue for received events.

Table 3.40: QueuedReceiverComSpec

3.7.4 Transport Layer Independent InstanceId

The usage of a PortPrototype is one way to define a service communication endpoint as described in chapter 3.7.1. But as an alternative there is also the possibility to define a communication endpoint without the usage of software-components and PortPrototypes.

The TransportLayerIndependentInstanceId represents to some extent a "port without the software-component". For this reason it contains similar configuration properties as a PortPrototype and therefore aggregates the PortPrototypeProps element as well.

Such a transport layer independent TransportLayerIndependentInstanceId can be used in the ara::com proxy class resp. skeleton class generation to define the Instance Identifier.

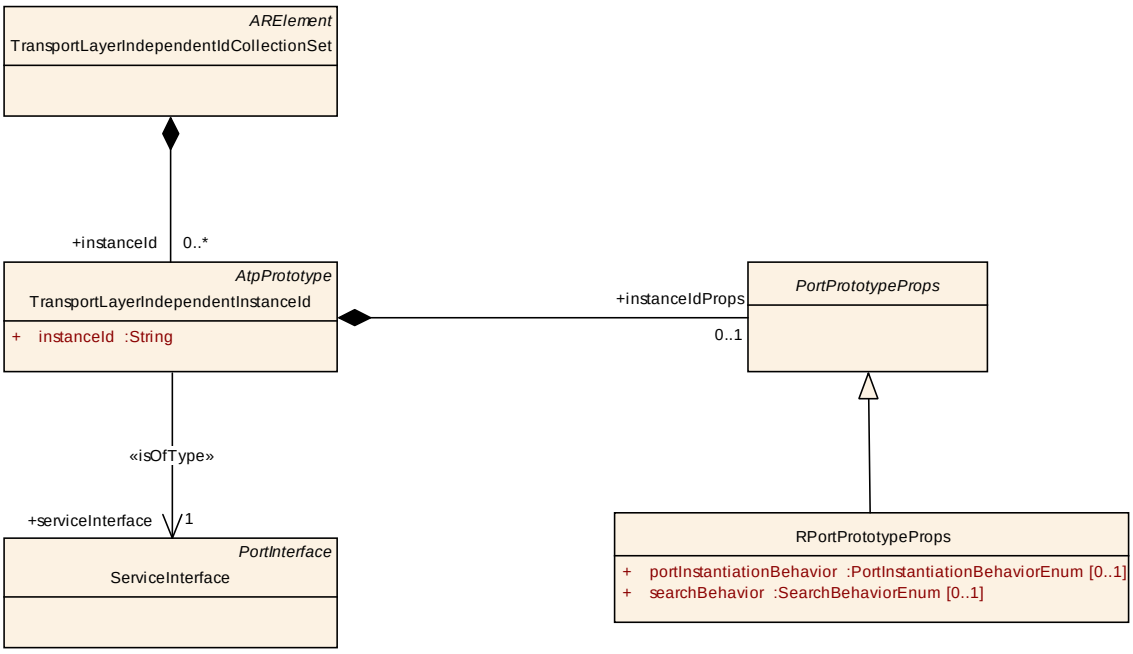

Figure 3.26: Transport Layer Independent InstanceId

[TPS_MANI_03100] Transport layer independent TransportLayerIndependentInstanceIds ⌈ The TransportLayerIndependentInstanceId represents a Service Instance of the ServiceInterface that is referenced in the role serviceInterface. ⌋*(RS_MANI_00011)*

[constr_3358] Usage of PortPrototype and TransportLayerIndependentInstanceId to define the same Service Instance is not allowed ⌈ A provided or required service shall be modeled either by a PortPrototype as defined by [TPS_MANI_01039] and [TPS_MANI_01040] or by a TransportLayerIndependentInstanceId. ⌋*()*

Class	**TransportLayerIndependentIdCollectionSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransportLayerIndependentServiceInstanceId			
Note	Collection of transport layer independent ServiceInstanceIds. **Tags:** atp.Status=draft; atp.recommendedPackage=TransportLayerIndependentIdCollectionSets			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
instanceId	TransportLayerIndependentInstanceId	*	aggr	Transport Layer independent ServiceInstanceId. **Tags:** atp.Status=draft

Table 3.41: TransportLayerIndependentIdCollectionSet

Class	**TransportLayerIndependentInstanceId**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransportLayerIndependentServiceInstanceId			
Note	This element defines a transport layer independent ServiceInstanceId that can be used in the ara::com proxy class resp. skeleton class generation to define the Instance Identifier. Please note that this ServiceInstanceId is an alternative to the fully qualified PortPrototype name. **Tags:** atp.Status=draft			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
instanceId	String	1	attr	This attribute defines a transport layer independent ServiceInstanceId.
instanceIdProps	PortPrototypeProps	0..1	aggr	TransportLayerIndependentInstanceId is an alternative to a PortPrototype in case that a component model is not used. It contains the same configuration properties as a PortPrototype and therefore aggregates the PortPrototypeProps element. **Tags:** atp.Status=draft
serviceInterface	ServiceInterface	1	tref	Reference to the ServiceInterface that is instantiated with the TransportLayerIndependentInstanceId. **Stereotypes:** isOfType **Tags:** atp.Status=draft

Table 3.42: TransportLayerIndependentInstanceId

Please note that examples in chapter A.4 are showing the usage and binding of `TransportLayerIndependentInstanceId` to a transport layer.

[constr_3360] `RPortPrototypeProps` are related only to `TransportLayerIndependentInstanceId`s representing a consumer Service Instance ⌈ The `RPortPrototypeProps` shall only be aggregated by a `TransportLayerIndependentInstanceId` in the role `instanceIdProps` that represents a consumer Service Instance. ⌋*()*

3.8 Adaptive AUTOSAR Application

This section contains the description of the formal modeling of the concept of an "application" itself. For this purpose, the meta-class `AdaptiveAutosarApplication` has been created.

[TPS_MANI_01008] Semantics of `AdaptiveAutosarApplication` ⌈ Meta-class `AdaptiveAutosarApplication` represents the unit of distribution of application software for the adaptive platform towards an integration step, i.e. application software developers shall pass the results of their work in the form of an `AdaptiveAutosarApplication` to the integration workflow. ⌋*(RS_MANI_00001)*

Class	**AdaptiveAutosarApplication**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign			
Note	This element describes a collection of executables that forms an Adaptive AUTOSAR Application. This corresponds to the definition of Application in SWS Execution Management. **Tags:** atp.Status=draft; atp.recommendedPackage=AdaptiveAutosarApplications			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
executable	Executable	1..*	ref	Reference to executables that are contained in the Adaptive Autosar Application. **Tags:** atp.Status=draft
version	String	0..1	attr	Version of the Adaptive Autosar Application

Table 3.43: AdaptiveAutosarApplication

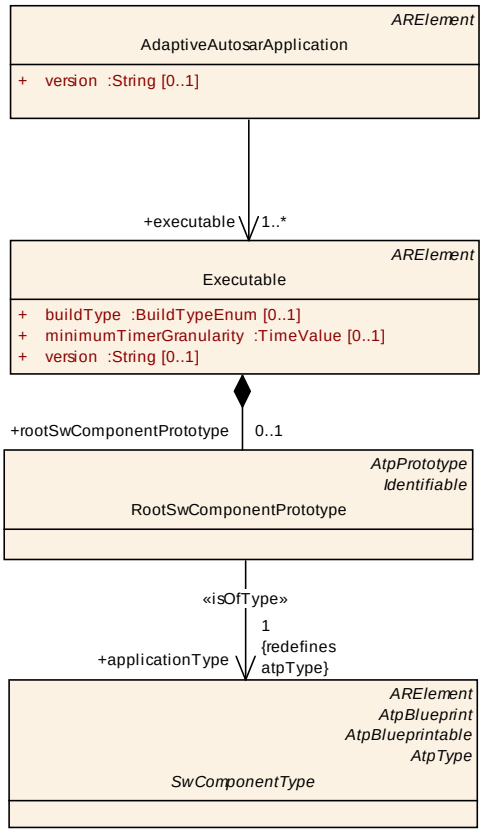

Figure 3.27: Modeling of the `AdaptiveAutosarApplication` and `Executable`

In general, an `AdaptiveAutosarApplication` may not be limited to the actual application level (i.e. conceptually located *above* the middleware), it is also supported to define an `AdaptiveAutosarApplication` that actually represents a part of the concrete implementation of an *AUTOSAR adaptive platform*.

A possible example for this kind of application could be a Diagnostic Manager (DM).

[TPS_MANI_01009] Standardized values of `AdaptiveAutosarApplication.category` ⌈ The following values of attribute `AdaptiveAutosarApplication.category` are standardized by AUTOSAR:

- `APPLICATION_LEVEL`: the `AdaptiveAutosarApplication` represents software on the application level (i.e. conceptually located *above* the middleware).
- `PLATFORM_LEVEL`: the `AdaptiveAutosarApplication` represents software on the platform level (i.e. conceptually located *on the level of* the middleware).

⌋*(RS_MANI_00001)*

Both the meta-class `AdaptiveAutosarApplication` and the meta-class `Executable` provide the ability to define a `version`.

The format and content of these version specifications is not constrained by the AUTOSAR standard, i.e the content of attribute `version` can be defined in custom

ways and the AUTOSAR standard does **not** make any assumptions on how different values of `version` are compared to each other.

Class	**Executable**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign			
Note	This meta-class represents an executable program. **Tags:** atp.Status=draft; atp.recommendedPackage=Executables			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
buildType	BuildTypeEnum	0..1	attr	This attribute describes the buildType of a module and/or platform implementation.
minimumTimerGranularity	TimeValue	0..1	attr	This attribute describes the minimum timer resolution (TimeValue of one tick) that is required by the Executable.
rootSwComponentPrototype	RootSwComponentPrototype	0..1	aggr	This represents the root SwCompositionPrototype of the Executable. This aggregation is required (in contrast to a direct reference of a SwComponentType) in order to support the definition of instanceRefs in Executable context. **Tags:** atp.Status=draft
transformationPropsToServiceInterfaceMappingSet	TransformationPropsToServiceInterfaceMappingSet	0..1	ref	Reference to a set of serialization properties that are defined for ServiceInterfaces of the Executable. **Tags:** atp.Status=draft
version	String	0..1	attr	Version of the executable.

Table 3.44: Executable

Each `AdaptiveAutosarApplication` can refer to 1..* `Executable`s. For practical purposes, this relation can be translated to "`AdaptiveAutosarApplication` consists of 1..* `Executable`s".

In contrast to a potential modeling of this relation as an aggregation, however, the reference-based approach supports the existence of the same `Executable` in the collection `AdaptiveAutosarApplication.executable` of several `AdaptiveAutosarApplication`s.

[TPS_MANI_01010] Root element for a hierarchical software-component ⌈ `Executable` aggregates meta-class `RootSwComponentPrototype` in the role `rootSwComponentPrototype` to provide a root element for an arbitrarily nested hierarchy of software-components represented by the reference `RootSwComponentPrototype.applicationType`. ⌋*(RS_MANI_00004)*

Please note that the aggregation of `RootSwComponentPrototype` by `Executable` is the basis for the applicability of an ≪`instanceRef`≫ reference into the hierarchy of software-components that represent the functionality of the `Executable`.

This modeling approach is similar to the modeling of a System on the *AUTOSAR classic platform*.

[TPS_MANI_03056] Optionality of Executable.rootSwComponentPrototype ⌈ The aggregation Executable.rootSwComponentPrototype has been made optional in order to support the implementation of *platform modules* that do not utilize any service oriented communication and don't require any further formalization. ⌋ *(RS_MANI_00023)*

[constr_1492] SwComponentType referenced as Executable.rootSwComponentPrototype.applicationType ⌈ Any SwComponentType referenced in the role Executable.rootSwComponentPrototype.applicationType, or used to type a SwComponentPrototype nested inside the SwComponentType referenced in the role Executable.rootSwComponentPrototype.applicationType shall **only** be either a CompositionSwComponentType or an AdaptiveApplicationSwComponentType. ⌋*()*

The example depicted in Figure 3.28 exemplifies the statement of [constr_1492]. The example shows a component hierarchy that consists of SwComponentPrototypes that are excursively typed by either a CompositionSwComponentType or an AdaptiveApplicationSwComponentType.

While the left part of Figure 3.28 resembles the modeling in the meta-model, the right part uses a simplified notation to give an idea how the nested definition of software-components could look like.

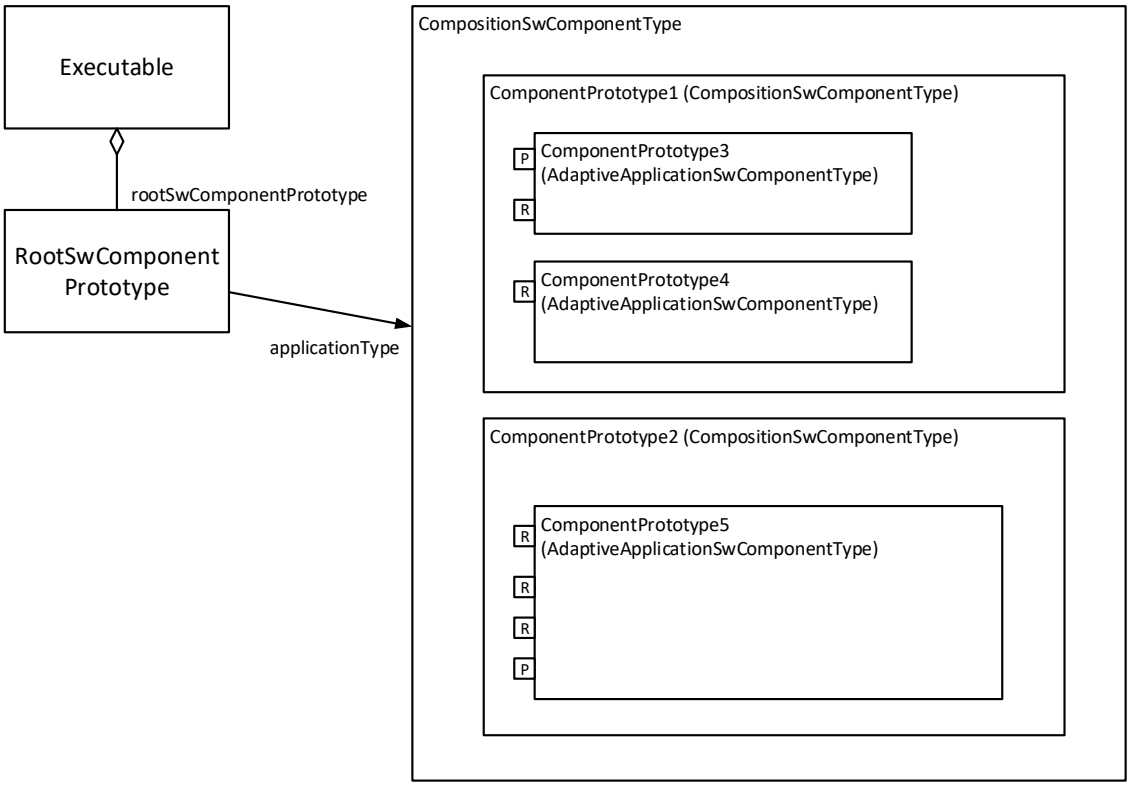

Figure 3.28: Example of the possible structure of an `Executable`

An obvious consequence of [constr_1492] is that no software-component that could be used on the *AUTOSAR classic platform* is allowed on the *AUTOSAR adaptive platform*, i.e. in the context of a `Executable.rootSwComponentPrototype.applicationType`.

Software-components on the *AUTOSAR adaptive platform* are mainly defined by their interaction with the outside world by means of `PortPrototype`s typed by `ServiceInterface`s. The definition of an internal behavior, with a minor exception, is not foreseen.

This lack of internal structure, in combination with decisions made regarding the scope of the generation of header files, leads to a situation where the implementation of a software component in source code is (in comparison to the situation on the *AUTOSAR classic platform*) way less subject to a strict separation.

In other words, there is no real motivation to implement software-components separately from each other. It would be possible, although not encouraged, to implement all software-components of a given executable program directly within the `Main()` function of the program.

Class	**RootSwComponentPrototype**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Process			
Note	The RootSwCompositionPrototype represents the top-level-composition of software components within an Executable. The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including PortPrototypes, PortInterfaces, VariableDataPrototypes, etc.). **Tags:** atp.Status=draft			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
applicationType	SwComponentType	1	tref	This SwComponnetType acts as the Type of the RootSwComponentPrototype. **Stereotypes:** isOfType **Tags:** atp.Status=draft

Table 3.45: RootSwComponentPrototype

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	***Type***	***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 PortPrototypes through which this SwComponentType 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 3.46: SwComponentType

Class	**CompositionSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	A CompositionSwComponentType aggregates SwComponentPrototypes (that in turn are typed by SwComponentTypes) as well as SwConnectors for primarily connecting SwComponentPrototypes among each others and towards the surface of the CompositionSwComponentType. By this means hierarchical structures of software-components can be created. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
component	SwComponentPrototype	*	aggr	The instantiated components that are part of this composition. The aggregation of SwComponentPrototype is subject to variability with the purpose to support the conditional existence of a SwComponentPrototype. Please be aware: if the conditional existence of SwComponentPrototypes is resolved post-build the deselected SwComponentPrototypes are still contained in the ECUs build but the instances are inactive in in that they are not scheduled by the RTE. The aggregation is marked as atpSplitable in order to allow the addition of service components to the ECU extract during the ECU integration. The use case for having 0 components owned by the CompositionSwComponentType could be to deliver an empty CompositionSwComponentType to e.g. a supplier for filling the internal structure. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild

connector	SwConnector	*	aggr	SwConnectors have the principal ability to establish a connection among PortPrototypes. They can have many roles in the context of a CompositionSwComponentType. Details are refined by subclasses. The aggregation of SwConnectors is subject to variability with the purpose to support variant data flow. The aggregation is marked as atpSplitable in order to allow the extension of the ECU extract with AssemblySwConnectors between ApplicationSwComponentTypes and ServiceSwComponentTypes during the ECU integration. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=postBuild
constantValueMapping	ConstantSpecificationMappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortComSpec. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=constantValueMapping
dataTypeMapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the used ApplicationDataTypes in PortInterfaces. Background: when developing subsystems it may happen that ApplicationDataTypes are used on the surface of CompositionSwComponentTypes. In this case it would be reasonable to be able to also provide the intended mapping to the ImplementationDataTypes. However, this mapping shall be informal and not technically binding for the implementers mainly because the RTE generator is not concerned about the CompositionSwComponentTypes. Rationale: if the mapping of ApplicationDataTypes on the delegated and inner PortPrototype matches then the mapping to ImplementationDataTypes is not impacting compatibility. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dataTypeMapping

instantiationRTEEventProps	InstantiationRTEEventProps	*	aggr	This allows to define instantiation specific properties for RTE Events, in particular for instance specific scheduling. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortLabel, variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime

Table 3.47: CompositionSwComponentType

3.9 Serialization Properties

In Adaptive Autosar the serialization code is generated out of the service description and is compiled and executed in the application context.

The meta-class `TransformationPropsToServiceInterfaceMapping` defines the serialization for a `ServiceInterface` and provides the necessary serialization settings with the `TransformationProps` element.

The existence of a `TransformationPropsToServiceInterfaceMapping` demands the existence of serialization code that is linked with the application component object file to an application binary.

Class	**TransformationPropsToServiceInterfaceMappingSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransformationConfiguration			
Note	Collection of TransformationPropsToServiceInterfaceMappings. **Tags:** atp.Status=draft; atp.recommendedPackage=TransformationPropsToServiceInterfaceMappingSets			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
transformationPropsToServiceInterfaceMapping	TransformationPropsToServiceInterfaceMapping	*	aggr	Mapping that assigns serialization properties to a ServiceInterface. **Tags:** atp.Status=draft

Table 3.48: TransformationPropsToServiceInterfaceMappingSet

Class	**TransformationPropsToServiceInterfaceMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign			
Note	This meta-class represents the ability to associate a ServiceInterface with TransformationProps. The elements of the referenced Service Interface will be serialized according to the settings defined in the TransformationProps. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
serviceInterface	ServiceInterface	0..1	ref	This represents the reference to the applicable ServiceInterface. **Tags:** atp.Status=draft
transformationProps	TransformationProps	0..1	ref	This represents the reference to the applicable Serialization properties. **Tags:** atp.Status=draft

Table 3.49: TransformationPropsToServiceInterfaceMapping

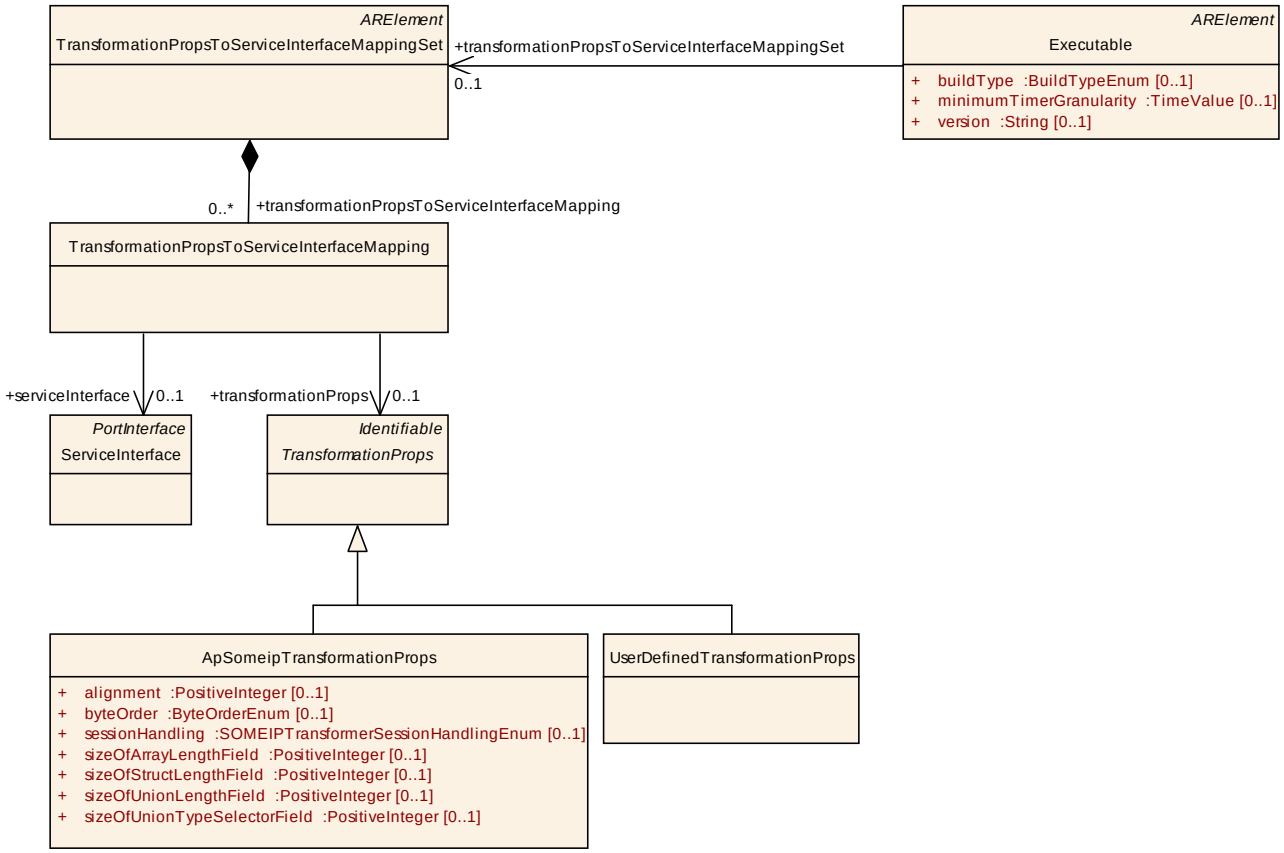

Figure 3.29: Association of serialization properties with a ServiceInterface in the context of an Executable

[TPS_MANI_03101] SOME/IP serialization ⌈ The `ApSomeipTransformationProps` meta-class that is referenced by the `TransformationPropsToServiceInterfaceMapping` in the role `transformationProps` provides the ability to define a

SOME/IP serialization for a ServiceInterface that is referenced by the TransformationPropsToServiceInterfaceMapping in the role serviceInterface. ⌋ *(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03103] Default size for all array length fields ⌈ The attribute sizeOfArrayLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available arrays defined in the ServiceInterface that is referenced by the TransformationPropsToServiceInterfaceMapping in the role serviceInterface. ⌋ *(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03104] Default size for all structure length fields ⌈ The attribute sizeOfStructLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available structures defined in the ServiceInterface that is referenced by the TransformationPropsToServiceInterfaceMapping in the role serviceInterface. ⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03105] Default size for all union length fields ⌈ The attribute sizeOfUnionLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available unions defined in the ServiceInterface that is referenced by the TransformationPropsToServiceInterfaceMapping in the role serviceInterface. ⌋ *(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03106] Default size for all union type selector fields ⌈ The attribute sizeOfUnionTypeSelectorField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceMapping in the role transformationProps defines the size of a type field generated by SOME/IP in front of all available unions defined in the ServiceInterface that is referenced by the TransformationPropsToServiceInterfaceMapping in the role serviceInterface. ⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03107] Default alignment for all dynamic DataPrototypes ⌈ The attribute alignment of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceMapping in the role transformationProps defines the padding for alignment purposes that will be added by SOME/IP after the serialized data of all variable data length data elements defined in the ServiceInterface that is referenced by the TransformationPropsToServiceInterfaceMapping in the role serviceInterface. ⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03108] Default Byte Order for all DataPrototypes ⌈ The attribute byteOrder of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceMapping in the role transformationProps defines the Byte Order in the serialized data stream resulting from the ServiceInterface

that is referenced by the TransformationPropsToServiceInterfaceMapping in the role serviceInterface. ⌋(*RS_MANI_00008*, *RS_MANI_00025*)

Please note that more details about ApSomeipTransformationProps can be found in chapter 5.3.

Class	**ApSomeipTransformationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransformationConfiguration			
Note	SOME/IP serialization properties. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, TransformationProps			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
alignment	PositiveInteger	0..1	attr	Specifies the alignment of dynamic data in the serialized data stream. The alignment is specified in Bits.
byteOrder	ByteOrderEnum	0..1	attr	Specifies the byte order of data in the serialized data stream.
sessionHandling	SOMEIPTransformerSessionHandlingEnum	0..1	attr	Defines whether the SOME/IP transformer shall use session handling for Sender/Receiver communication.
sizeOfArrayLengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of an Array. It describes the size of the length field (in Bytes) that will be put in front of the Array in the SOME/IP message. In contrast to Classic AUTOSAR this attirbute defines the value for both, fixed-size and dynamic-size arrays.
sizeOfStructLengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of an Struct. It describes the size of the length field (in Bytes) that will be put in front of the Struct in the SOME/IP message.
sizeOfUnionLengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a Union. It describes the size of the length field (in Bytes) that will be put in front of the Union in the SOME/IP message.
sizeOfUnionTypeSelectorField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a Union. It describes the size of the type selector field (in Bytes) that will be put in front of the Union in the SOME/IP message.

Table 3.50: ApSomeipTransformationProps

[TPS_MANI_03102] UserDefined serialization ⌈ The UserDefinedTransformationProps meta-class that is referenced by the TransformationPropsToServiceInterfaceMapping in the role transformationProps provides the ability to define a User defined serialization for a ServiceInterface that is referenced by the

TransformationPropsToServiceInterfaceMapping in the role serviceInterface. ⌋(*RS_MANI_00014*, *RS_MANI_00025*)

Please note that UserDefinedTransformationProps is derived from meta-class Identifiable and therefore has the ability to describe special data (sdg) by which it is possible to define custom structural extensions of an AUTOSAR model in a generic way. For more information about special data please refer to [5].

Class	**UserDefinedTransformationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransformationConfiguration			
Note	UserDefined serialization properties. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, TransformationProps			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 3.51: UserDefinedTransformationProps

4 Diagnostic Mapping

4.1 Overview

The configuration of diagnostics on the *AUTOSAR adaptive platform* will typically be done by creating a Diagnostic Extract by means of the Diagnostic Extract Template [8] that is also used on the *AUTOSAR classic platform*.

Therefore, concepts within the Diagnostic Extract should be similarly applicable to models on both platforms in a uniform fashion.

It can even be safely expected that a given Diagnostic Extract can be divided into parts that apply for ECUs build on top of the *AUTOSAR classic platform* and parts that apply to ECUs built on top of the *AUTOSAR adaptive platform* that all belong to the same vehicle.

In terms of applicability to this document, the part of the Diagnostic Extract that is relevant in this context is the mapping between the definition of information related to diagnostic protocol content and the application software.

Following the pattern of communication on the *AUTOSAR adaptive platform*, interaction between the application software and the platform module for diagnostics (the so-called `Diagnostic Manager`) is also using service-oriented communication.

This raises the question how the communication ends on both application and platform software get together in the course of a service discovery. This issue can be addressed by utilizing modeling concepts existing in a Diagnostic Extract on the *AUTOSAR adaptive platform*.

Specifically, by formally modeling the relation between the `Diagnostic Manager` and specific endpoints in the application software it is possible to configure the service-oriented communication in a way that ends that are supposed to be connected are connected as the service discovery unfolds.

The meta-classes that need to be considered for this purpose are in the following list:

- `DiagnosticServiceDataMapping`
- `DiagnosticServiceSwMapping`
- `DiagnosticEventPortMapping`
- `DiagnosticOperationCyclePortMapping`
- `DiagnosticEnableConditionPortMapping`
- `DiagnosticStorageConditionPortMapping`

In order to exemplify the approach, The diagram depicted in Figure 4.1 describes a very simplistic situation where **one** `event` contained **in one of two** different `PPortPrototypes` exposed by an `AdaptiveApplicationSwComponentType` is accessed by

the Diagnostic Manager on the *AUTOSAR adaptive platform* with the purpose of adding the value to e.g. a DID response telegram.

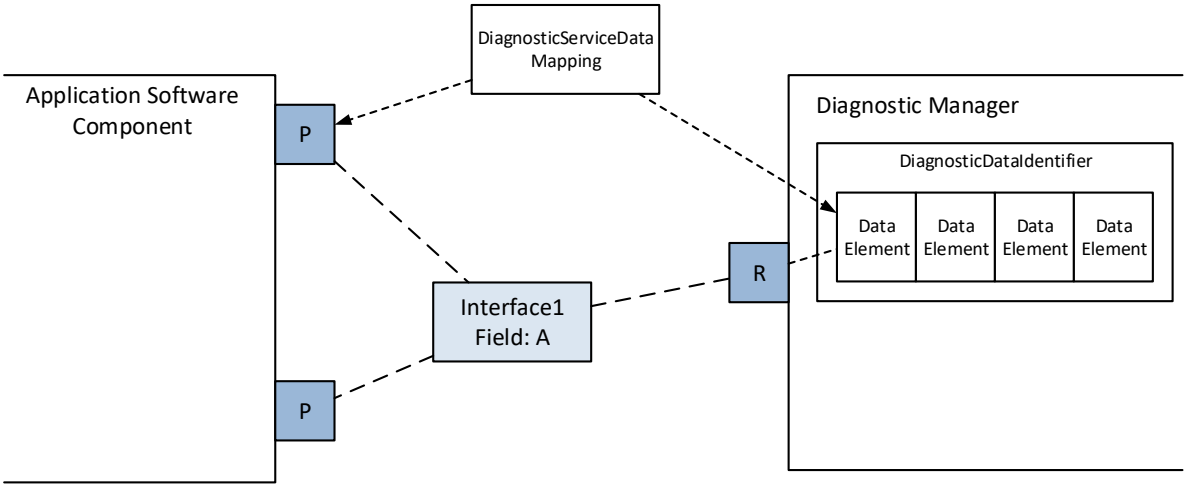

Figure 4.1: Example data exchange for diagnostic purpose

In this situation, the Diagnostic Manager obviously needs to be aware which of the two available events has to be accessed in particular. In other words, the service discovery settings of the Diagnostic Manager need to be clear about which of the two available PortPrototypes to connect to.

The process of configuring the Diagnostic Manager's service discovery settings accordingly can be assisted by the existence of (in this case) a DiagnosticServiceDataMapping that formally identifies the applicable event in the context of the enclosing PortPrototype.

Of course, the specifics of the PortPrototype on the side of the Diagnostic Manager need to be derived from the configuration (in this case, the definition of a DiagnosticDataElement owned by a DiagnosticDataIdentifier) of the external behavior of the diagnostic stack on the *AUTOSAR adaptive platform*, as described by a corresponding Diagnostic Extract [8].

A further kind of mapping that is necessary to enable diagnostics on the *AUTOSAR adaptive platform* comes with slightly more complexity. In this case use-cases are implemented that may or may not involve several communication ends (in the form of PortPrototypes).

Class	**DiagnosticDataIdentifier**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to model a diagnostic data identifier (DID) that is fully specified regarding the payload at configuration-time. **Tags:** atp.recommendedPackage=DiagnosticDataIdentifiers			
Base	ARElement, ARObject, CollectableElement, DiagnosticAbstractDataIdentifier, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataElement	DiagnosticParameter	1..*	aggr	This is the dataElement associated with the DiagnosticDataIdentifier. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataElement, variationPoint.shortLabel vh.latestBindingTime=postBuild
didSize	PositiveInteger	0..1	attr	This attribute indicates the size of the DiagnosticDataIdentifier.
representsVin	Boolean	0..1	attr	This attributes indicates whether the specific DiagnosticDataIdentifier represents the vehicle identification.
supportInfoByte	DiagnosticSupportInfoByte	0..1	aggr	This attribute represents the supported information associated with the DiagnosticDataIdentifier.

Table 4.1: DiagnosticDataIdentifier

The response to this situation on the *AUTOSAR classic platform* has been the definition of the `SwcServiceDependency` that allows for associating several `PortPrototype`s in specific roles to a given use-case.

Although (thanks to the existence of the `ServiceInterface`) the need for involving different `PortPrototype`s in the implementation of a given use case has slightly gone down, there is still enough motivation to keep using this pattern on the *AUTOSAR adaptive platform* as well.

For example, one benefit of this approach over a seemingly more straightforward implementation to refer to a `PortPrototype` directly is the ability to let several `PortPrototype`s (where e.g. some may represent server functionality, and the rest could represent client functionality) in concert in order to implement a given use case.

Figure 4.2 provides a visual explanation of how this kind of diagnostic mapping to model elements on the *AUTOSAR adaptive platform* works.

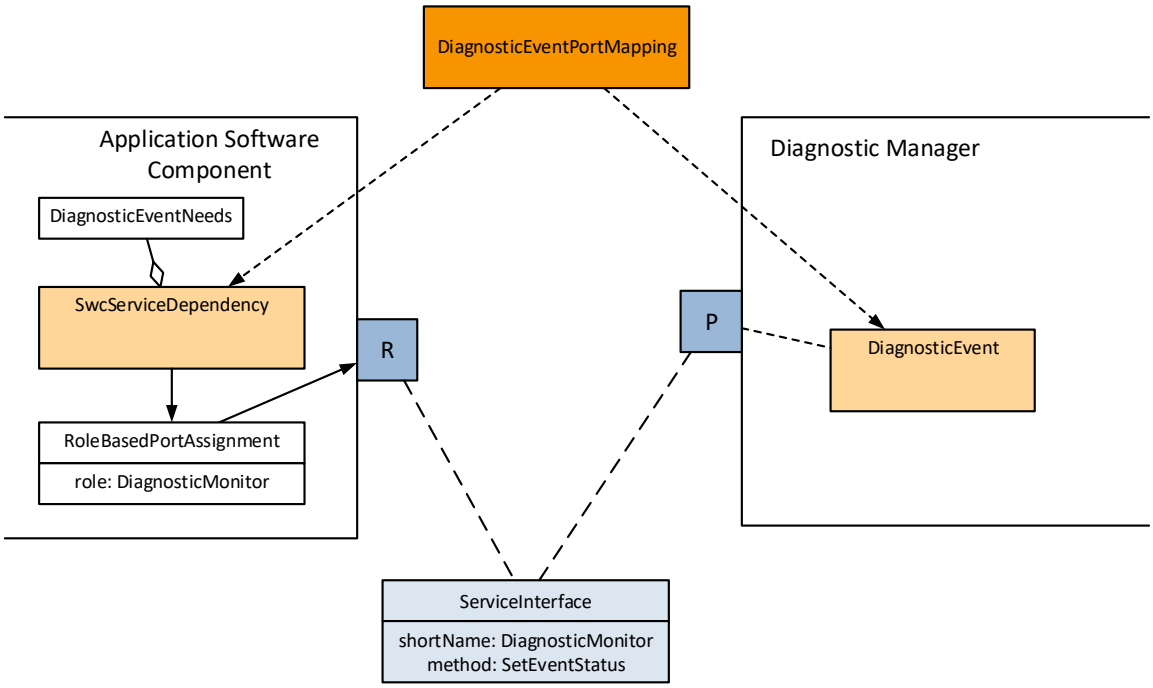

Figure 4.2: Example mapping to associate a `PortPrototype` with a `DiagnosticEvent`

4.2 Diagnostic Data Mapping

[TPS_MANI_01037] Diagnostic data mapping on the *AUTOSAR adaptive platform* ⌈ The diagnostic data mapping on the *AUTOSAR adaptive platform* is created by means of meta-class `DiagnosticServiceDataMapping` that maps a `DiagnosticDataElement` to a `DataPrototype` referenced in the role `mappedApDataElement`. ⌋*(RS_MANI_00005)*

[TPS_MANI_01060] Use cases for the application of `DiagnosticServiceDataMapping` ⌈ `DiagnosticServiceDataMapping` shall only be used where access to data is free of side-effects. This is the case for `field`s and, at least with respect to the value, `event`s. ⌋*(RS_MANI_00005)*

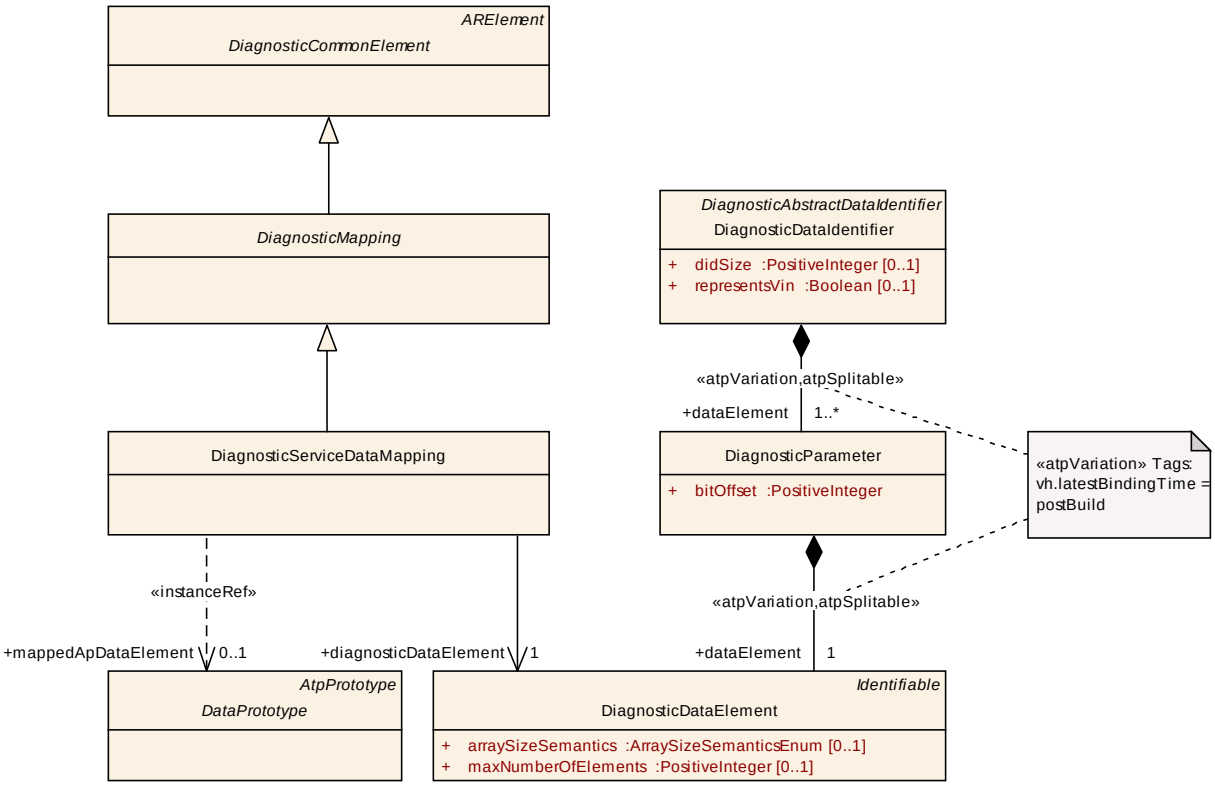

Figure 4.3: Modeling of the diagnostic data mapping

Please note that the `DiagnosticServiceDataMapping` can be applied on models on the *AUTOSAR adaptive platform* because the mapping target is a `DataPrototype` that is aggregated by a `ServiceInterface` in the context of a `PortPrototype`.

In other words, the `DiagnosticServiceDataMapping` applies for the mapping to an `event` or `field`, **or even to an element of** an `event` or `field`.

[constr_1496] `DiagnosticServiceDataMapping.mappedApDataElement` shall only refer to specific sub-classes of `DataPrototype` ⌈ A `DiagnosticServiceDataMapping.mappedApDataElement` shall only refer to an `event` or a `field` or a `DataPrototype` owned by an `event` or a `field`. ⌋*()*

Please not that the existence of [constr_1496] is a direct consequence of the existence of [TPS_MANI_01060].

In particular, [constr_1496] prevents the creation of a `DiagnosticServiceDataMapping` to a `ArgumentDataPrototype`. In the diagnostic context, `ArgumentDataPrototype` are mainly used in the argument list of the sub-functions of diagnostic routines which are rarely free of side-effects.

Class	**DiagnosticServiceDataMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::ServiceMapping			
Note	This represents the ability to define a mapping of a diagnostic service to a software-component. This kind of service mapping is applicable for the usage of SenderReceiverInterfaces. **Tags:** atp.recommendedPackage=DiagnosticServiceMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
diagnosticDataElement	DiagnosticDataElement	1	ref	This represents the applicable payload that corresponds to the referenced DataPrototype in the role mappedDataElement.
mappedApDataElement	DataPrototype	0..1	iref	This represents the dataElement in the application software of an adaptive AUTOSAR application that is accessed for diagnostic purpose. **Tags:** atp.Status=draft
mappedDataElement	DataPrototype	0..1	iref	This represents the dataElement in the application software that is accessed for diagnostic purpose.

Table 4.2: DiagnosticServiceDataMapping

Class	**DiagnosticDataElement**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to describe a concrete piece of data to be taken into account for diagnostic purposes.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
arraySizeSemantics	ArraySizeSemanticsEnum	0..1	attr	This attribute controls the meaning of the value of the array size.
maxNumberOfElements	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of how many elements the array can take.
swDataDefProps	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties in order to support the definition of e.g. computation formulae and data constraints.

Table 4.3: DiagnosticDataElement

4.3 Diagnostic Software Mapping

[TPS_MANI_01038] Diagnostic software mapping on the *AUTOSAR adaptive platform* ⌈ The diagnostic software mapping on the *AUTOSAR adaptive platform* is created by means of meta-class `DiagnosticServiceSwMapping` that maps a `DiagnosticServiceInstance` to a `SwcServiceDependency` referenced in the

role `mappedSwcServiceDependencyInExecutable` respectively a `DiagnosticDataElement` in the role `diagnosticDataElement`. ⌋(*RS_MANI_00005*)

As depicted by Figure 4.4, the application of a `DiagnosticServiceSwMapping` on the *AUTOSAR adaptive platform* requires the existence of a `SwcServiceDependency`, defined in the context of an `AdaptiveApplicationSwComponentType` (see section 3.2).

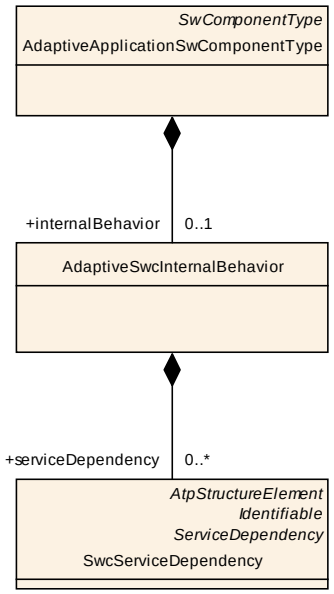

Figure 4.4: Modeling of internal behavior for the modeling of `DiagnosticServiceSwMapping`

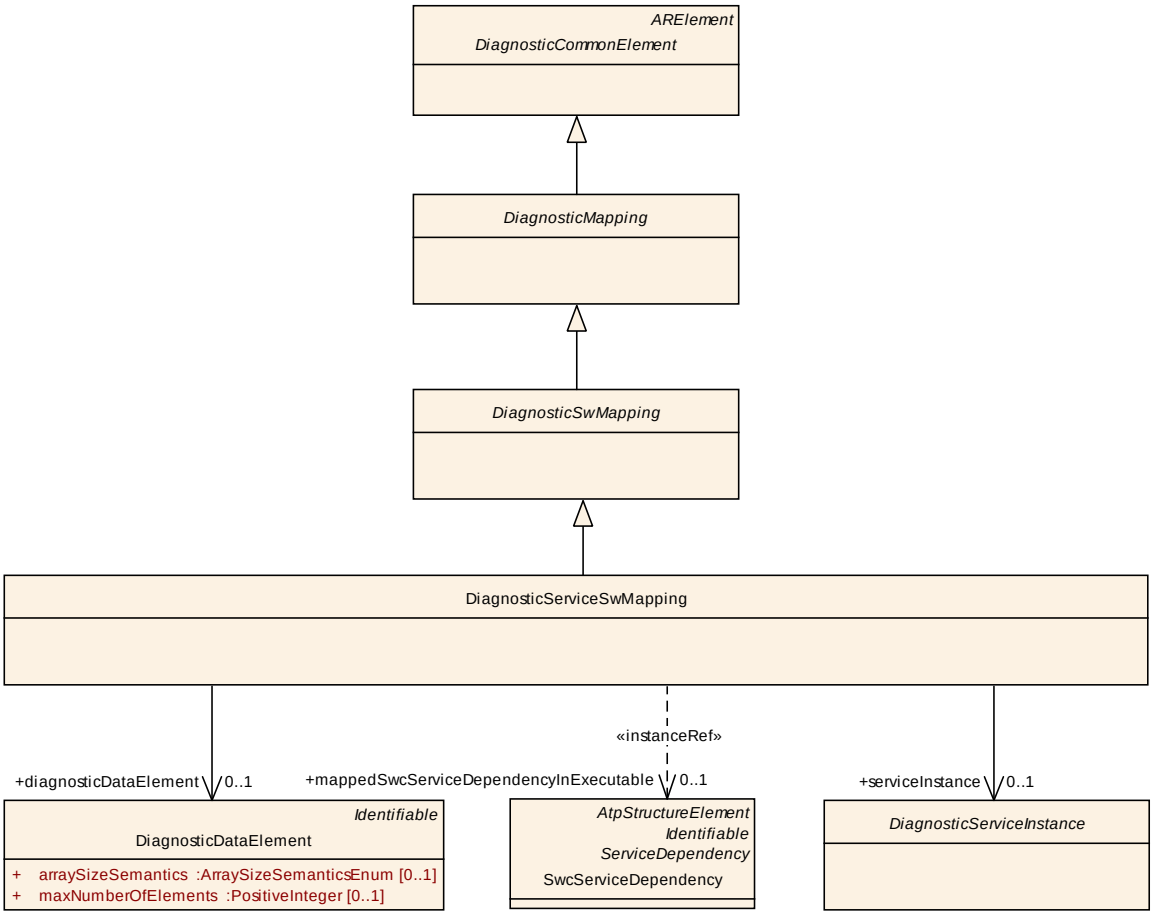

Figure 4.5: Modeling of the diagnostic software mapping

[constr_1499] Target `SwcServiceDependency` of `DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable` ⌈ Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable` shall **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior`. ⌋()

Class	**DiagnosticServiceSwMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::ServiceMapping			
Note	This represents the ability to define a mapping of a diagnostic service to a software-component or a basic-software module. If the former is used then this kind of service mapping is applicable for the usage of ClientServerInterfaces. **Tags:** atp.recommendedPackage=DiagnosticServiceMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
diagnosticDataElement	DiagnosticDataElement	0..1	ref	This represents a DiagnosticDataElement required to execute the respective diagnostic service in the context of the diagnostic service mapping,

mappedBswServiceDependency	BswServiceDependencyIdent	0..1	ref	This is supposed to represent a reference to a BswServiceDependency. the latter is not derived from Referrable and therefore this detour needs to be implemented to still let BswServiceDependency become the target of a reference.
mappedFlatSwcServiceDependency	SwcServiceDependency	0..1	ref	This represents the ability to refer to an AtomicSwComponentType that is available without the definition of how it will be emebdded into the component hiearchy.
mappedSwcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	This represents the ability to point into the component hiearchy of an adaptive AUTOSAR model (under possible consideration of the rootSoftwareComposition) **Tags:** atp.Status=draft
mappedSwcServiceDependencyInSystem	SwcServiceDependency	0..1	iref	This represents the ability to point into the component hiearchy (under possible consideration of the rootSoftwareComposition)
serviceInstance	DiagnosticServiceInstance	0..1	ref	This represents the service instance that needs to be considered in this diagnostics service mapping,

Table 4.4: DiagnosticServiceSwMapping

Class	**SwcServiceDependency**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServiceMapping			
Note	Specialization of ServiceDependency in the context of an SwcInternalBehavior. It allows to associate ports, port groups and (in special cases) data defined for an atomic software component to a given ServiceNeeds element.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable, ServiceDependency			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
assignedData	RoleBasedDataAssignment	*	aggr	Defines the role of an associated data object of the same component. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
assignedPort	RoleBasedPortAssignment	*	aggr	Defines the role of an associated port of the same component. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=assignedPort, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

representedPortGroup	PortGroup	0..1	ref	This reference specifies an association between the ServiceNeeeds and a PortGroup, for example to request a communication mode which applies for communication via these ports. The referred PortGroup shall be local to this atomic SWC, but via the links between the PortGroups, a tool can evaluate this information such that all the ports linked via this port group on the same ECU can be found.
serviceNeeds	ServiceNeeds	1	aggr	The associated ServiceNeeds.

Table 4.5: SwcServiceDependency

4.4 Diagnostic Event to Port Mapping

[TPS_MANI_01048] Mapping of `DiagnosticEvent` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈ On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticEvent` and one or many `PortPrototype`s is created by using the `DiagnosticEventPortMapping` that refers to a `DiagnosticEvent` in the role `diagnosticEvent` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋*(RS_MANI_00005)*

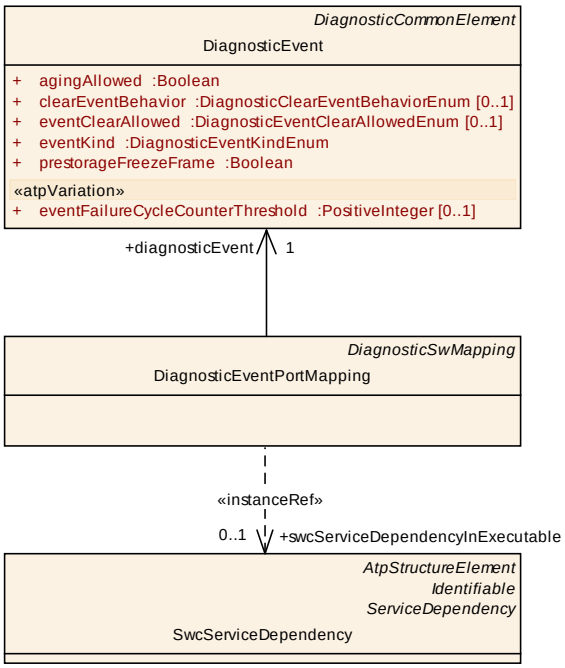

Figure 4.6: Modeling of `DiagnosticEventPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1500] Target `SwcServiceDependency` of `DiagnosticEventPortMapping.swcServiceDependencyInExecutable` ⌈ Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticEventPortMapping.swcSer-`

viceDependencyInExecutable shall **only** be aggregated in the role serviceDependency by an AdaptiveSwcInternalBehavior. ⌋()

Class	**DiagnosticEvent**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent			
Note	This element is used to configure DiagnosticEvents. **Tags:** atp.recommendedPackage=DiagnosticEvents			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
agingAllowed	Boolean	1	attr	This represents the decision whether aging is allowed for this DiagnosticEvent.
clearEventBehavior	DiagnosticClearEventBehaviorEnum	0..1	attr	This attribute defines the resulting UDS status byte for the related event, which shall not be cleared according to the ClearEventAllowed callback.
connectedIndicator	DiagnosticConnectedIndicator	*	aggr	Event specific description of Indicators. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
eventClearAllowed	DiagnosticEventClearAllowedEnum	0..1	attr	This attribute defines whether the Dem has access to a "ClearEventAllowed" callback.
eventFailureCycleCounterThreshold	PositiveInteger	0..1	attr	This attribute defines the number of failure cycles for the event based fault confirmation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
eventKind	DiagnosticEventKindEnum	1	attr	This attribute is used to distinguish between SWC and BSW events.
prestorageFreezeFrame	Boolean	1	attr	This attribute describes whether the Prestorage of FreezeFrames is supported by the assigned event or not. True: Prestorage of FreezeFrames is supported False: Prestorage of FreezeFrames is not supported

Table 4.6: DiagnosticEvent

Class	**DiagnosticEventPortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticEventNeeds the DiagnosticEvent is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
bswServiceDependency	BswServiceDependencyIdent	0..1	ref	Reference to a BswServiceDependency that links ServiceNeeds to BswModuleEntries.
diagnosticEvent	DiagnosticEvent	1	ref	Reference to the DiagnosticEvent that is assigned to SWC service ports with DiagnosticEventNeeds.
swcFlatServiceDependency	SwcServiceDependency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports.
swcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	Thia sggregation allows for the usage of the DiagnosticEventPortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft
swcServiceDependencyInSystem	SwcServiceDependency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports.

Table 4.7: DiagnosticEventPortMapping

4.5 Diagnostic Operation Cycle to Port Mapping

[TPS_MANI_01049] Mapping of `DiagnosticOperationCycle` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈ On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticOperationCycle` and one or many `PortPrototype`s is created by using the `DiagnosticEventPortMapping` that refers to a `DiagnosticOperationCycle` in the role `operationCycle` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋ *(RS_MANI_00005)*

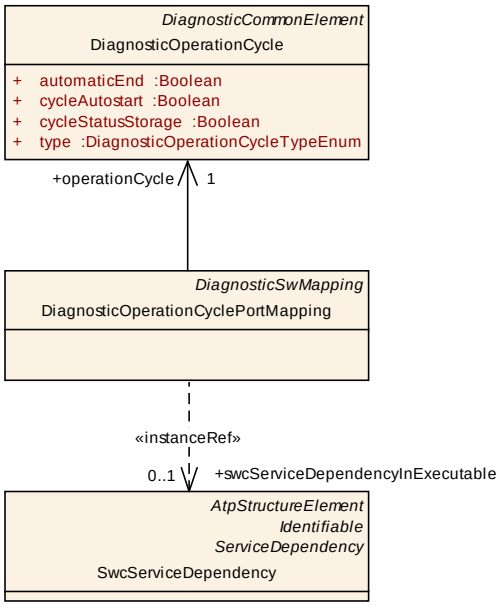

Figure 4.7: Modeling of `DiagnosticOperationCyclePortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1501] Target `SwcServiceDependency` of `DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable` ⌈ Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable` shall **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior`. ⌋*()*

Class	**DiagnosticOperationCycle**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticOperationCycle			
Note	Definition of an operation cycle that is the base of the event qualifying and for Dem scheduling. **Tags:** atp.recommendedPackage=DiagnosticOperationCycles			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
automaticEnd	Boolean	1	attr	If set to true the driving cycle shall automatically end at either Dem_Shutdown() or Dem_Init().
cycleAutostart	Boolean	1	attr	This attribute defines if the operation cycles is automatically re-started during Dem_PreInit.
cycleStatusStorage	Boolean	1	attr	Defines if the operation cycle state is available over the power cycle (stored non-volatile) or not. true: the operation cycle state is stored non-volatile false: the operation cycle state is only stored volatile
type	DiagnosticOperationCycleTypeEnum	1	attr	Operation cycles types for the Dem to be supported by cycle-state APIs.

Table 4.8: DiagnosticOperationCycle

Class	**DiagnosticOperationCyclePortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticOperationCycleNeeds the DiagnosticOperationCycle is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
operationCycle	DiagnosticOperationCycle	1	ref	Reference to the DiagnosticOperationCycle that is assigned to SWC service ports with DiagnosticOperationCycleNeeds.
swcFlatServiceDependency	SwcServiceDependency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports.
swcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	This aggrataion allows for the usage of the DiagnosticOperationCyclePortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft
swcServiceDependencyInSystem	SwcServiceDependency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports.

Table 4.9: DiagnosticOperationCyclePortMapping

4.6 Diagnostic Enable Condition to Port Mapping

[TPS_MANI_01050] Mapping of `DiagnosticEnableCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈ On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticEnableCondition` and one or many `PortPrototype`s is created by using the `DiagnosticEventPortMapping` that refers to a `DiagnosticEnableCondition` in the role `enableCondition` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋ *(RS_MANI_00005)*

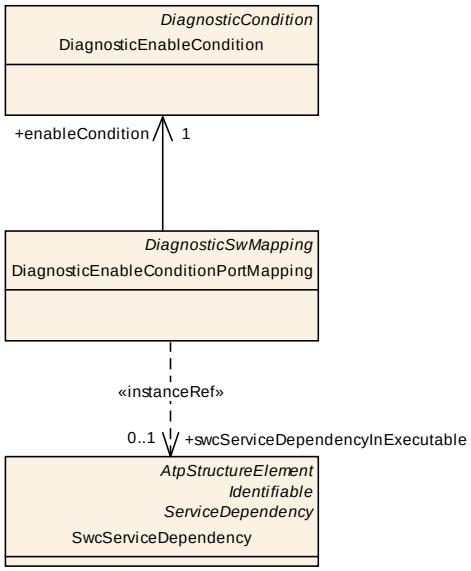

Figure 4.8: Modeling of `DiagnosticEnableConditionPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1502] Target `SwcServiceDependency` of `DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable` ⌈ Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable` shall **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior`. ⌋ *()*

Class	**DiagnosticEnableCondition**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
Note	Specification of an enable condition. **Tags:** atp.recommendedPackage=DiagnosticConditions			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticCondition, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 4.10: DiagnosticEnableCondition

Class	**DiagnosticEnableConditionPortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticEnableConditionNeeds the DiagnosticEnableCondition is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

enableCondition	DiagnosticEnableCondition	1	ref	Reference to the EnableCondition which is mapped to a SWC service port with DiagnosticEnableConditionNeeds.
swcFlatServiceDependency	SwcServiceDependency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports. This reference can be used in early stages of the development in order to identify the SwcServiceDependency without a full System Context.
swcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	This aggregation allows for the usage of the DiagnosticEnableConditionPortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft
swcServiceDependencyInSystem	SwcServiceDependency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports.

Table 4.11: DiagnosticEnableConditionPortMapping

4.7 Diagnostic Storage Condition to Port Mapping

[TPS_MANI_01051] Mapping of `DiagnosticStorageCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform* ⌈ On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticStorageCondition` and one or many `PortPrototype`s is created by using the `DiagnosticEventPortMapping` that refers to a `DiagnosticStorageCondition` in the role `diagnosticStorageCondition` as well as to a `SwcServiceDependency` in the role `swcServiceDependencyInExecutable`. ⌋*(RS_MANI_00005)*

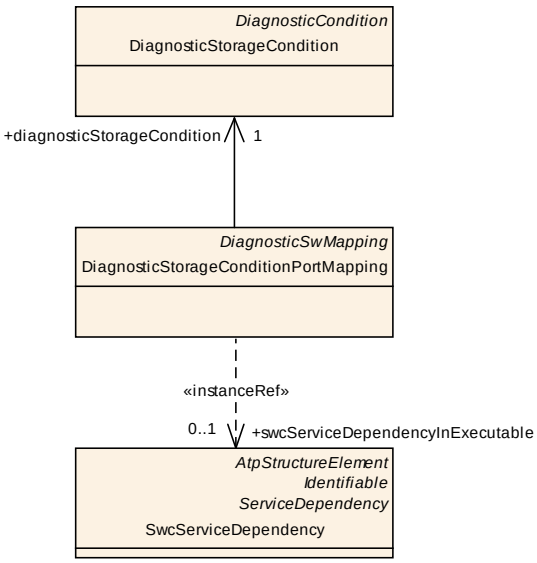

Figure 4.9: Modeling of `DiagnosticStorageConditionPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_1503] Target `SwcServiceDependency` of `DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable` ⌈ Any particular `SwcServiceDependency` that is referenced in the role `DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable` shall **only** be aggregated in the role `serviceDependency` by an `AdaptiveSwcInternalBehavior`. ⌋()

Class	**DiagnosticStorageCondition**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
Note	Specification of a storage condition. **Tags:** atp.recommendedPackage=DiagnosticConditions			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticCondition, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 4.12: DiagnosticStorageCondition

Class	**DiagnosticStorageConditionPortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMapping			
Note	Defines to which SWC service ports with DiagnosticStorageConditionNeeds the DiagnosticStorageCondition is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
diagnosticStorageCondition	DiagnosticStorageCondition	1	ref	Reference to the StorageCondition which is mapped to a SWC service port with DiagnosticStorageConditionNeeds.
swcFlatServiceDependency	SwcServiceDependency	0..1	ref	Reference to a SwcServiceDependencyType that links ServiceNeeds to SWC service ports.
swcServiceDependencyInExecutable	SwcServiceDependency	0..1	iref	This aggregation allows for the usage of the DiagnosticStorageConditionPortMapping on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft
swcServiceDependencyInSystem	SwcServiceDependency	0..1	iref	Instance reference to a SwcServiceDependency that links ServiceNeeds to SWC service ports.

Table 4.13: DiagnosticStorageConditionPortMapping

5 Application Manifest

5.1 Overview

The purpose of the application manifest is to provide information that is needed for the actual deployment of an application (formally modeled as an `SwComponentType`) onto the AUTOSAR adaptive platform.

One aspect of the deployment information is the provision of information that could in principle be provided as part of the application software code but which would make the application software code become very much bound to specific usage scenarios.

The general idea is to keep the application software code as independent as possible from the deployment scenario in order to increase the odds that the application software can be reused in different deployment scenarios.

In particular, the usage of `PortPrototype`s as a means to express communication with the "outside" of the application software allows for abstracting away the details (the concrete service instance identification) of the service configuration. As far as the model is concerned, the API between the application and the middleware is represented by the `PortPrototype`.

The application code does not use specific service instances but takes the `PortPrototype` as a symbolic replacement for this information. The specifics of this modeling aspect are described in section 6.

The top-level element of the `Application Manifest` definition is the `Process`, in reference to the fact that the unit of deployment on the *AUTOSAR adaptive platform* is a binary that, at runtime, makes a POSIX process.

[TPS_MANI_01011] Connection between application design and application deployment ⌈ The connection between the *application design* and the *application deployment* is implemented by means of a reference from meta-class `Process` to meta-class `Executable` in the role `executable`.

By modeling the reference in this direction it is possible to keep the design level independent of the deployment level and, at the same time, bind the deployment to a specific design. ⌋*(RS_MANI_00006)*

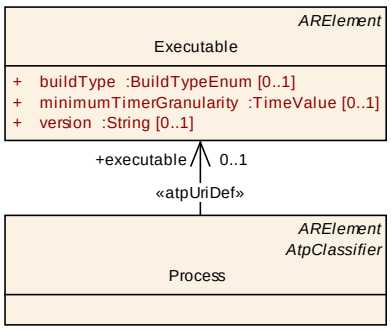

Figure 5.1: Relation of meta-classes `Executable` and `Process`

Class	**Process**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Process			
Note	This meta-class provides information required to execute the referenced executable. **Tags:** atp.Status=draft; atp.recommendedPackage=Processes			
Base	ARElement, ARObject, AtpClassifier, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
applicationModeMachine	ModeDeclarationGroupPrototype	0..1	aggr	Set of ApplicationStates (Modes) that are defined for the process. **Tags:** atp.Status=draft
executable	Executable	0..1	ref	Reference to executable that is executed in the process. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
modeDependentStartupConfig	ModeDependentStartupConfig	*	aggr	Applicable startup configurations. **Tags:** atp.Status=draft

Table 5.1: Process

Please note that the meta-model, as depicted in Figure 5.1 supports the existence of two or more `Process`es that reference the same `Executable`.

This is an indication that the specific `Executable` is supposed to be executed in several instances (i.e. in the form of POSIX processes) on the same platform. Such a situation is sketched in Figure 5.2

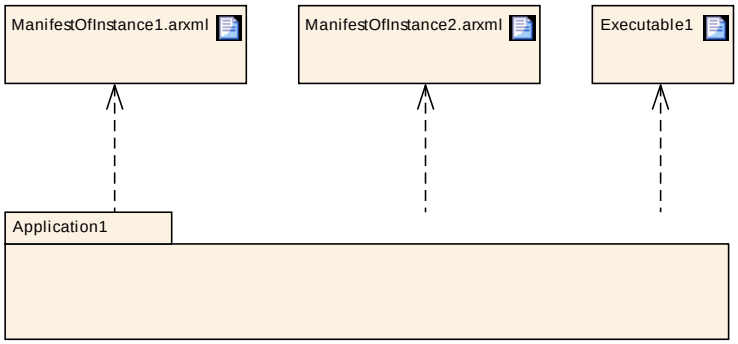

Figure 5.2: Example deployment where one `Executable` is bundled with two ARXML files that each contain the description of one `Process`

It is somehow likely that the startup conditions and startup parameters of different `Process`es may be different (in order to achieve a variation of the functionality of the `Executable`).

Therefore, it is necessary to allow for the definition of startup configurations on a per-`Process`-basis.

This aspect is described in section 5.2.

5.2 Startup Configuration

The configuration of startup behavior is an essential part of the application manifest.

[TPS_MANI_01012] Formal modeling of application startup behavior ⌈ The formal modeling of application startup behavior is implemented by means of the aggregation of meta-class ModeDependentStartupConfig in the role Process.modeDependentStartupConfig. ⌋*(RS_MANI_00007)*

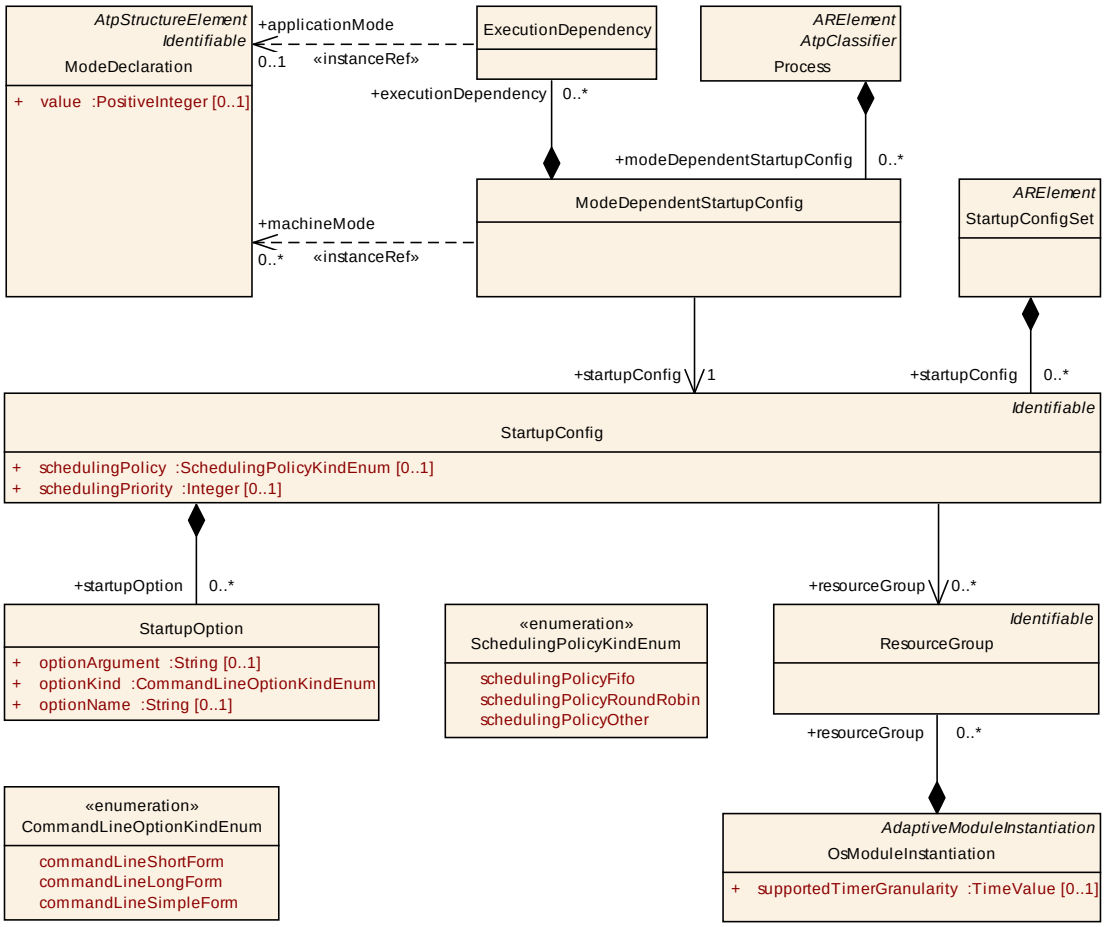

Figure 5.3: Content of a Process

As a consequence of the reference from the ModeDependentStartupConfig to ModeDeclaration the Application Manifest is defined for a specific Machine to which the binary and the Manifest is deployed.

[TPS_MANI_01045] Process.modeDependentStartupConfig that does not refer to a ModeDeclaration ⌈ If one Process.modeDependentStartupConfig does not refer to a ModeDeclaration then this means that one approach to execute the Process does not depend on any ModeDeclaration. ⌋*(RS_MANI_00007)*

It is necessary to specify constraint [constr_1504] to regulate the number of ModeDependentStartupConfig that refer to the same ModeDeclaration in the context of one Process because the resulting startup configuration would be ambiguous.

[constr_1504] Number of `Process.modeDependentStartupConfig` that refer to the same `ModeDeclaration` ⌈ Within the context of a given `Process`, no two `modeDependentStartupConfig` shall refer to the same `ModeDeclaration` in the role `machineMode`. ⌋*()*

In the same spirit, it is necessary to limit (see [constr_1505]) the number of `modeDependentStartupConfig` if there is one `ModeDependentStartupConfig` that does not refer to any `ModeDeclaration` in the context of one `Process`.

That is, the existence of of multiple `modeDependentStartupConfig`s (within the context of one `Process`) with no reference to a `ModeDeclaration` would also create an ambiguous startup configuration.

[constr_1505] Number of `Process.modeDependentStartupConfig` that do not refer to a `ModeDeclaration` ⌈ If a `Process` has **one** `modeDependentStartupConfig` that does not refer to a `ModeDeclaration` then the `Process` shall not aggregate **any other** `modeDependentStartupConfig`. ⌋*()*

[TPS_MANI_01046] Semantics of `ModeDependentStartupConfig.machineMode` ⌈ The `ModeDeclaration`s referenced in the role `ModeDependentStartupConfig.machineMode` shall be considered in a way such that the `ModeDependentStartupConfig` applies if **any** of the referenced `ModeDeclaration`s is active.

In other words, the `ModeDeclaration`s are or-ed for the determination of whether a `ModeDependentStartupConfig` is applicable. ⌋*(RS_MANI_00007)*

Class	**ModeDependentStartupConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Process			
Note	This meta-class defines the startup configuration for the process depending on a collection of machine states. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
executionDependency	ExecutionDependency	*	aggr	This attribute defines that all processes that are referenced via the ExecutionDependency shall be launched and shall reach a certain ApplicationState before the referencing process is started. **Tags:** atp.Status=draft
machineMode	ModeDeclaration	*	iref	This represent the applicable modeDeclaration. **Tags:** atp.Status=draft
startupConfig	StartupConfig	1	ref	Reference to a reusable startup configuration with startup parameters. **Tags:** atp.Status=draft

Table 5.2: ModeDependentStartupConfig

Class	**ModeDeclaration**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this ModeDeclaration.

Table 5.3: ModeDeclaration

[TPS_MANI_01013] Semantics of meta-class `ModeDependentStartupConfig` ⌈ The purpose of meta-class `ModeDependentStartupConfig` to qualify the startup configuration represented by meta-class `StartupConfig` for a specific `ModeDeclaration`.

In other words, the intention is to express that the `StartupConfig` is applicable if the mode machine that controls the startup is in the mode represented by the `ModeDeclaration` referenced in the role `ModeDependentStartupConfig.machineMode`. ⌋*(RS_MANI_00007)*

Please note that the corresponding SWS for the definition of the Execution Manager may refer to *states*. Similar to the situation on the *classic AUTOSAR platform*, the term *mode* used in this document directly corresponds to a *state* on the level of middleware software.

Class	**StartupConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Process			
Note	This meta-class represents a reusable startup configuration for processes.. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
resourceGroup	ResourceGroup	*	ref	Reference to applicable resource groups. **Tags:** atp.Status=draft
schedulingPolicy	SchedulingPolicyKindEnum	0..1	attr	This attribute represents the ability to define the scheduling policy.
schedulingPriority	Integer	0..1	attr	This is the scheduling priority requested by the application itself.
startupOption	StartupOption	*	aggr	Applicable startup options **Tags:** atp.Status=draft

Table 5.4: StartupConfig

Enumeration	**SchedulingPolicyKindEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Process

Note	This meta-class provides a set of settings that allow for the specification of a scheduling policy. For a detailed description of the scheduling policies defined in the context of this meta-class, please refer to The Open Group Base Specifications Issue 7, IEEE Std 1003.1, 2013 Edition. **Tags:** atp.Status=draft
Literal	***Description***
scheduling PolicyFifo	This attribute represents the setting for a FIFO scheduling policy. **Tags:** atp.EnumerationValue=0
scheduling PolicyOther	This attribute represents the setting for a custom scheduling policy. **Tags:** atp.EnumerationValue=2
scheduling PolicyRound Robin	This attribute represents the setting for a round robin scheduling policy **Tags:** atp.EnumerationValue=1

Table 5.5: SchedulingPolicyKindEnum

[TPS_MANI_01061] Requirements on scheduling ⌈ The attributes `StartupConfig.schedulingPolicy` and `StartupConfig.schedulingPriority` make requirements on the scheduling of the process that is created out of launching the `Executable`, i.e. the “outer” scheduling.

The value of these attributes has no direct impact on the behavior of any “inner” scheduling of threads. ⌋*(RS_MANI_00007)*

[TPS_MANI_01014] Semantics of meta-class `StartupConfigSet` ⌈ The existence of a mode-dependent startup procedure implies the existence of a number of `StartupConfig`s within a given project.

Meta-class `StartupConfigSet` is therefore used as some sort of bucket to collect a number of `StartupConfig`s. ⌋*(RS_MANI_00007)*

Class	**StartupConfigSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Process			
Note	Collection of reusable startup configurations for processes. **Tags:** atp.Status=draft; atp.recommendedPackage=StartupConfigSets			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
startupConfig	StartupConfig	*	aggr	Startup configuration that is contained in the StartupConfigSet **Tags:** atp.Status=draft

Table 5.6: StartupConfigSet

A POSIX process is usually started by a parent process, on the *AUTOSAR adaptive platform* this boils down to the `Execution Manager`. It is possible to pass a number of command-line options along with the command to launch the process.

The command-line options are then evaluated and taken into account by the process internally. In principle, command-line options are just a collection of tokens separated by whitespaces.

In most cases, it is not enough to have single tokens passed to the program because then the semantics of an individual token would not be unambiguous.

Therefore, conventions have evolved how to structure the collection of command-line options for launching a program.

In particular, the conventions assume the definition of pairs of command-line tokens where one token takes the role of a qualifier and the other takes the role of the value of that qualifier (example: `-v 1.0` or `--version=1.0`).

Whether or not single tokens can have a meaning depends on the individual program. For the modeling of command-line options this means:

- The model shall be able to describe a pair of command tokens that form a higher semantics in the sense that one qualifies and the other provides a value for that qualifier (example: `-v 1.0` or `--version=1.0`).
- Single tokens may have a fully-specified semantics (example: `-h`).
- It shall also be possible to just pass arguments along without any further markup (example: `../docs/config.txt`)
- Arbitrary number of tokens may appear on the command line of a program

These conclusions, along with the intention of the *AUTOSAR adaptive platform* to model the command line in a detailed way (as opposed to one opaque string), lead to the modeling of meta-class `StartupOption`.

[TPS_MANI_01015] Semantics of meta-class `StartupOption` ⌈ Each `StartupOption` represents a command-line parameter that may (depending on the value of `optionKind`, see [constr_1497] and [constr_1498]) consist of one or two token.

On top of that, it is possible to specify the convention for tokens to be arranged in order to make a valid command-line parameter. The convention is represented by attribute `optionKind`. ⌋*(RS_MANI_00007)*

[TPS_MANI_01059] Different values of `optionKind` within a `StartupConfig.startupOption` ⌈ The attribute `optionKind` may have a different value for each `optionKind` within a given `StartupConfig`. ⌋*(RS_MANI_00007)*

A simpler form of the statement made by [TPS_MANI_01059] is to say that different styles of startup options can be mixed within the context of a `StartupConfig`.

Please note that the usage of the value `commandLineSimpleForm` for attribute `optionKind` implicitly supports the usage of so-called “indirect files” that contain a list

of startup options in order to overcome limitations regarding the total length of startup options on the command line.

In this case the typical strategy is to define a lead-in token that signals the nature of the command-line option, e.g. `@config.txt`.

[constr_1497] Attribute `optionKind` set to `commandLineSimpleForm` ⌈ For any `StartupOption` where attribute `optionKind` is set to `CommandLineOptionKindEnum.commandLineSimpleForm` the attribute `optionName` **shall not** and attribute `optionArgument` **shall** exist. ⌋*()*

[constr_1498] Attribute `optionKind` set to `commandLineShortForm` or `commandLineLongForm` ⌈ For any `StartupOption` where attribute `optionKind` is set to value `CommandLineOptionKindEnum.commandLineShortForm` or `CommandLineOptionKindEnum.commandLineLongForm` the attribute `optionName` **shall** exist. ⌋*()*

Class	**StartupOption**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Process			
Note	This meta-class represents a single startup option consisting of option name and an optional argument. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
optionArgument	String	0..1	attr	This attribute defines option value.
optionKind	CommandLineOptionKindEnum	1	attr	This attribute specifies the style how the command line options appear in the command line.
optionName	String	0..1	attr	This attribute defines option name.

Table 5.7: StartupOption

Enumeration	**CommandLineOptionKindEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Process
Note	This enum defines the different styles how the command line option appear in the command line. **Tags:** atp.Status=draft
Literal	***Description***
commandLineLongForm	Long form of command line option. **Tags:** atp.EnumerationValue=1
commandLineShortForm	Short form of command line option. **Tags:** atp.EnumerationValue=0
commandLineSimpleForm	In this case the command line option does not have any formal structure. Just the value is passed to the program. **Tags:** atp.EnumerationValue=2

Table 5.8: CommandLineOptionKindEnum

Meta-class StartupConfig also supports the specification of a relation to a resource group.

[TPS_MANI_01017] Relation of startup configuration to resource groups ⌈ The modeling of resource groups is possible by means of meta-class ResourceGroup and the association from StartupConfig to ResourceGroup in the role resourceGroup ⌋*(RS_MANI_00007)*

Class	**ResourceGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::AdaptiveModuleImplementation			
Note	This meta-class represents a resource group. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 5.9: ResourceGroup

[TPS_MANI_01041] Startup configuration supports the definition of a launch dependency ⌈ The modeling of startup configuration also supports the definition of a launch dependency, formalized by the meta-class ExecutionDependency that is aggregated by ModeDependentStartupConfig in the role executionDependency.

The ExecutionDependency allows to define a dependency to a process that needs to be in a specific application state before the process that aggregates the ExecutionDependency via ModeDependentStartupConfig is launched. ⌋ *(RS_MANI_00007)*

Please note that, in addition to the explicit definition a launch dependency, there are further ways to specify a dependency between different applications. For example there is an implicit dependency between an application that offers a given service and an application that requires this service.

Class	**ExecutionDependency**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Process			
Note	This element defines an ApplicationState in which a dependent process needs to be before the process that aggregates the ExecutionDependency element can be started. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

application Mode	ModeDeclaration	0..1	iref	This represent the applicable modeDeclaration that represents an ApplicationState. **Tags:** atp.Status=draft

Table 5.10: ExecutionDependency

Obviously, the most elegant approach for startup in this case would be to launch the server application first and then launch the client application.

Service discovery would still work if this implicit dependency is not observed but the inverse launch order may lead to a certain delay until the connection between the server and the client is fully set up.

Small delays may add up and create a significant offset to the overall startup time of an ECU running the *AUTOSAR adaptive platform*. Therefore, it may be advised to observe the implicit launch dependency between applications based on the configuration of service-oriented communication.

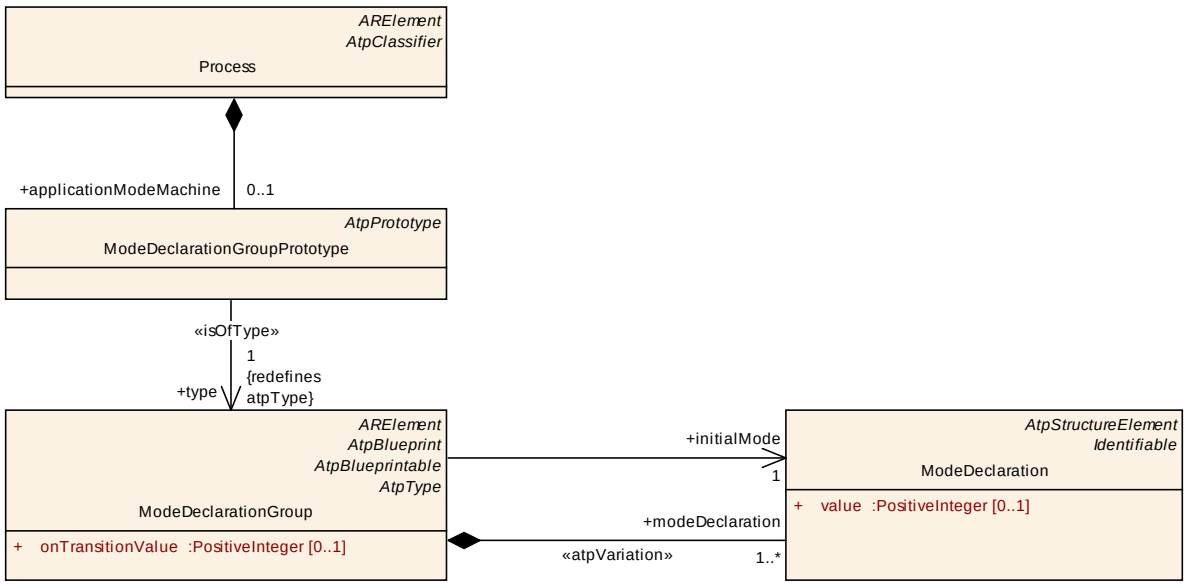

Figure 5.4: Modeling of how `Process` relates to `ModeDeclaration`

However, it may become counterproductive if – in addition to the existence of implicit dependencies – further explicit dependencies are created by means of using the `ModeDependentStartupConfig.executionDependency`.

This may very easily lead to contradictions that could not be resolved conflict-free and may lead to increased startup times.

[constr_1484] Applicability of `ModeDependentStartupConfig.executionDependency` ⌈ The following restrictions apply for the existence of `ModeDependentStartupConfig.executionDependency`:

- The `Process` that contains the `applicationMode` that is referenced by the `ExecutionDependency` shall **only** reference an `Executable` that in turn is ref-

erenced by an AdaptiveAutosarApplication that has the value of attribute category set to PLATFORM_LEVEL (see [TPS_MANI_01009]).

- The Process that aggregates the ExecutionDependency via ModeDependentStartupConfig that refers indirectly to another Process via the applicationMode shall **only** reference an Executable that in turn is referenced by an AdaptiveAutosarApplication that has the value of attribute category set to PLATFORM_LEVEL.

⌋()

In other words: the explicit launch dependency is reserved for platform modules that, in all likelihood, do not use service-oriented communication to communicate with each other.

[constr_3350] Consistent value of category for AdaptiveAutosarApplications referencing an Executable ⌈ All AdaptiveAutosarApplications that reference a specific Executable shall have the value of attribute category set to the same value. ⌋()

5.3 SOME/IP Serialization Properties

The serialization of SOME/IP is based on the ServiceInterface specification. If an AutosarDataPrototype that is used within a ServiceInterface is composite like a structure, union or array then SOME/IP supports the configuration of length fields that will be put in front of the serialized data.

AUTOSAR supports the configuration of such serialization settings on two different levels:

- modeling on ServiceInterface level in the context of an Executable that is valid for all available occurrences of a DataPrototype in the ServiceInterface. This is described in detail in chapter 3.9.
- fine granular modeling on the level of DataPrototypes described in this chapter.

[TPS_MANI_03109] TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface ⌈ The fine granular modeling of TransformationProps in the Application Manifest on the level of DataPrototypes overwrites the TransformationProps settings defined on the level of a ServiceInterface described with the TransformationPropsToServiceInterfaceMapping. ⌋()

[constr_3361] Selective definition of serialization settings ⌈ If a SomeipDataPrototypeTransformationProps is defined for a composite DataPrototype of an element of a ServiceInterface (method, field, event) then SomeipDataPrototypeTransformationProps shall be defined for all other composite DataPrototypes of the ServiceInterface element as well. ⌋()

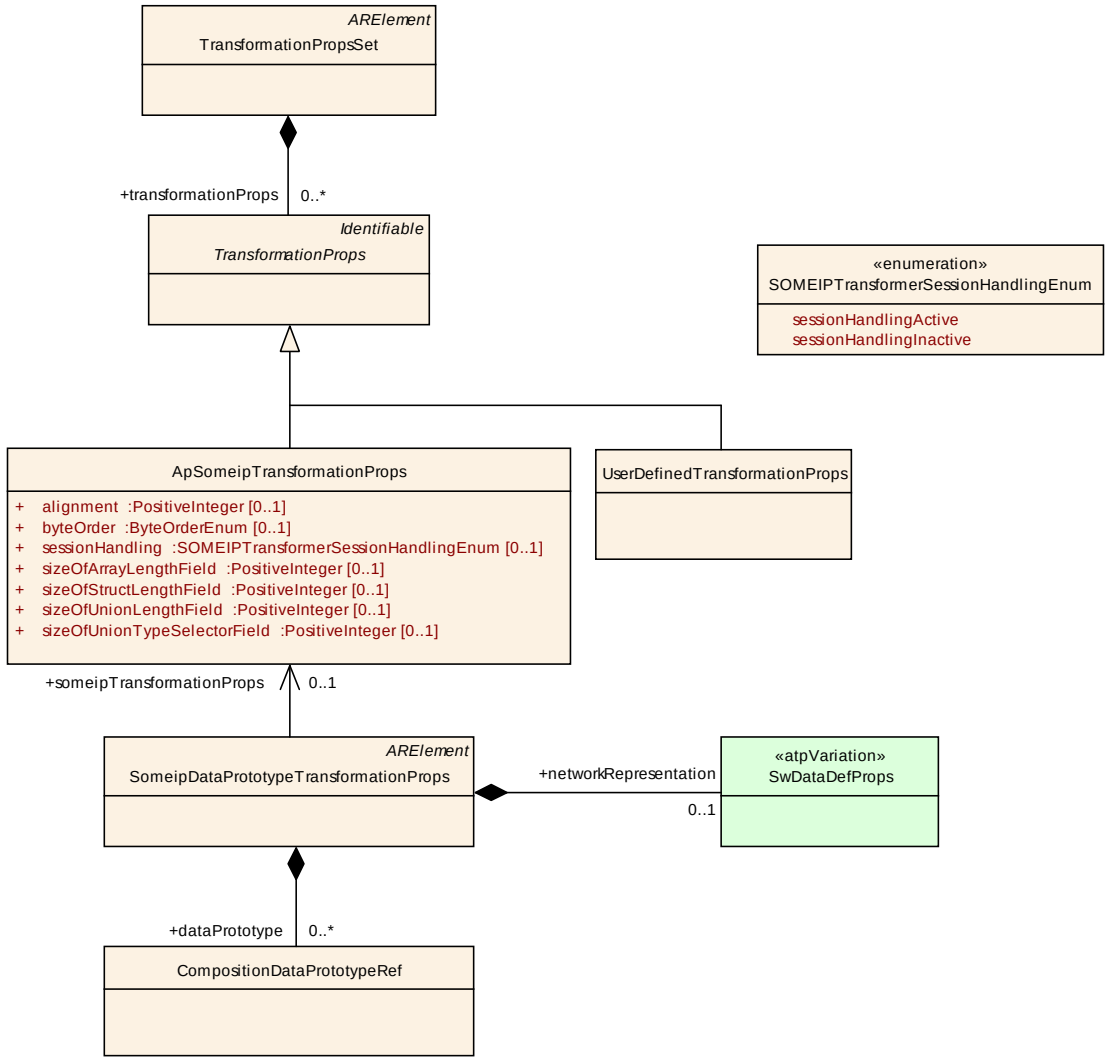

Figure 5.5: Overview about SOME/IP Serialization Properties

Class	**ApSomeipTransformationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransformationConfiguration			
Note	SOME/IP serialization properties. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, TransformationProps			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
alignment	PositiveInteger	0..1	attr	Specifies the alignment of dynamic data in the serialized data stream. The alignment is specified in Bits.
byteOrder	ByteOrderEnum	0..1	attr	Specifies the byte order of data in the serialized data stream.
sessionHandling	SOMEIPTransformerSessionHandlingEnum	0..1	attr	Defines whether the SOME/IP transformer shall use session handling for Sender/Receiver communication.

sizeOfArrayLengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of an Array. It describes the size of the length field (in Bytes) that will be put in front of the Array in the SOME/IP message. In contrast to Classic AUTOSAR this attirbute defines the value for both, fixed-size and dynamic-size arrays.
sizeOfStructLengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of an Struct. It describes the size of the length field (in Bytes) that will be put in front of the Struct in the SOME/IP message.
sizeOfUnionLengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a Union. It describes the size of the length field (in Bytes) that will be put in front of the Union in the SOME/IP message.
sizeOfUnionTypeSelectorField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a Union. It describes the size of the type selector field (in Bytes) that will be put in front of the Union in the SOME/IP message.

Table 5.11: ApSomeipTransformationProps

[TPS_MANI_03070] Size of a length field for a chosen array ⌈ The attribute `sizeOfArrayLengthField` of `ApSomeipTransformationProps` defines the size of a length field generated by SOME/IP in front of an array for which the `SomeipDataPrototypeTransformationProps` is defined, i.e. the array that is referenced within the aggregated `CompositionDataPrototypeRef`. ⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3353] Restriction in usage of `ApSomeipTransformationProps.sizeOfArrayLengthField` ⌈ The value of the attribute `sizeOfArrayLengthField` shall be either 0, 1, 2 or 4. ⌋*()*

[TPS_MANI_03071] Size of a length field for a chosen structure ⌈ The attribute `sizeOfStructLengthField` of `ApSomeipTransformationProps` defines the size of a length field generated by SOME/IP in front of a structure for which the `SomeipDataPrototypeTransformationProps` is defined, i.e. the structure that is referenced within the aggregated `CompositionDataPrototypeRef`. ⌋ *(RS_MANI_00008, RS_MANI_00024)*

[constr_3354] Restriction in usage of `ApSomeipTransformationProps.sizeOfStructLengthField` ⌈ The value of the attribute `sizeOfStructLengthField` shall be either 0, 1, 2 or 4. ⌋*()*

[TPS_MANI_03072] Size of a length field for a chosen union ⌈ The attribute `sizeOfUnionLengthField` of `ApSomeipTransformationProps` defines the size of a length field generated by SOME/IP in front of a union for which the `SomeipDataPrototypeTransformationProps` is defined, i.e. the union that is referenced

within the aggregated CompositionDataPrototypeRef. ⌋(*RS_MANI_00008, RS_MANI_00024*)

[constr_3355] Restriction in usage of ApSomeipTransformationProps.sizeOfUnionLengthField ⌈ The value of the attribute sizeOfUnionLengthField shall be either 0, 1, 2 or 4. ⌋()

[TPS_MANI_03073] Alignment of a dynamic DataPrototype ⌈ The attribute alignment of ApSomeipTransformationProps defines the padding for alignment purposes that will be added by SOME/IP after the serialized data of the variable data length data element for which the SomeipDataPrototypeTransformationProps is defined, i.e. the variable data length DataPrototype that is referenced within the aggregated CompositionDataPrototypeRef. ⌋(*RS_MANI_00008, RS_MANI_00024*)

[constr_3356] Restriction in usage of ApSomeipTransformationProps.alignment ⌈ The value of the attribute alignment shall always be divisible by 8. ⌋()

[TPS_MANI_03074] Size of a type selector field for a chosen union ⌈ The attribute sizeOfUnionTypeSelectorField of ApSomeipTransformationProps defines the size of a type selector field generated by SOME/IP in front of a union for which the SomeipDataPrototypeTransformationProps is defined, i.e. the union that is referenced within the aggregated CompositionDataPrototypeRef. ⌋(*RS_MANI_00008, RS_MANI_00024*)

[constr_3357] Restriction in usage of ApSomeipTransformationProps.sizeOfUnionTypeSelectorField ⌈ The value of the attribute sizeOfUnionTypeSelectorField shall be either 1, 2 or 4. ⌋()

[TPS_MANI_03075] Byte Order of chosen DataPrototype in the serialized data stream ⌈ The attribute byteOrder of ApSomeipTransformationProps defines the Byte Order in front of the DataPrototype in the serialized data stream for which the SomeipDataPrototypeTransformationProps is defined, i.e. the DataPrototype that is referenced within the aggregated CompositionDataPrototypeRef. ⌋(*RS_MANI_00008, RS_MANI_00024*)

Class	**SomeipDataPrototypeTransformationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::TransformationConfiguration			
Note	This meta-class represents the ability to define data transformation props specifically for a SOME/IP serialization for a given DataPrototype. **Tags:** atp.Status=draft; atp.recommendedPackage=SomeipDataPrototype TransformationPropss			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

dataPrototype	CompositionDataPrototypeRef	*	aggr	Collection of DataPrototypes for which the settings in SomeipDataPrototypeTransformationProps are valid. For reuse reasons the SomeipDataPrototypeTransformationProps is able to aggregate several DataPrototypes. **Tags:** atp.Status=draft
networkRepresentation	SwDataDefProps	0..1	aggr	Optional specification of the actual network representation for the referenced primitive DataPrototype. If a network representation is provided then the baseType available in the SwDataDefProps shall be used as input for the serialization/deserialization. If the networkRepresentation is not provided then the baseType of the ImplementationDataType shall be used for the serialization/deserialization. **Tags:** atp.Status=draft
someipTransformationProps	ApSomeipTransformationProps	0..1	ref	This reference represents the ability to define data transformation props specifically for a SOME/IP serialization. **Tags:** atp.Status=draft

Table 5.12: SomeipDataPrototypeTransformationProps

Class	**CompositionDataPrototypeRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class represents the ability to refer to an AUTOSAR DataPrototype in the context of a CompositionSwComponentType. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataPrototype	DataPrototype	0..1	iref	This attribute shall exist if the InstanceRef points to a DataPrototype typed by an ApplicationDataType. **Tags:** atp.Status=draft
elementInImplDatatype	ElementInImplementationDatatypeInstanceRef	0..1	aggr	This attribute shall exist if the InstanceRef points to a DataPrototype typed by an ImplementationDataType. **Tags:** atp.Status=draft

Table 5.13: CompositionDataPrototypeRef

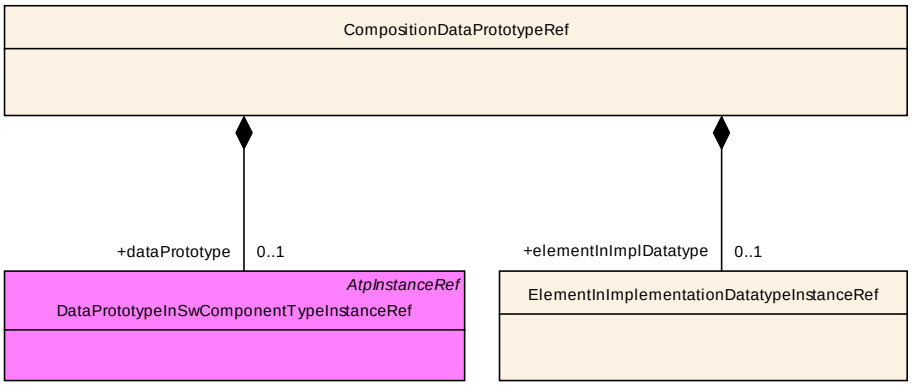

Figure 5.6: Reference to a DataPrototype in the context of a CompositionSwComponent-Type that is typed by an ApplicationDataType or by an ImplementationDataType

The usage of the `networkRepresentation` is explained in more detail in the System Template [9] in [TPS_SYST_02136] and [TPS_SYST_02137].

6 Service Instance Manifest

6.1 Service Interface Deployment

The different meta-class specializations of `ServiceInterfaceDeployment` define a binding of a `ServiceInterface` to a middleware transport layer.

This chapter describes the usage of the `ServiceInterfaceDeployment` in different bindings that are supported by AUTOSAR.

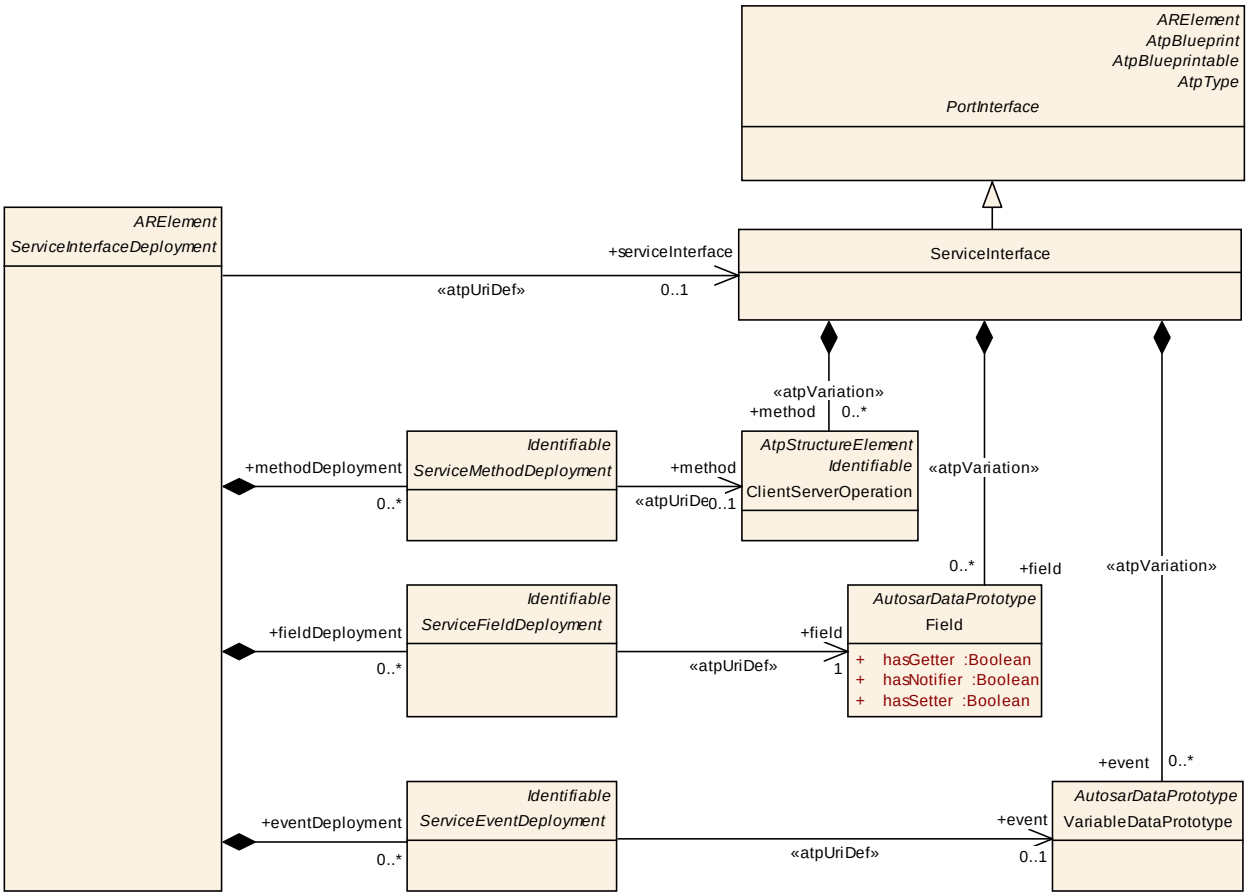

Figure 6.1: Deployment-related modeling of ServiceInterface

[TPS_MANI_03036] `ServiceInterface` deployment to a middleware transport layer ⌈ The `ServiceInterfaceDeployment` meta-class provides the ability to map a `ServiceInterface` to a middleware transport layer that is represented by a concrete class that is derived from the abstract `ServiceInterfaceDeployment` meta-class. ⌋*(RS_MANI_00008)*

Class	**ServiceInterfaceDeployment (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	Middleware transport layer specific configuration settings for the ServiceInterface and all contained ServiceInterface elements. **Tags:** atp.Status=draft			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
eventDeployment	ServiceEventDeployment	*	aggr	Middleware transport layer specific configuration settings for an Event that is defined in the ServiceInterface. **Tags:** atp.Status=draft
fieldDeployment	ServiceFieldDeployment	*	aggr	Middleware transport layer specific configuration settings for a Field that is defined in the ServiceInterface. **Tags:** atp.Status=draft
methodDeployment	ServiceMethodDeployment	*	aggr	Middleware transport layer specific configuration settings for a method that is defined in the ServiceInterface. **Tags:** atp.Status=draft
serviceInterface	ServiceInterface	0..1	ref	Reference to a ServiceInterface that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft

Table 6.1: ServiceInterfaceDeployment

[TPS_MANI_03037] Purpose of `ServiceMethodDeployment` ⌈ The `ServiceMethodDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for a `method` that is defined in the context of a `ServiceInterface`. ⌋*(RS_MANI_00008)*

[constr_3300] Allowed `ServiceMethodDeployment.method` references ⌈ The `ClientServerOperation` that is referenced by `ServiceMethodDeployment` in the role `method` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceMethodDeployment`. ⌋*()*

[TPS_MANI_03038] Purpose of `ServiceEventDeployment` ⌈ The `ServiceEventDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for an `event` that is defined in the context of a `ServiceInterface`. ⌋*(RS_MANI_00008)*

[constr_3301] Allowed `ServiceEventDeployment.event` references ⌈ The `VariableDataPrototype` that is referenced by `ServiceEventDeployment` in the role `event` shall be defined in the context of a `ServiceInterface` that is referenced

by the ServiceInterfaceDeployment in the role serviceInterface that contains the ServiceEventDeployment. ⌋()

[TPS_MANI_03039] Purpose of ServiceFieldDeployment ⌈ The ServiceFieldDeployment meta-class provides the ability to define middleware transport layer specific configuration settings relevant for a field that is defined in the context of a ServiceInterface. ⌋(*RS_MANI_00008*)

[constr_3302] Allowed ServiceFieldDeployment.field references ⌈ The Field that is referenced by ServiceFieldDeployment in the role field shall be defined in the context of a ServiceInterface that is referenced by the ServiceInterfaceDeployment in the role serviceInterface that contains the ServiceFieldDeployment. ⌋()

Class	**ServiceMethodDeployment (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	This abstract meta-class represents the ability to specify a deployment of a Method to a middleware transport layer. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
method	ClientServerOperation	0..1	ref	Reference to a method that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft

Table 6.2: ServiceMethodDeployment

Class	**ServiceEventDeployment (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	This abstract meta-class represents the ability to specify a deployment of an Event to a middleware transport layer. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
event	VariableDataPrototype	0..1	ref	Reference to an Event that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft

Table 6.3: ServiceEventDeployment

Class	**ServiceFieldDeployment (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	This abstract meta-class represents the ability to specify a deployment of a Field to a middleware transport layer. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
field	Field	1	ref	Reference to a Field that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft

Table 6.4: ServiceFieldDeployment

6.1.1 SOME/IP Service Interface Deployment

This chapter describes the SOME/IP deployment of a `ServiceInterface`.

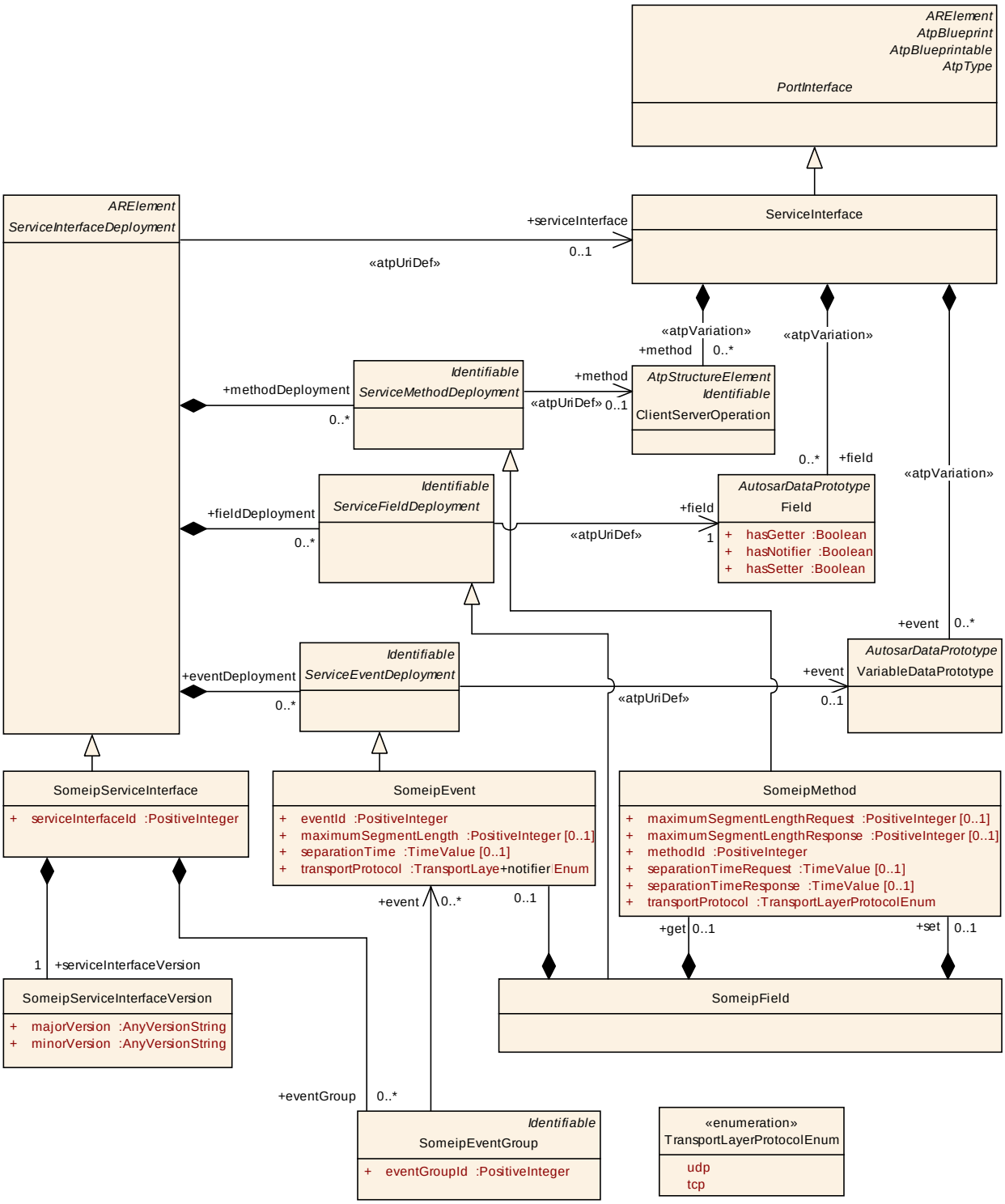

Figure 6.2: SOME/IP deployment of ServiceInterface

[TPS_MANI_03040] SOME/IP ServiceInterface binding ⌈ The `SomeipServiceInterface` meta-class provides the ability to bind a `ServiceInterface` to SOME/IP and to assign a SOME/IP Service identifier to the `ServiceInterface` with the `serviceInterfaceId` attribute. ⌋*(RS_MANI_00024)*

[TPS_MANI_03041] Definition of SOME/IP EventGroups ⌈ The `SomeipServiceInterface.eventGroup` allows to define SOME/IP *EventGroups* that are included in the SOME/IP Service and provide a logical grouping of events and notification events used for publish/subscribe handling. ⌋*(RS_MANI_00024)*

[constr_3304] Value of attribute `SomeipEventGroup.eventGroupId` shall be unique ⌈ The value of `eventGroupId` shall be unique in the in the context of the enclosing `SomeipServiceInterface`. ⌋*()*

[TPS_MANI_03042] Definition of SOME/IP Service Version ⌈ The `SomeipServiceInterface.serviceInterfaceVersion` allows to define a major and a minor version for the SOME/IP Service. ⌋*(RS_MANI_00024)*

[constr_3303] `ANY` not allowed for `SomeipServiceInterface.serviceInterfaceVersion` ⌈ The value `ANY` is not allowed for the `majorVersion` and `minorVersion` of the `SomeipServiceInterface.serviceInterfaceVersion`. ⌋*()*

Class	**SomeipServiceInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a ServiceInterface. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInterfaceDeployments			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, ServiceInterfaceDeployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	*	aggr	SOME/IP EventGroups that are defined within the SOME/IP ServiceClass. **Tags:** atp.Status=draft
serviceInterfaceId	PositiveInteger	1	attr	Unique Identifier that identifies the ServiceInterface in SOME/IP. This Identifier is sent as Service ID in SOME/IP Service Discovery messages.
serviceInterfaceVersion	SomeipServiceInterfaceVersion	1	aggr	The SOME/IP major and minor Version of the Service. **Tags:** atp.Status=draft

Table 6.5: SomeipServiceInterface

Class	**SomeipServiceInterfaceVersion**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This meta-class represents the ability to describe a version of a SOME/IP ServiceInterface. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
majorVersion	AnyVersionString	1	attr	Major Version of the ServiceInterface. Value can be set to a number that represents the Major Version of the searched service or to ANY.

minorVersion	AnyVersionString	1	attr	Minor Version of the ServiceInterface. Value can be set to a number that represents the Minor Version of the searched service or to ANY.

Table 6.6: SomeipServiceInterfaceVersion

Class	**SomeipEventGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	Grouping of events and notification events inside a ServiceInterface in order to allow subscriptions. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
event	SomeipEvent	*	ref	Reference to an event that is part of the EventGroup. **Tags:** atp.Status=draft
eventGroupId	PositiveInteger	1	attr	Unique Identifier that identifies the EventGroup in SOME/IP. This Identifier is sent as Eventgroup ID in SOME/IP Service Discovery messages.

Table 6.7: SomeipEventGroup

[TPS_MANI_03043] SOME/IP `VariableDataPrototype` binding ⌈ The `SomeipEvent` meta-class provides the ability to bind a `VariableDataPrototype` to SOME/IP and to assign a SOME/IP Event identifier to the `event` with the `eventId` attribute. ⌋*(RS_MANI_00024)*

[constr_3305] Value of attribute `SomeipEvent.eventId` shall be unique ⌈ The value of `eventId` shall be unique in the in the context of the enclosing `SomeipServiceInterface` and shall also not overlap with any defined `methodId` used in the context of the enclosing `SomeipServiceInterface`. ⌋*()*

[TPS_MANI_03050] Usage of `SomeipEvent.transportProtocol` ⌈ The value of `SomeipEvent.transportProtocol` defines over which Transport Layer Protocol the `SomeipEvent.event` is provided. ⌋*(RS_MANI_00024)*

[constr_3307] `SomeipEvent.transportProtocol` setting to `udp` and the impact on `ProvidedSomeipServiceInstance`s ⌈ If `SomeipEvent.transportProtocol` is set to `udp` then each `ProvidedSomeipServiceInstance` that refers the `SomeipServiceInterface` in the role `serviceInterface` shall only be mapped to a `Machine` with a `SomeipServiceInstanceToMachineMapping` that aggregates a `portConfig` with a configured `udpPort`. ⌋*()*

[constr_3308] `SomeipEvent.transportProtocol` setting to `tcp` and the impact on `ProvidedSomeipServiceInstance`s ⌈ If `SomeipEvent.transportProtocol` is set to `tcp` then each `ProvidedSomeipServiceInstance` that refers the `SomeipServiceInterface` in the role `serviceInterface` shall only be mapped

to a Machine with a SomeipServiceInstanceToMachineMapping that aggregates a portConfig with a configured tcpPort. ⌋()

[TPS_MANI_03067] SOME/IP segmentation of udp SomeipEvents ⌈ If the maximumSegmentLength is set to a value and the data length is larger than maximumSegmentLength then SOME/IP shall segment the SomeipEvent into several packets and transmit them over the network.

The sender shall wait the separationTime between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipEvent. ⌋(*RS_MANI_00024*)

[constr_3351] SOME/IP segmentation allowed for udp SomeipEvents ⌈ Attribute SomeipEvent.maximumSegmentLength shall only be used if the value of attribute SomeipEvent.transportProtocol is set to udp. ⌋()

Class	**SomeipEvent**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for an Event. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceEventDeployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
eventId	PositiveInteger	1	attr	Unique Identifier within a ServiceInterface that identifies the Event in SOME/IP. This Identifier is sent as part of the Message ID in SOME/IP messages.
maximumSegmentLength	PositiveInteger	0..1	attr	This attribute describes the length in bytes of the SOME/IP segment. This includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLength then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
separationTime	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments.
transportProtocol	TransportLayerProtocolEnum	1	attr	This attribute defines over which Transport Layer Protocol this event is intended to be sent.

Table 6.8: SomeipEvent

[TPS_MANI_03044] SOME/IP ClientServerOperation binding ⌈ The SomeipMethod meta-class provides the ability to bind a ClientServerOperation to SOME/IP and to assign a SOME/IP Method identifier to the method with the methodId attribute. ⌋(*RS_MANI_00024*)

[constr_3306] Value of attribute `methodId` shall be unique per `SomeipServiceInterface` ⌈ The value of `methodId` shall be unique in the in the context of the enclosing `SomeipServiceInterface` and shall also not overlap with any defined `eventId` used in the context of the enclosing `SomeipServiceInterface`. ⌋*()*

[TPS_MANI_03051] Usage of `SomeipMethod.transportProtocol` ⌈ The value of `SomeipMethod.transportProtocol` defines over which Transport Layer Protocol this method is provided. ⌋*(RS_MANI_00024)*

[constr_3309] `SomeipMethod.transportProtocol` setting to `udp` and the impact on `ProvidedSomeipServiceInstance`s ⌈ If `SomeipMethod.transportProtocol` is set to `udp` then each `ProvidedSomeipServiceInstance` that refers the `SomeipServiceInterface` in the role `serviceInterface` shall only be mapped to a `Machine` with a `SomeipServiceInstanceToMachineMapping` that aggregates a `portConfig` with a configured `udpPort`. ⌋*()*

[constr_3310] `SomeipMethod.transportProtocol` setting to `tcp` and the impact on `ProvidedSomeipServiceInstance`s ⌈ If `SomeipMethod.transportProtocol` is set to `tcp` then each `ProvidedSomeipServiceInstance` that refers the `SomeipServiceInterface` in the role `serviceInterface` shall only be mapped to a `Machine` with a `SomeipServiceInstanceToMachineMapping` that aggregates a `portConfig` with a configured `tcpPort`. ⌋*()*

[TPS_MANI_03068] SOME/IP segmentation of `SomeipMethod` Calls ⌈ If the `maximumSegmentLengthRequest` is set to a value and the data length is larger than `maximumSegmentLengthRequest` then SOME/IP shall segment the `SomeipMethod` Call-Message into several packets and transmit them over the network.

The sender shall wait the `separationTimeRequest` between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original `SomeipMethod` Call-Message. ⌋*(RS_MANI_00024)*

[TPS_MANI_03069] SOME/IP segmentation of `SomeipMethod` Responses ⌈ If the `maximumSegmentLengthResponse` is set to a value and the data length is larger than `maximumSegmentLengthResponse` then SOME/IP shall segment the `SomeipMethod` Response-Message into several packets and transmit them over the network.

The sender shall wait the `separationTimeResponse` between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original `SomeipMethod` Response-Message. ⌋*(RS_MANI_00024)*

[constr_3352] SOME/IP segmentation allowed for udp `SomeipMethod`s ⌈ `SomeipMethod.maximumSegmentLengthRequest` and `SomeipMethod.maximumSegmentLengthResponse` shall only be used if `SomeipMethod.transportProtocol` is set to udp. ⌋*()*

Class	**SomeipMethod**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a Method. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceMethodDeployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
maximumSegmentLengthRequest	PositiveInteger	0..1	attr	This attribute describes the length in bytes of one SOME/IP segment into which the Method Call Message will be divided. This length field includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLengthRequest then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
maximumSegmentLengthResponse	PositiveInteger	0..1	attr	This attribute describes the length in bytes of one SOME/IP segment into which the Method Return Message will be divided. This length field includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLengthResponse then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.
methodId	PositiveInteger	1	attr	Unique Identifier within a ServiceInterface that identifies the Method in SOME/IP. This Identifier is sent as part of the Message ID in SOME/IP messages.
separationTimeRequest	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments into which the Method Call Message will be divided.
separationTimeResponse	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments into which the Method Return Message will be divided.
transportProtocol	TransportLayerProtocolEnum	1	attr	This attribute defines over which Transport Layer Protocol this method is intended to be sent.

Table 6.9: SomeipMethod

Class	**SomeipServiceInstanceToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceMapping			
Note	This meta-class allows to map SomeipServiceInstances to a CommunicationConnector of a Machine. In this step the network configuration (IP Address, Transport Protocol, Port Number) for the ServiceInstance is defined. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInstanceToMachine Mappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, ServiceInstanceToMachineMapping			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
ipv4MulticastIpAddress	Ip4AddressString	0..1	attr	Multicast IPv4 Address that is transmitted in the EventGroupSubscribeAck message for all available EventGroups that are available in the ProvidedSomeipServiceInstance.
ipv6MulticastIpAddress	Ip6AddressString	0..1	attr	Multicast IPv6 Address that is transmitted in the EventGroupSubscribeAck message for all available EventGroups that are available in the ProvidedSomeipServiceInstance.
portConfig	ServiceInstancePortConfig	*	aggr	Transport Layer Protocol configuration for a ServiceInstance that is mapped to a CommunicationConnector of a Machine. **Tags:** atp.Status=draft

Table 6.10: SomeipServiceInstanceToMachineMapping

[TPS_MANI_03057] SOME/IP `Field` binding ⌈ The `SomeipField` meta-class provides the ability to bind a `Field` to SOME/IP.

If the `Field` contains a notifier (`hasNotifier` = true) it is possible to assign a SOME/IP notifier identifier to the `field` by setting the value of attribute `SomeipField.notifier.eventId`.

If the `Field` contains a getter method (`hasGetter` = true) it is possible to assign a SOME/IP notifier identifier to the `field` by setting the value of attribute `SomeipField.get.methodId`.

If the `Field` contains a setter method (`hasSetter` = true) it is possible to assign a SOME/IP notifier identifier to the `field` by setting the value of attribute `SomeipField.set.methodId` ⌋*(RS_MANI_00024)*

Please note that each `methodId` and each `eventId` of a `SomeipField` shall be unique in the context of a `ServiceInterface` as defined in [constr_3306] and [constr_3305].

Class	**SomeipField**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a Field. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceFieldDeployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
get	SomeipMethod	0..1	aggr	This aggregation represents the setting of the get method. **Tags:** atp.Status=draft
notifier	SomeipEvent	0..1	aggr	This aggregation represents the settings of the notifier. **Tags:** atp.Status=draft
set	SomeipMethod	0..1	aggr	This aggregation represents the settings of the set method **Tags:** atp.Status=draft

Table 6.11: SomeipField

[constr_3362] `SomeipEvent`s aggregated by a `SomeipField` ⌈ A `SomeipEvent` that is aggregated by a `SomeipField` in the role `notifier` shall not reference a `VariableDataPrototype` in the role `event`. ⌋*()*

[constr_3363] `SomeipMethod`s aggregated by a `SomeipField` ⌈ A `SomeipMethod` that is aggregated by a `SomeipField` in the role `get` or `set` shall not reference a `ClientServerOperation` in the role `method`. ⌋*()*

6.1.2 User Defined Service Interface

This chapter describes a user defined deployment of a `ServiceInterface` to a middleware technology that is not standardized by AUTOSAR. Such `UserDefinedServiceInterfaceDeployment` can for example also be used to describe a machine local IPC communication.

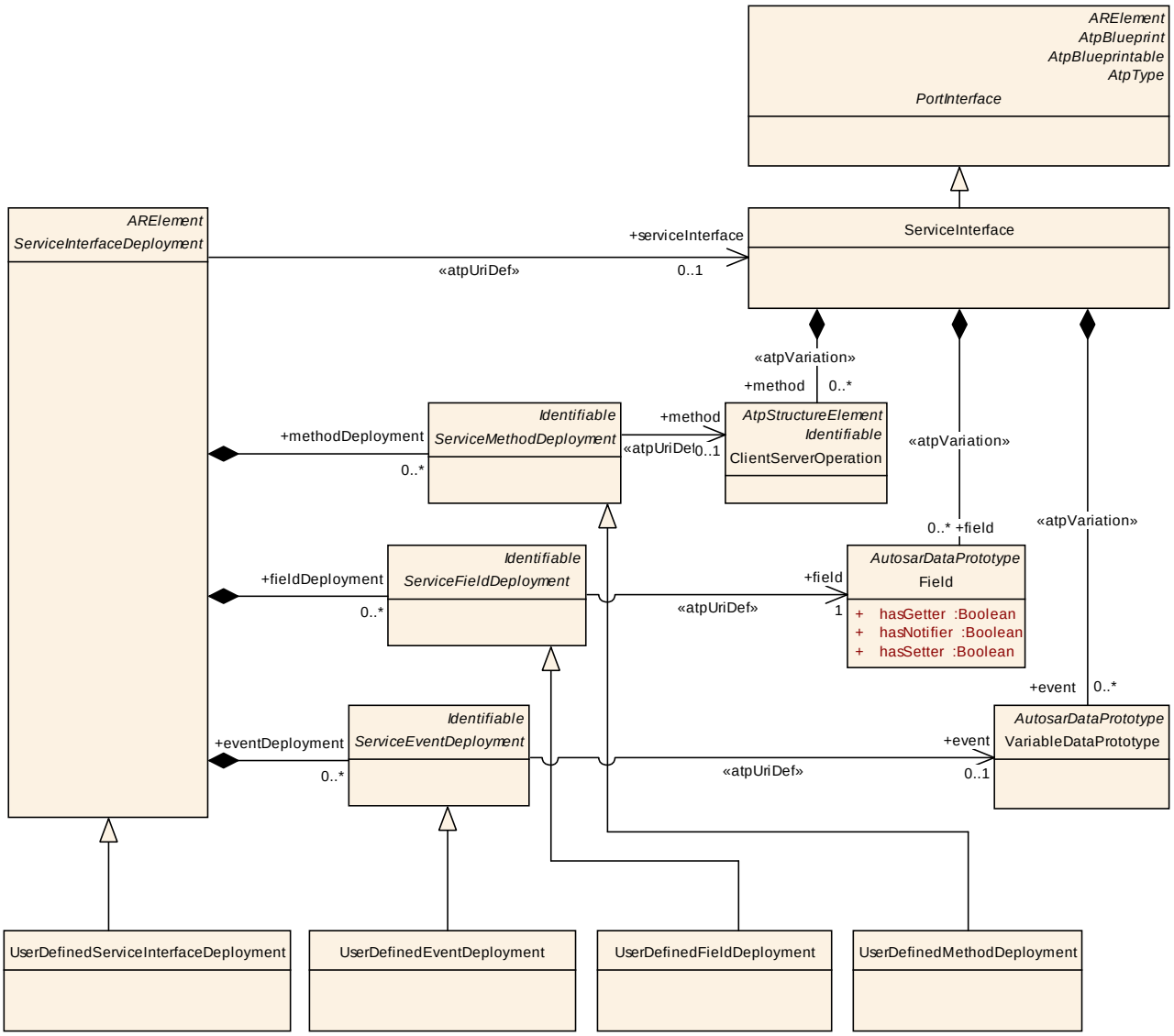

Figure 6.3: User defined deployment of ServiceInterface

[TPS_MANI_03045] UserDefined `ServiceInterface` binding ⌈ The `UserDefinedServiceInterfaceDeployment` meta-class provides the ability to bind a `ServiceInterface` that is referenced in the role `serviceInterface` to a middleware technology that is not standardized by AUTOSAR. ⌋*(RS_MANI_00014)*

Please note that `UserDefinedServiceInterfaceDeployment` is `Identifiable` and therefore is able to describe special data (sdg) which is not represented by the standard model.

Class	**UserDefinedServiceInterfaceDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a ServiceInterface. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInterfaceDeployments			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, ServiceInterfaceDeployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 6.12: UserDefinedServiceInterfaceDeployment

[TPS_MANI_03046] User defined `VariableDataPrototype` binding ⌈ The `UserDefinedEventDeployment` meta-class provides the ability to bind a `VariableDataPrototype` that is referenced in the role `event` to a middleware technology that is not standardized by AUTOSAR. ⌋*(RS_MANI_00014)*

Please note that `UserDefinedEventDeployment` is `Identifiable` and therefore is able to describe special data (sdg) which is not represented by the standard model.

Class	**UserDefinedEventDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for an Event. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceEvent Deployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 6.13: UserDefinedEventDeployment

[TPS_MANI_03047] User defined `ClientServerOperation` binding ⌈ The `UserDefinedMethodDeployment` meta-class provides the ability to bind a `ClientServerOperation` that is referenced in the role `method` to a middleware technology that is not standardized by AUTOSAR. ⌋*(RS_MANI_00014)*

Please note that `UserDefinedMethodDeployment` is `Identifiable` and therefore is able to describe special data (sdg) which is not represented by the standard model.

Class	**UserDefinedMethodDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a Method. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceMethod Deployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 6.14: UserDefinedMethodDeployment

[TPS_MANI_03048] User defined `Field` binding ⌈ The `UserDefinedFieldDeployment` meta-class provides the ability to bind a `Field` that is referenced in the role `field` to a middleware technology that is not standardized by AUTOSAR. ⌋ *(RS_MANI_00014)*

Please note that `UserDefinedFieldDeployment` is `Identifiable` and therefore is able to describe special data (sdg) which is not represented by the standard model.

Class	**UserDefinedFieldDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a Field. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, ServiceFieldDeployment			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 6.15: UserDefinedFieldDeployment

6.2 Service Instance Deployment

An `AdaptivePlatformServiceInstance` makes the functionality of a `ServiceInterface` available on the *AUTOSAR adaptive platform*. Several `AdaptivePlatformServiceInstance`s may be set up for the same `ServiceInterface`. They deliver the same functionality, but for different purposes and/or to different users.

The `ProvidedApServiceInstance` represents a provider that offers the functionality of a `ServiceInterface` with particular properties. Clients that are represented by the `RequiredApServiceInstance` observe offers and choose a provider with respect to service properties.

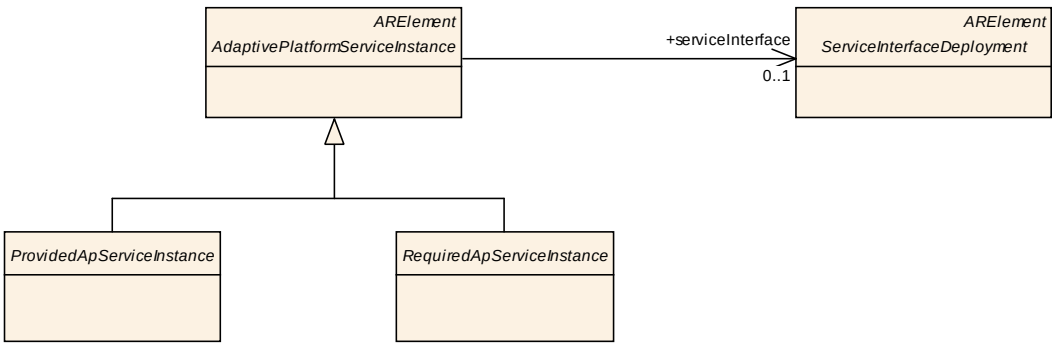

Figure 6.4: Modeling of the `AdaptivePlatformServiceInstance`

Note that the abstract meta-class `AdaptivePlatformServiceInstance` is derived from `ARElement`. This means that all meta-classes derived from `AdaptivePlatformServiceInstance` can be declared on the M1 level as part of an `ARPackage` and thus can be used in several different Manifest descriptions.

Class	**AdaptivePlatformServiceInstance (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a service instance in an abstract way. **Tags:** atp.Status=draft			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
serviceInterface	ServiceInterfaceDeployment	0..1	ref	Reference to a ServiceInterfaceDeployment that identifies the ServiceInterface that is represented by the ServiceInstance. **Tags:** atp.Status=draft

Table 6.16: AdaptivePlatformServiceInstance

Class	**RequiredApServiceInstance (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in an abstract way. **Tags:** atp.Status=draft			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 6.17: RequiredApServiceInstance

Class	**ProvidedApServiceInstance (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in an abstract way. **Tags:** atp.Status=draft			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 6.18: ProvidedApServiceInstance

There are two alternative ways to relate a `AdaptivePlatformServiceInstance` with a `Machine` as described in [TPS_MANI_03000] and [TPS_MANI_03001]. Figure Figure 6.5 shows both approaches in an example.

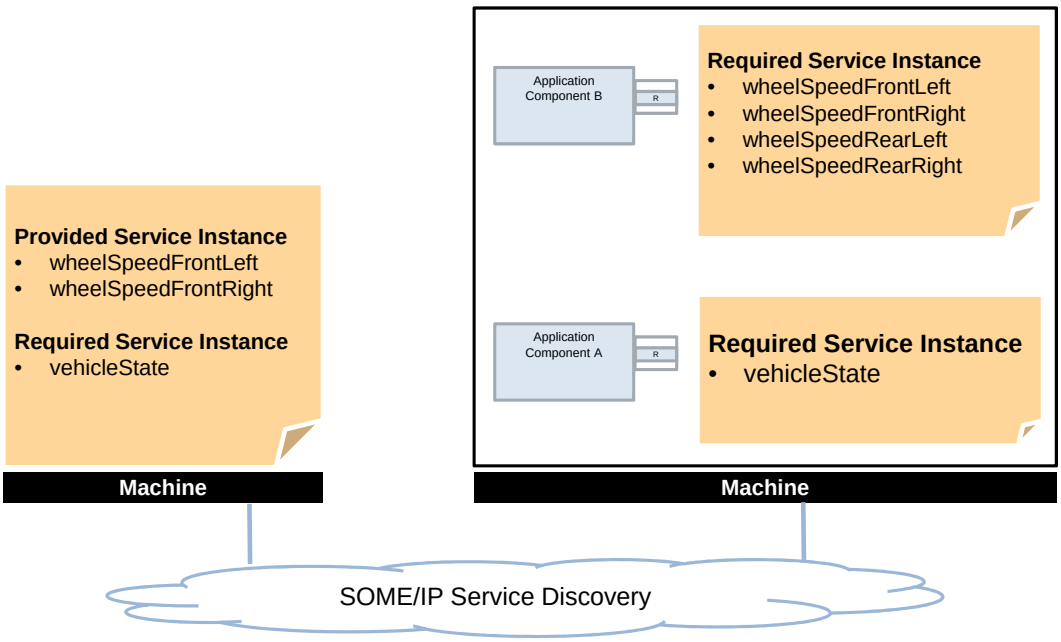

Figure 6.5: Different approaches for ServiceInstanceMapping

[TPS_MANI_03001] Mapping of `AdaptivePlatformServiceInstance` to a `Machine` ⌈ `ServiceInstanceToMachineMapping` is used to assign an `AdaptivePlatformServiceInstance` to (via a `CommunicationConnector`) a `Machine`. This allows to define a “black box” machine view without any assumption on the application software but with all necessary information to configure the communication (e.g. SOME/IP). ⌋*(RS_MANI_00009)*

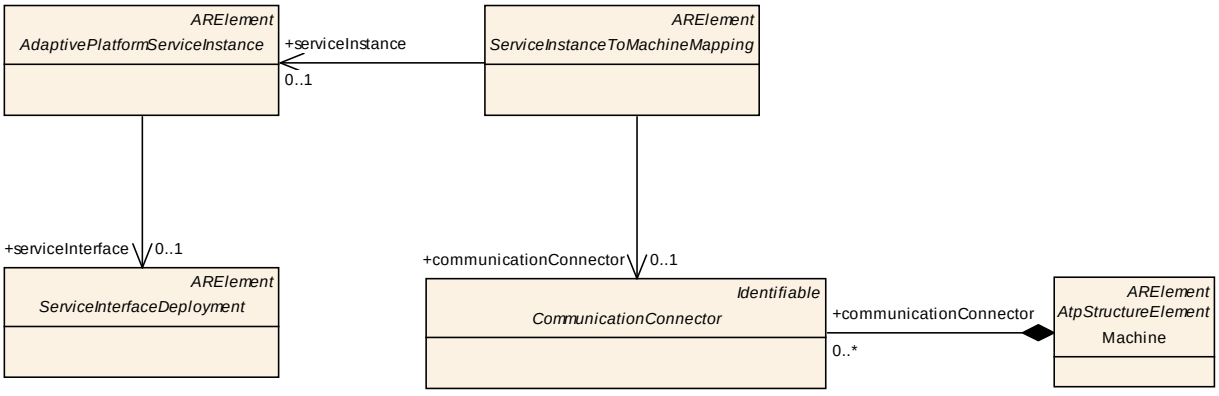

Figure 6.6: ServiceInstanceToMachineMapping

Class	**ServiceInstanceToMachineMapping (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceMapping			
Note	This meta-class represents the ability to map a AdaptivePlatformServiceInstance to a CommunicationConnector of a Machine. **Tags:** atp.Status=draft			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
communicationConnector	CommunicationConnector	0..1	ref	Reference to the Machine to which the ServiceInstance is mapped. **Tags:** atp.Status=draft
serviceInstance	AdaptivePlatformServiceInstance	0..1	ref	Reference to a ServiceInstance that is mapped to the Machine. **Tags:** atp.Status=draft

Table 6.19: ServiceInstanceToMachineMapping

Please note that the `ServiceInstanceToMachineMapping` defines the base for the communication and shall always be defined. The usage of `ServiceInstanceToApplicationEndpointMapping` however is optional and refines the configuration by linking the Software Components with the middleware.

[constr_3297] `SomeipServiceInstanceToMachineMapping` only supports a single Address Family ⌈ A `SomeipServiceInstanceToMachineMapping` shall only support a single Address Family, i.e. either IPv4 or IPv6. The address family shall be consistent with the `Ipv4Configuration`/`Ipv6Configuration` of the `NetworkEndpoint` referenced by the `EthernetCommunicationConnector` that is referenced by the `SomeipServiceInstanceToMachineMapping` in the role `communicationConnector`. ⌋*()*

[constr_3291] `SomeipServiceInstanceToMachineMapping.portConfig` aggregation restriction ⌈ A `SomeipServiceInstanceToMachineMapping` is not allowed to aggregate more than one `ServiceInstancePortConfig` in the role `portConfig`. ⌋*()*

[TPS_MANI_03000] Mapping of `AdaptivePlatformServiceInstance` to `PortPrototype`s ⌈ `ServiceInstanceToApplicationEndpointMapping` is used to assign an `AdaptivePlatformServiceInstance` to a `PortPrototype` of a `SwComponentType`. This allows to define how specific `PortPrototype`s of a Software Component are represented in the middleware in terms of the service configuration. ⌋*(RS_MANI_00011)*

In other words, the “outside” appearance of a `PortPrototype` from the middleware point of view is the `AdaptivePlatformServiceInstance`, resp. the concrete subclasses `RequiredApServiceInstance` and `ProvidedApServiceInstance`.

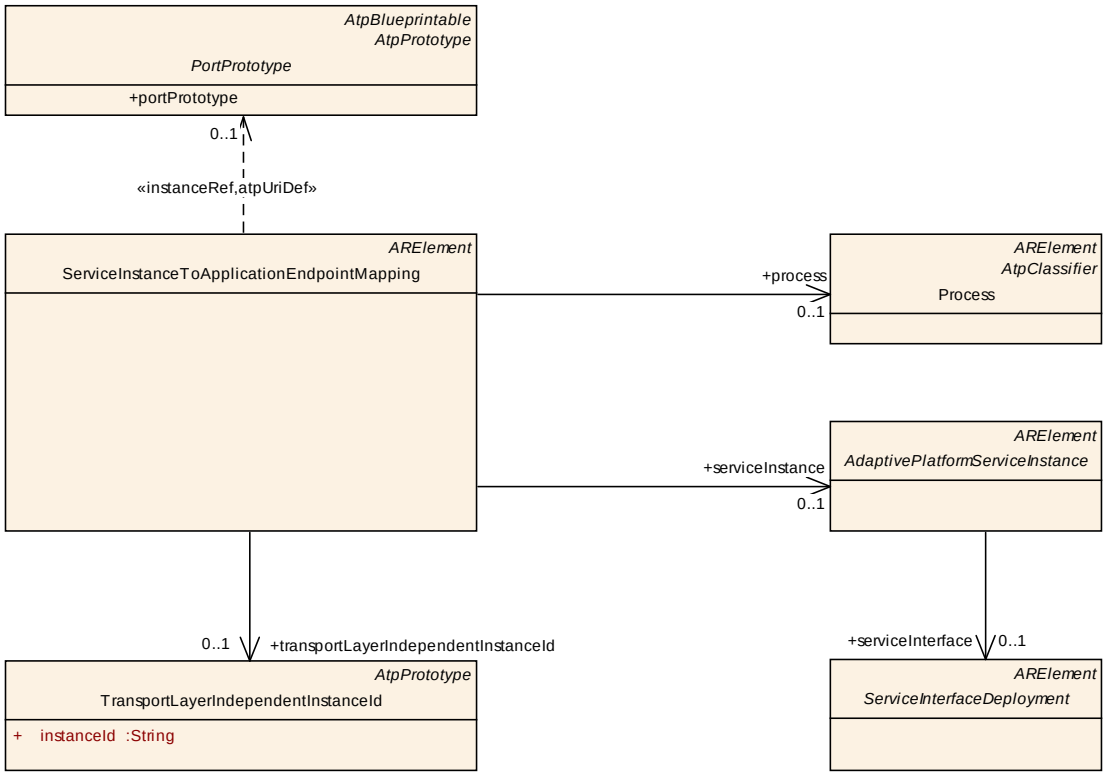

Figure 6.7: ServiceInstanceToPortPrototypeMapping

Class	**ServiceInstanceToApplicationEndpointMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceMapping			
Note	This meta-class represents the ability to assign a transport layer dependent ServiceInstance to an ApplicationEndpoint. The ApplicationEndpoint is either represented by a PortPrototype of a Software Component defined in the context of an Executable or as an alternative as a TransportLayerIndependentInstanceId. With this mapping it is possible to define how specific ApplicationEndpoints are represented in the middleware in terms of service configuration. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInstanceToApplication EndpointMappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
portPrototype	PortPrototype	0..1	iref	Reference to a specific PortPrototypes that represents the ServiceInstance. **Tags:** atp.Status=draft
process	Process	0..1	ref	Reference to the Process in which the Executable that contains the SoftwareComponent and the referenced PortPrototype is executed. **Tags:** atp.Status=draft

serviceInstance	AdaptivePlatformServiceInstance	0..1	ref	Reference to a ServiceInstance that is represented in the Software Component by the mapped group of PortPrototypes. **Tags:** atp.Status=draft
transportLayerIndependentInstanceId	TransportLayerIndependentInstanceId	0..1	ref	Reference to a specific instanceId that represents the ApplicationEndpoint that is independent of the Transport Layer. **Tags:** atp.Status=draft

Table 6.20: ServiceInstanceToApplicationEndpointMapping

Meta-classes `ProvidedApServiceInstance` and `RequiredApServiceInstance` are abstract and this allows for using specific derived classes that fit the underlying middleware (e.g. SOME/IP). The following sub-chapters will detail the supported specializations.

6.2.1 SOME/IP Service Instance Deployment

In the case of SOME/IP used as the middleware the derived meta-classes are `ProvidedSomeipServiceInstance` resp. `RequiredSomeipServiceInstance`. These meta-classes also carry attributes that apply for the service discovery on SOME/IP.

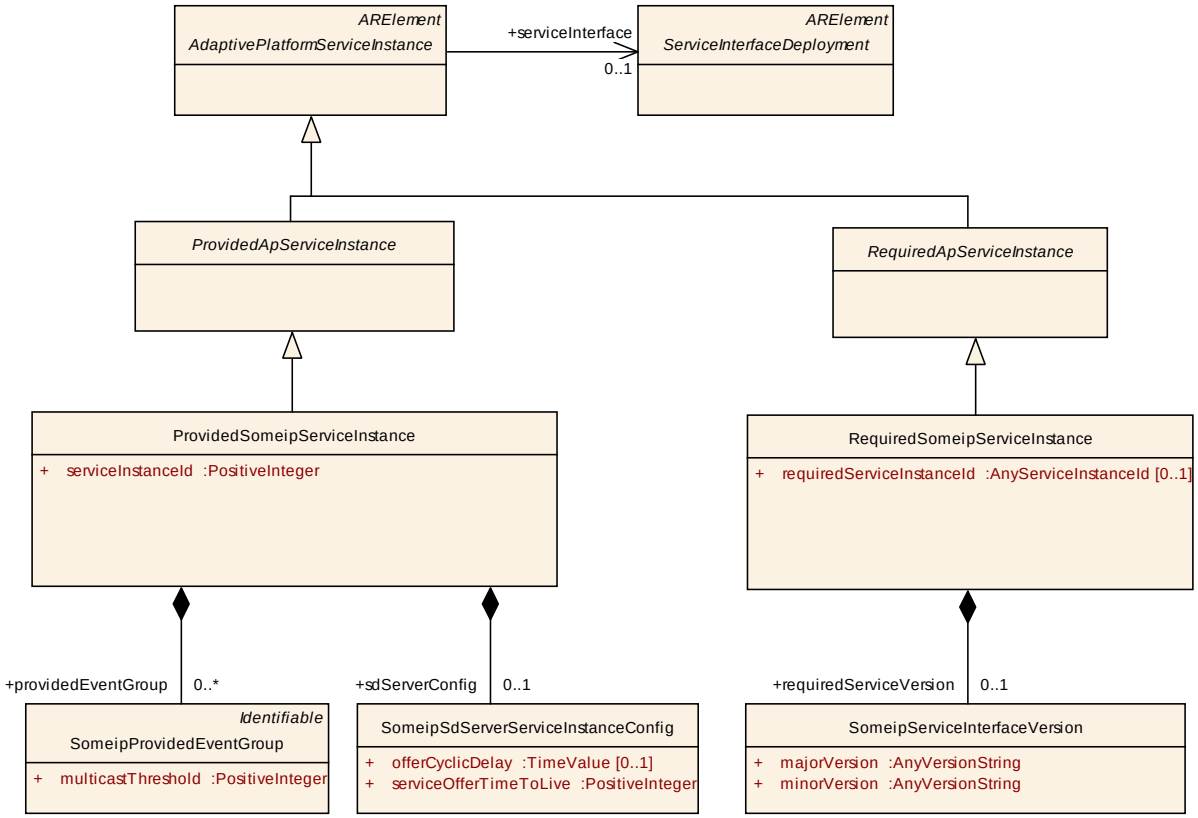

Figure 6.8: SOME/IP Service Instances

6.2.1.1 Provided Service Instance

The `ProvidedSomeipServiceInstance` defines the `serviceInstanceId` for the Service Instance of the `SomeipServiceInterface` that is referenced with the `serviceInterface` reference.

It means that the Server on which the `ProvidedSomeipServiceInstance` is deployed offers the Service Instance over SOME/IP with the `serviceInstanceId` and `serviceInterfaceId`.

Class	**ProvidedSomeipServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation on top of SOME/IP. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInstances			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, ProvidedApServiceInstance, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

providedEventGroup	SomeipProvidedEventGroup	*	aggr	List of EventGroups that are provided by the Service Instance. **Tags:** atp.Status=draft
sdServerConfig	SomeipSdServerServiceInstanceConfig	0..1	aggr	Server specific configuration settings relevant for the SOME/IP service discovery. **Tags:** atp.Status=draft
serviceInstanceId	PositiveInteger	1	attr	Identification number that is used by SOME/IP service discovery to identify the instance of the service.

Table 6.21: ProvidedSomeipServiceInstance

[constr_3287] Mandatory information of a `ProvidedSomeipServiceInstance` ⌈ The `ProvidedSomeipServiceInstance` shall always define the `serviceInstanceId`. ⌋*()*

In addition to the service identification properties a SOME/IP offer message contains so called endpoint options that define how the service instance is reachable by clients.

6.2.1.1.1 IP Configuration

In SOME/IP the Offer service entry references IPv4 or IPv6 Endpoint options to indicate to the client where the server accepts the method calls and where the server sends the event messages.

Such an Endpoint contains the IP address of the sender. The IP address configuration is described in this chapter.

[TPS_MANI_03002] IP configuration for a `ProvidedSomeipServiceInstance` ⌈ A `ProvidedSomeipServiceInstance` can be mapped to a `CommunicationConnector` of a `Machine` with the `SomeipServiceInstanceToMachineMapping`.

With this mapping an assignment of the `ProvidedSomeipServiceInstance` to a unicast IP Address is established since the `EthernetCommunicationConnector` refers to a `NetworkEndpoint` in the role `unicastNetworkEndpoint`. ⌋ *(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03003] `ProvidedSomeipServiceInstance` Fanout ⌈ It is allowed to map the same `ProvidedSomeipServiceInstance` to different `CommunicationConnector`s of a `Machine`. In such a case, several `SomeipServiceInstanceToMachineMapping`s shall be defined.

This allows for offering the same `ProvidedSomeipServiceInstance` on different VLANs or even on different `CommunicationCluster`s. ⌋*(RS_MANI_00009, RS_MANI_00024)*

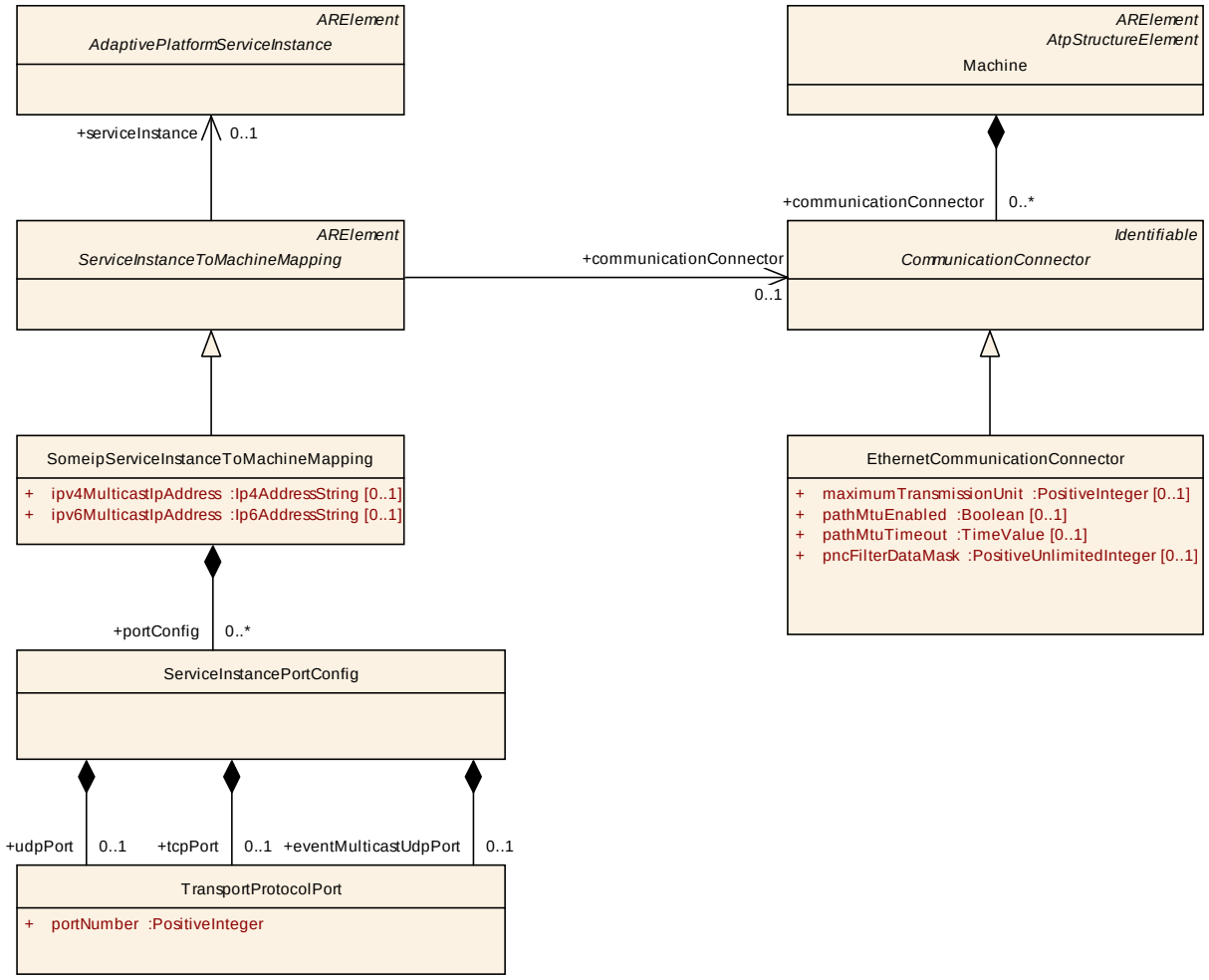

Figure 6.9: `SomeipServiceInstanceToMachineMapping` with TP and IP configuration

Class	≪`atpVariation`≫ **CommunicationCluster (abstract)**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	The CommunicationCluster is the main element to describe the topological connection of communicating ECUs. A cluster describes the ensemble of ECUs, which are linked by a communication medium of arbitrary topology (bus, star, ring, ...). The nodes within the cluster share the same communication protocol, which may be event-triggered, time-triggered or a combination of both. A CommunicationCluster aggregates one or more physical channels. **Tags:** vh.latestBindingTime=postBuild			
Base	ARObject, CollectableElement, FibexElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
baudrate	PositiveUnlimitedInteger	0..1	attr	Channels speed in bits/s.

physicalChannel	PhysicalChannel	1..*	aggr	This relationship defines which channel element belongs to which cluster. A channel must be assigned to exactly one cluster, whereas a cluster may have one or more channels. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
protocolName	String	0..1	attr	The name of the protocol used.
protocolVersion	String	0..1	attr	The version of the protocol used.

Table 6.22: CommunicationCluster

Class	**CommunicationConnector (abstract)**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	The connection between the referencing ECU and the referenced channel via the referenced controller. Connectors are used to describe the bus interfaces of the ECUs and to specify the sending/receiving behavior. Each CommunicationConnector has a reference to exactly one communicationController. Note: Several CommunicationConnectors can be assigned to one PhysicalChannel in the scope of one ECU Instance.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 6.23: CommunicationConnector

Class	**EthernetCommunicationConnector**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Ethernet specific attributes to the CommunicationConnector.			
Base	ARObject, CommunicationConnector, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
maximumTransmissionUnit	PositiveInteger	0..1	attr	This attribute specifies the maximum transmission unit in bytes.
networkEndpoint	NetworkEndpoint	*	ref	NetworkEndpoints
pathMtuEnabled	Boolean	0..1	attr	If enabled the IPv4/IPv6 processes incoming ICMP "Packet Too Big" messages and stores a MTU value for each destination address.
pathMtuTimeout	TimeValue	0..1	attr	If this value is >0 the IPv4/IPv6 will reset the MTU value stored for each destination after n seconds.

pncFilterDataMask	PositiveUnlimitedInteger	0..1	attr	Bit mask for Ethernet Payload used to configure the Ethernet Transceiver for partial network wakeup. This attribute should not be computed from the pncIdentifier values in order to support future introduction of additional PNCs. Note that for one EcuInstance all contributing pncFilterDataMask will be bitwise ORed to obtain the value of UdpNmPnFilterMaskByte. Note that this data mask is calculated over the whole payload (8 Byte) of the NmPdu ignoring the leading bytes which do not contain pncVector information. The number of leading bytes which shall be ignored is equivalent to the value of System.pncVectorOffset.
unicastNetworkEndpoint	NetworkEndpoint	0..1	ref	Network Endpoint that defines the IPAddress of the machine. **Tags:** atp.Status=draft

Table 6.24: EthernetCommunicationConnector

[constr_3288] IP configuration restriction for `unicastNetworkEndpoint`s ⌈ A `NetworkEndpoint` that is referenced by a `EthernetCommunicationConnector` in the role `unicastNetworkEndpoint` shall have either

- `Ipv4Configuration` or
- `Ipv6Configuration`

as `networkEndpointAddress` that is defined in the unicast IP range according to the rules defined in [TPS_MANI_03005] and [TPS_MANI_03006]. ⌋*()*

In SOME/IP, a server that offers a `ProvidedSomeipServiceInstance` is able to send events and notification events to an IP-Multicast address.

To indicate to the client to which Multicast IP address the event messages are send the Subscribe Eventgroup Acknowledgement entry contains a reference an IPv4 Multicast Option and/or and IPv6 Multicast Option.

[TPS_MANI_03004] IPv4 Multicast event destination address ⌈ Meta-class `SomeipServiceInstanceToMachineMapping` defines the multicast IPv4 address to which the `event`s and notification events are send with the attribute `ipv4MulticastIpAddress`. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03061] IPv6 Multicast event destination address ⌈ Meta-class `SomeipServiceInstanceToMachineMapping` defines the multicast IPv6 address to which the `event`s and notification events are sent with the attribute `ipv6MulticastIpAddress`. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03005] IPv4 Multicast address range ⌈ The IPv4 addresses reserved for multicast communication are in the range 224.0.0.0 through 239.255.255.255. Addresses between 0.0.0.0 and 223.255.255.255 are reserved for unicast communication. ⌋*()*

[TPS_MANI_03006] IPv6 Multicast address range ⌈ IPv6 multicast addresses are distinguished from unicast addresses by the value of the high-order octet of the addresses: a value of 0xFF (binary 11111111) identifies an address as an address reserved for multicast communication; any other value identifies an address as a unicast address. ⌋*()*

Class	**NetworkEndpoint**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	The network endpoint defines the network addressing (e.g. IP-Address or MAC multicast address).			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
fullyQualifiedDomainName	String	0..1	attr	Defines the fully qualified domain name (FQDN) e.g. some.example.host.
infrastructureServices	InfrastructureServices	0..1	aggr	Defines the network infrastructure services provided or consumed.
networkEndpointAddress	NetworkEndpointAddress	1..*	aggr	Definition of a Network Address. **Tags:** xml.namePlural=NETWORK-ENDPOINT-ADDRESSES
priority	PositiveInteger	0..1	attr	Priority of this Network-Endpoint.

Table 6.25: NetworkEndpoint

Class	**NetworkEndpointAddress (abstract)**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	To build a valid network endpoint address there has to be either one MAC multicast group reference or an ipv4 configuration or an ipv6 configuration.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 6.26: NetworkEndpointAddress

Class	**Ipv4Configuration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Internet Protocol version 4 (IPv4) configuration.			
Base	ARObject, NetworkEndpointAddress			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

assignmentPriority	PositiveInteger	0..1	attr	Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority.
defaultGateway	Ip4AddressString	0..1	attr	IP address of the default gateway.
dnsServerAddress	Ip4AddressString	*	attr	IP addresses of preconfigured DNS servers. **Tags:** xml.namePlural=DNS-SERVER-ADDRESSES
ipAddressKeepBehavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.
ipv4Address	Ip4AddressString	0..1	attr	IPv4 Address. Notation: 255.255.255.255. The IP Address shall be declared in case the ipv4AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv4AddressSource	Ipv4AddressSourceEnum	0..1	attr	Defines how the node obtains its IP address.
networkMask	Ip4AddressString	0..1	attr	Network mask. Notation 255.255.255.255
ttl	PositiveInteger	0..1	attr	Lifespan of data (0..255). The purpose of the TimeToLive field is to avoid a situation in which an undeliverable datagram keeps circulating on a system.

Table 6.27: Ipv4Configuration

Class	**Ipv6Configuration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Internet Protocol version 6 (IPv6) configuration.			
Base	ARObject, NetworkEndpointAddress			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
assignmentPriority	PositiveInteger	0..1	attr	Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority.
defaultRouter	Ip6AddressString	0..1	attr	IP address of the default router.
dnsServerAddress	Ip6AddressString	*	attr	IP addresses of pre configured DNS servers. **Tags:** xml.namePlural=DNS-SERVER-ADDRESSES
enableAnycast	Boolean	0..1	attr	This attribute is used to enable anycast addressing (i.e. to one of multiple receivers).
hopCount	PositiveInteger	0..1	attr	The distance between two hosts. The hop count n means that n gateways separate the source host from the destination host (Range 0..255)

ipAddress KeepBehavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.
ipAddress PrefixLength	PositiveInteger	0..1	attr	IPv6 prefix length defines the part of the IPv6 address that is the network prefix.
ipv6Address	Ip6AddressString	0..1	attr	IPv6 Address. Notation: FFFF:...:FFFF. The IP Address shall be declared in case the ipv6AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv6AddressSource	Ipv6AddressSourceEnum	0..1	attr	Defines how the node obtains its IP address.

Table 6.28: Ipv6Configuration

6.2.1.1.2 TP Configuration

The IPv4 or IPv6 Endpoint option that is referenced in the SOME/IP Offer message contains besides the IP address the transport layer protocol (e.g. UDP or TCP), and the port number of the sender.

With the `SomeipServiceInstanceToMachineMapping` the Transport Layer configuration attributes are assigned to the `ProvidedSomeipServiceInstance`.

The configuration attributes for the `OfferService` entry are available in the `ServiceInstancePortConfig` that is aggregated by `SomeipServiceInstanceToMachineMapping` in the role `portConfig`.

The same element contains the Transport Layer configuration attributes for the IPv4/IPv6 Multicast Option that may be used in the SOME/IP `SubscribeEventGroupAck` message.

[TPS_MANI_03007] Udp Transport Protocol Configuration for `ProvidedSomeipServiceInstance` ⌈ The meta-class `SomeipServiceInstanceToMachineMapping.portConfig` defines with the `udpPort` the Transport Protocol `portNumber` for a UDP communication.

This setting is used in an IPv4 or IPv6 Endpoint Option that is referenced by an `OfferService` entry. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03008] Tcp Transport Protocol Configuration for `ProvidedSomeipServiceInstance` ⌈ The meta-class `SomeipServiceInstanceToMachineMapping.portConfig` defines with the `tcpPort` the Transport Protocol `portNumber` for a TCP communication.

This setting is used in an IPv4 or IPv6 Endpoint Option that is referenced by an `OfferService` entry. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03009] Tcp and Udp Transport Protocol Configuration for `ProvidedSomeipServiceInstance` ⌈ It is allowed to set `tcpPort` and `udpPort` in the same `SomeipServiceInstanceToMachineMapping`.

Such a setting shall be used to indicate that one UDP endpoint and one TCP endpoint are referenced in the `OfferService` entry. It means that the Server provides the `ProvidedSomeipServiceInstance` over both Transport Protocols. ⌋ *(RS_MANI_00009, RS_MANI_00024)*

If a Tcp and Udp Transport Protocol Configuration is defined for a `ProvidedSomeipServiceInstance` as described in [TPS_MANI_03009] then the SOME/IP `ServiceInterfaceDeployment` settings decide which content of the `ProvidedSomeipServiceInstance` is transported over `udp` and which content is transported over `tcp`.

This is described in [TPS_MANI_03050] and [TPS_MANI_03051].

[TPS_MANI_03010] Udp Transport Protocol Configuration in case of IP-Multicast ⌈ The `SomeipServiceInstanceToMachineMapping.portConfig` defines with the `eventMulticastUdpPort` the Transport Protocol Port Number for a UDP event communication in case IP-Multicast is used.

This setting is used in an IPv4 or IPv6 Multicast Option that is referenced by a `SubscribeEventGroupAck` Service entry. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[constr_3290] Usage of `ServiceInstancePortConfig` defined for a `ProvidedSomeipServiceInstance` ⌈ Each `ServiceInstancePortConfig` of a `SomeipServiceInstanceToMachineMapping` that is defined for a `ProvidedSomeipServiceInstance` shall define either

- a `udpPort` or
- a `tcpPort` or
- a `udpPort` and a `tcpPort`.

⌋*()*

Class	**ServiceInstancePortConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This element is used to configure the Transport Protocol (UDP/TCP) and TP Port for the ProvidedServiceInstance. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

eventMulticastUdpPort	TransportProtocolPort	0..1	aggr	UdpPort configuration that is used for Event communication in the IP-Multicast case. SOME/IP Service Discovery: Send in the SD-SubscribeEventGroupAck Message to client (answer to SD-SubscribeEventGroup). Event: This is the destination-port where the server sends the multicast event messages if the mulicastThreshold of the corresponding ProvidedEventGroupInSomeipServiceInstance is exceeded. **Tags:** atp.Status=draft
tcpPort	TransportProtocolPort	0..1	aggr	TcpPort configuration that is used for Method and Event communication in IP-Unicast case. SOME/IP Service Discovery: PortNumber that is sent in the SD-Offer Message to client (answer on SD-find) or clients (SD-offer). Method: This is the destination-port where the server accepts the method call messages (from the clients). This is the source-port where the server sends the method response messages (to the client). Event: This is the event source-port where the server sends the event messages to the subscribed clients in IP-Unicast case. **Tags:** atp.Status=draft
udpPort	TransportProtocolPort	0..1	aggr	UdpPort configuration that is used for Method and Event communication in IP-Unicast case. SOME/IP Service Discovery: PortNumber that is sent in the SD-Offer Message to client (answer on SD-find) or clients (SD-offer). Method: This is the destination-port where the server accepts the method call messages (from the clients). This is the source-port where the server sends the method response messages (to the client). Event: This is the event source-port where the server sends the event messages to the subscribed clients in IP-Unicast case. **Tags:** atp.Status=draft

Table 6.29: ServiceInstancePortConfig

Class	**TransportProtocolPort**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This meta-class represents the ability to describe a UDP/TCP Port. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
portNumber	PositiveInteger	1	attr	This attribute represents the ability to specify a UDP or TCP Port number.

Table 6.30: TransportProtocolPort

6.2.1.1.3 Service Discovery Server Configuration

The multicast messages of the SOME/IP Service Discovery come with the risk of overflowing `Machine`s with too many messages. Therefore, the Service Discovery can be configured with a suitable message sending behavior.

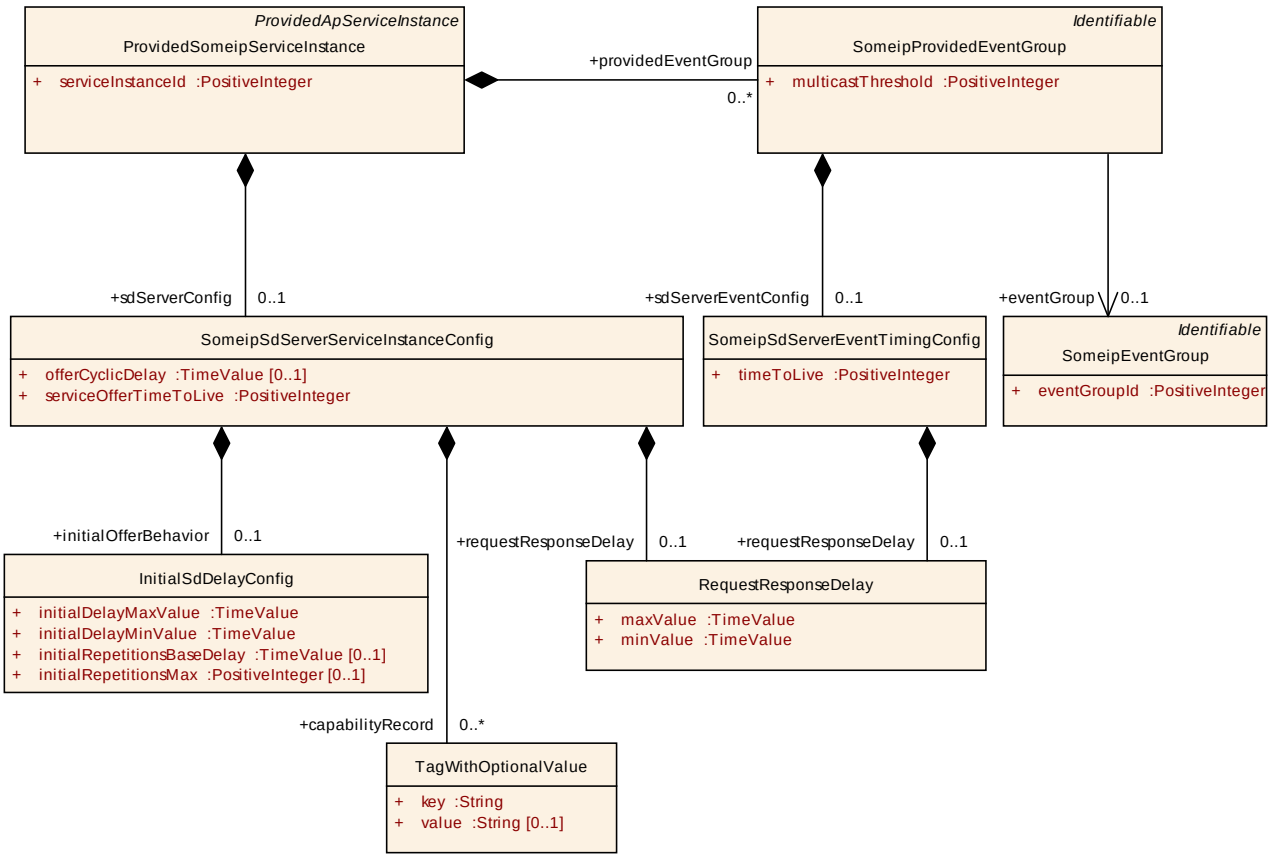

Figure 6.10: SOME/IP Service Discovery Server configuration settings

For every `ProvidedSomeipServiceInstance` on a Server different phases are existing:

- Down

- Available
  - Initial Wait Phase
  - Repetition Phase
  - Main Phase

[TPS_MANI_03011] Server Timing configuration for a `ProvidedSomeipServiceInstance` ⌈ The Server Timing is configurable with `SomeipSdServerServiceInstanceConfig` that is aggregated in the role `sdServerConfig` by the `ProvidedSomeipServiceInstance` for which the Timing is valid. ⌋*(RS_MANI_00024)*

Class	**SomeipSdServerServiceInstanceConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	Server specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
capabilityRecord	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service. **Tags:** atp.Status=draft
initialOfferBehavior	InitialSdDelayConfig	0..1	aggr	Controls offer behavior of the server. **Tags:** atp.Status=draft
offerCyclicDelay	TimeValue	0..1	attr	Optional attribute to define cyclic offers. Cyclic offer is active, if the delay is set (in seconds).
requestResponseDelay	RequestResponseDelay	0..1	aggr	Maximum/Minimum allowable response delay to entries received by multicast in seconds. The Service Discovery shall delay answers to entries that were transported in a multicast SOME/IP-SD message (e.g. FindService). **Tags:** atp.Status=draft
serviceOfferTimeToLive	PositiveInteger	1	attr	Defines the time in seconds the service offer is valid.

Table 6.31: SomeipSdServerServiceInstanceConfig

[TPS_MANI_03012] Initial Wait Phase configuration for a `ProvidedSomeipServiceInstance` ⌈ The Initial Wait Phase for a `ProvidedSomeipServiceInstance` is configured with the `initialOfferBehavior` and the two attributes `initialDelayMinValue` and `initialDelayMaxValue`.

When a calculated random timer based on these min and max values expires the first `OfferService` entry will be sent out. ⌋*(RS_MANI_00024)*

When the calculated random timer expires the Repetition Phase will be entered.

[TPS_MANI_03013] Repetition Wait Phase configuration for a ProvidedSomeipServiceInstance ⌈ The Repetition Wait Phase for a ProvidedSomeipServiceInstance is configured with the initialOfferBehavior and the two attributes initialRepetitionsMax and initialRepetitionsBaseDelay. ⌋*(RS_MANI_00024)*

If the Repetition Phase is entered the Service Discovery waits for the initialRepetitionsBaseDelay and then sends an OfferService entry. If the amount of sent OfferService entries reaches initialRepetitionsMax the Main Phase will be entered.

[TPS_MANI_03014] Main Phase configuration for a ProvidedSomeipServiceInstance ⌈ The Main Phase for a ProvidedSomeipServiceInstance is configured with the offerCyclicDelay attribute of SomeipSdServerServiceInstanceConfig.

The OfferService entry will be sent cyclically with an interval that is defined by the value of attribute offerCyclicDelay. ⌋*(RS_MANI_00024)*

Class	**InitialSdDelayConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This element is used to configure the offer behavior of the server and the find behavior on the client.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
initialDelayMaxValue	TimeValue	1	attr	Max Value in seconds to delay randomly the first offer (if aggregated by SdServerConfig) or the transmission of a find message (if aggregated by SdClientConfig).
initialDelayMinValue	TimeValue	1	attr	Min Value in seconds to delay randomly the first offer (if aggregated by SdServerConfig) or the transmission of a find message (if aggregated by SdClientConfig).
initialRepetitionsBaseDelay	TimeValue	0..1	attr	The base delay for offer repetitions (if aggregated by SdServerConfig) or find repetitions (if aggregated by SdClientConfig. Successive find messages have an exponential back off delay.
initialRepetitionsMax	PositiveInteger	0..1	attr	Describes the maximum amount of offer repetitions (if aggregated by SdServerConfig) or the maximum amount of find repetitions (if aggregated by SdClientConfig).

Table 6.32: InitialSdDelayConfig

[TPS_MANI_03015] TTL for Offer Service Entries ⌈ The lifetime of a ProvidedSomeipServiceInstance is configurable with the serviceOfferTimeToLive attribute of SomeipSdServerServiceInstanceConfig.

If the time that is configured by `serviceOfferTimeToLive` expires the `ProvidedSomeipServiceInstance` is no longer offered. ⌋*(RS_MANI_00024)*

[TPS_MANI_03016] Servers `RequestResponseDelay` for received `FindService` entries ⌈ The Server will delay the `OfferService` answer to a received multicast `FindService` entry by the configured `SomeipSdServerServiceInstanceConfig.requestResponseDelay`.

The actual delay will be randomly chosen between the `maxValue` and `minValue`. ⌋ *(RS_MANI_00024)*

Class	**RequestResponseDelay**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Time to wait before answering the query.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
maxValue	TimeValue	1	attr	Maximum allowable response delay to entries received by multicast in seconds.
minValue	TimeValue	1	attr	Minimum allowable response delay to entries received by multicast in seconds.

Table 6.33: RequestResponseDelay

Figure 6.11 shows an example of the different SOME/IP phases on the Server side.

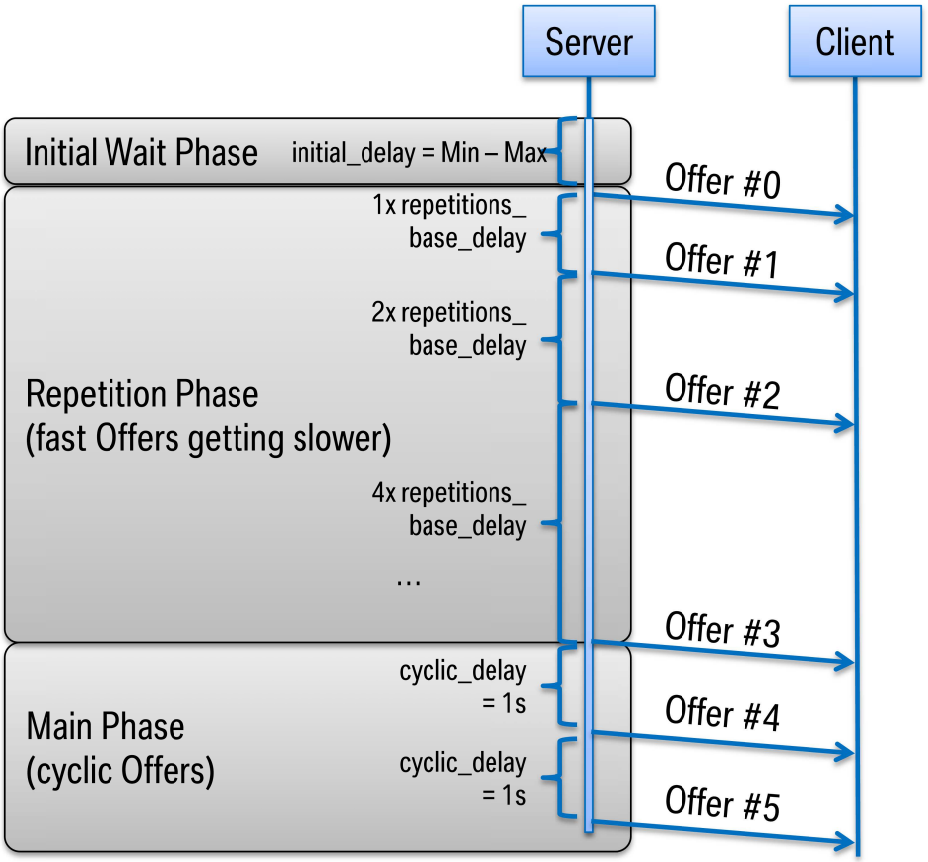

Figure 6.11: SOME/IP Server Timing example

SOME/IP allows for the specification of additional information about the `ProvidedSomeipServiceInstance` with the Capability Record that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.

[TPS_MANI_03017] Server Capability Records ⌈ A Capability Record (key/value pair) on the Server side is configurable with the `capabilityRecord` and the two attributes `key` and `value`. ⌋*(RS_MANI_00024)*

Class	**TagWithOptionalValue**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::TagWithOptionalValue			
Note	A tagged value is a combination of a tag (key) and a value that gives supplementary information that is attached to a model element. Please note that keys without a value are allowed.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
key	String	1	attr	Defines a key.
value	String	0..1	attr	Defines the corresponding value.

Table 6.34: TagWithOptionalValue

6.2.1.1.4 Provided Event Group

The ProvidedSomeipServiceInstance aggregates a SomeipProvidedEvent-Group in the role providedEventGroup that allows to define service instance specific configuration settings for a SomeipEventGroup.

Class	**SomeipProvidedEventGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the provided side for each EventGroup separately. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	0..1	ref	Reference to the SomeipEventGroup in the System Manifest for which the ServiceInstance related EventGroup settings are valid. **Tags:** atp.Status=draft
multicastThreshold	PositiveInteger	1	attr	Specifies the number of subscribed clients that trigger the server to change the transmission of events to multicast. Example: If configured to 0 only unicast will be used. If configured to 1 the first client will be already served by multicast. If configured to 2 the first client will be server with unicast and as soon as the 2nd client arrives both will be served by multicast. This does not influence the handling of initial events, which are served using unicast only.
sdServerEventConfig	SomeipSdServerEventTimingConfig	0..1	aggr	Server Timing configuration settings that are EventGroup specific. **Tags:** atp.Status=draft

Table 6.35: SomeipProvidedEventGroup

[TPS_MANI_03018] Usage of SomeipProvidedEventGroup.multicastThreshold ⌈ The switching between IP-Unicast and IP-Multicast is guided by the server with the SomeipProvidedEventGroup.multicastThreshold attribute and by the number of subscribed clients to the SomeipProvidedEventGroup.

The Server will change the transmission of events to Multicast if the multicastThreshold of the corresponding SomeipProvidedEventGroup is exceeded by the number of subscribed clients. If the number of subscribe clients is smaller or equal to this multicastThreshold, the transmission of events takes place via unicast communication. ⌋*(RS_MANI_00024)*

[TPS_MANI_03019] TTL for `SubscribeEventGroupAck` Entries ⌈ The lifetime of a event subscription is configurable with the `timeToLive` attribute of `SomeipSdServerEventTimingConfig`.

If the time that is configured by `timeToLive` expires the event subscription is canceled. ⌋*(RS_MANI_00024)*

[TPS_MANI_03020] Servers `RequestResponseDelay` for received `SubscribeEventGroup` entries ⌈ The Server will delay the `SubscribeEventGroupAck` answer to a received `SubscribeEventGroup` message that was triggered by a multicast `ServiceOffer` by the configured `SomeipSdClientEventGroupTimingConfig.requestResponseDelay`.

The actual delay will be randomly chosen between the `maxValue` and `minValue`. ⌋ *(RS_MANI_00024)*

Class	**SomeipSdServerEventTimingConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	EventGroup specific timing configuration settings. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
requestResponseDelay	RequestResponseDelay	0..1	aggr	The Service Discovery shall delay answers to unicast messages triggered by multicast messages (e.g. Subscribe Eventgroup after Offer Service). **Tags:** atp.Status=draft
timeToLive	PositiveInteger	1	attr	Defines the time in seconds the subscription of this eventGroup is valid. This value is send from the server to the client in the SD subscribeEventGroupAck message.

Table 6.36: SomeipSdServerEventTimingConfig

6.2.1.2 Required Service Instance

[TPS_MANI_03059] `RequiredSomeipServiceInstance.requiredServiceInstanceId` ⌈ The `RequiredSomeipServiceInstance` defines the `requiredServiceInstanceId` of a `SomeipServiceInterface` that the client searches.

The client may search for a specific `requiredServiceInstanceId` or for ANY `requiredServiceInstanceId` of the `serviceInterface`. ⌋*(RS_MANI_00024)*

Class	**RequiredSomeipServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation on top of SOME/IP. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInstances			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, RequiredApServiceInstance			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
requiredEventGroup	SomeipRequiredEventGroup	*	aggr	List of EventGroups that are used by the RequiredServiceInstance. **Tags:** atp.Status=draft
requiredServiceInstanceId	AnyServiceInstanceId	0..1	attr	This attribute represents the ability to describe the required service instance ID.
requiredServiceVersion	SomeipServiceInterfaceVersion	0..1	aggr	This element is used to configure for which version (major version/minor version) of the SomeIp Service the Service Discovery will search. **Tags:** atp.Status=draft
sdClientConfig	SomeipSdClientServiceInstanceConfig	0..1	aggr	Client specific configuration settings relevant for the SOME/IP service discovery. **Tags:** atp.Status=draft

Table 6.37: RequiredSomeipServiceInstance

[constr_3293] Mandatory information of a `RequiredSomeipServiceInstance` ⌈ The `RequiredSomeipServiceInstance` shall always define the attributes `requiredServiceInstanceId` and `requiredServiceVersion`. ⌋*()*

[TPS_MANI_03021] Requirements on the service version from the client's point of view ⌈ The meta-class `RequiredSomeipServiceInstance` can also make further specifications regarding the version of the service from the client's point of view.

For this purpose, the attribute `RequiredSomeipServiceInstance.requiredServiceVersion` exists and provides the ability to define the required major version (`SomeipServiceInterfaceVersion.majorVersion`) and the minor version (`SomeipServiceInterfaceVersion.minorVersion`). ⌋*(RS_MANI_00024)*

Class	**SomeipServiceInterfaceVersion**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This meta-class represents the ability to describe a version of a SOME/IP ServiceInterface. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

majorVersion	AnyVersionString	1	attr	Major Version of the ServiceInterface. Value can be set to a number that represents the Major Version of the searched service or to ANY.
minorVersion	AnyVersionString	1	attr	Minor Version of the ServiceInterface. Value can be set to a number that represents the Minor Version of the searched service or to ANY.

Table 6.38: SomeipServiceInterfaceVersion

6.2.1.2.1 IP Configuration

In SOME/IP, the SubscribeEventGroup entry references IPv4 or IPv6 Endpoint options to indicate to the server where the client wants to receive the events of the SomeipEventGroup. Such an Endpoint contains the IP address of the client.

[TPS_MANI_03022] Context of RequiredSomeipServiceInstance ⌈ A RequiredSomeipServiceInstance can be mapped to a CommunicationConnector of a Machine with the SomeipServiceInstanceToMachineMapping.

With this mapping an assignment of the RequiredSomeipServiceInstance to a unicast IP Address is established since the EthernetCommunicationConnector refers to a NetworkEndpoint in the role unicastNetworkEndpoint. The unicastNetworkEndpoint defines the local IP address of the client. ⌋ *(RS_MANI_00009, RS_MANI_00024)*

6.2.1.2.2 TP Configuration

The IPv4 or IPv6 Endpoint option that is referenced in the SOME/IP SubscribeEventGroup message contains besides the IP address the transport layer protocol (e.g. UDP or TCP), and the port number of the client.

With the SomeipServiceInstanceToMachineMapping the Transport Layer configuration attributes are assigned to the RequiredSomeipServiceInstance.

The Transport Layer (TCP/UDP) configuration attributes for the SubscribeEventGroup entry are available in the ServiceInstancePortConfig that is aggregated by SomeipServiceInstanceToMachineMapping in the role portConfig.

The same ServiceInstancePortConfig defines the source-port where the client sends the method call messages to the server and the destination-port where the client receives the method responses from the server.

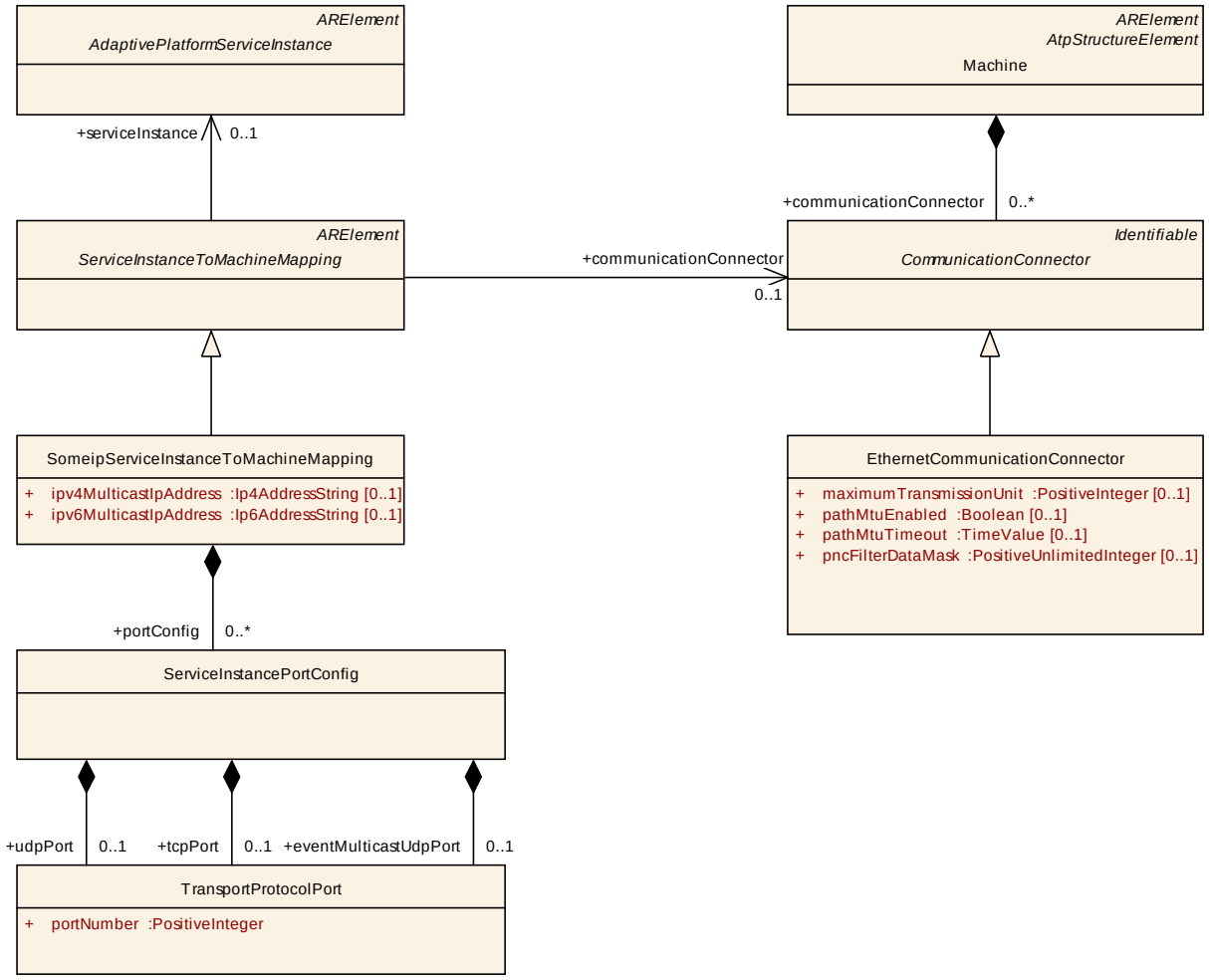

Figure 6.12: `SomeipServiceInstanceToMachineMapping` with TP and IP configuration

[TPS_MANI_03023] Udp Transport Protocol Configuration for `RequiredSomeipServiceInstance` ⌈ The `SomeipServiceInstanceToMachineMapping.portConfig` defines with the `udpPort` the Transport Protocol `portNumber` for a UDP communication in case that the server provides `ServiceInterface` content over UDP and the client wants to use it. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03024] Tcp Transport Protocol Configuration for `RequiredSomeipServiceInstance` ⌈ The `SomeipServiceInstanceToMachineMapping.portConfig` defines with the `tcpPort` the Transport Protocol `portNumber` for a TCP communication in case that the server provides `ServiceInterface` content over TCP and the client wants to use it. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03049] Tcp and Udp Transport Protocol Configuration for `RequiredSomeipServiceInstance` ⌈ It is allowed to set `tcpPort` and `udpPort` in the same `SomeipServiceInstanceToMachineMapping`. Such a setting shall be used in case that the server provides `ServiceInterface` content over Udp and Tcp and the client wants to use it. ⌋*(RS_MANI_00009, RS_MANI_00024)*

[constr_3296] Usage of `ServiceInstancePortConfig` defined for a `RequiredSomeipServiceInstance` ⌈ Each `ServiceInstancePortConfig` of a

SomeipServiceInstanceToMachineMapping that is defined for a RequiredSomeipServiceInstance shall define either

- a udpPort or
- a tcpPort or
- a udpPort and a tcpPort.

⌋()

If a Tcp and Udp Transport Protocol Configuration is defined for a RequiredSomeipServiceInstance as described in [TPS_MANI_03049] then the SOME/IP ServiceInterfaceDeployment settings decide which content of the ProvidedSomeipServiceInstance is transported over udp and which content is transported over tcp. This is described in [TPS_MANI_03050] and [TPS_MANI_03051].

6.2.1.2.3 Service Discovery Client Configuration

Service Discovery phases on the Client side allow minimizing the number of Service Discovery messages and allow a fast synchronization upon ECU start.

For every RequiredSomeipServiceInstance on a Client different phases are existing:

- Down
- Requested
  - Initial Wait Phase
  - Repetition Phase
  - Main Phase

[TPS_MANI_03025] Client Timing configuration for a RequiredSomeipServiceInstance ⌈ The Client Timing is configurable with SomeipSdClientServiceInstanceConfig that is aggregated in the role sdClientConfig by the RequiredSomeipServiceInstance for which the Timing is valid. ⌋*(RS_MANI_00024)*

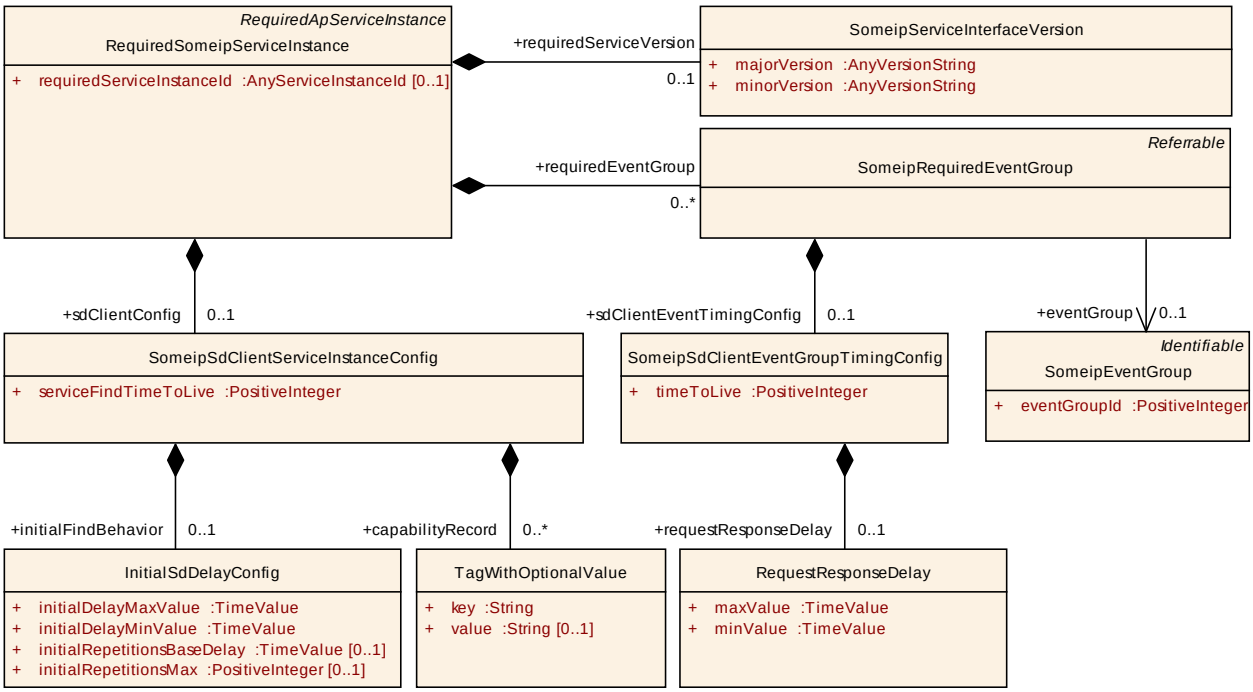

Figure 6.13: SOME/IP Service Discovery Client configuration settings

Class	**SomeipSdClientServiceInstanceConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	Client specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
capabilityRecord	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service.
initialFindBehavior	InitialSdDelayConfig	0..1	aggr	Controls initial find behavior of clients.
serviceFindTimeToLive	PositiveInteger	1	attr	This attribute represents the ability to define the time in seconds the service find is valid.

Table 6.39: SomeipSdClientServiceInstanceConfig

[TPS_MANI_03026] Initial Wait Phase configuration for a `RequiredSomeipServiceInstance` ⌈ The Initial Wait Phase for a `RequiredSomeipServiceInstance` is configured with the `initialFindBehavior` and the two attributes `initialDelayMinValue` and `initialDelayMaxValue`.

If a calculated random timer based on these min and max values expires the first `FindService` entry will be sent out. ⌋*(RS_MANI_00024)*

When the calculated random timer expires and no `OfferService` is received the Repetition Phase will be entered.

[TPS_MANI_03027] Repetition Wait Phase configuration for a `RequiredSomeipServiceInstance` ⌈ The Repetition Wait Phase for a `RequiredSomeipServiceInstance` is configured with the `initialFindBehavior` and the two attributes `initialRepetitionsMax` and `initialRepetitionsBaseDelay`. ⌋*(RS_MANI_00024)*

If the Repetition Phase is entered, the Service Discovery waits the `initialRepetitionsBaseDelay` and sends an `FindService` entry.

If the amount of sent `FindService` entries reaches `initialRepetitionsMax` and no `OfferService` is received the Main Phase will be entered. In the Main Phase no further `FindService` entries are send by the client.

[TPS_MANI_03028] TTL for Find Service Entries ⌈ The lifetime of a `RequiredSomeipServiceInstance` is configurable with the `serviceFindTimeToLive` attribute of `SomeipSdClientServiceInstanceConfig`.

If the time that is configured by `serviceFindTimeToLive` expires the `FindService` entry shall be considered not existing. ⌋*(RS_MANI_00024)*

Figure 6.14 shows an example of the different SOME/IP phases on the Client side.

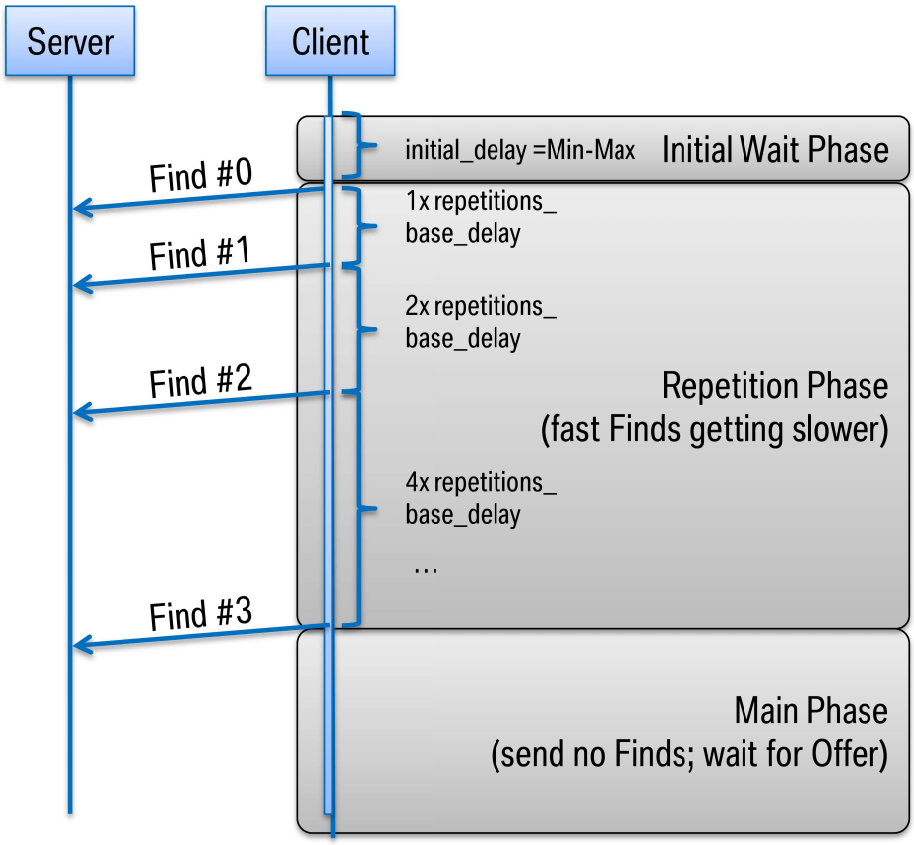

Figure 6.14: SOME/IP Client Timing example

SOME/IP allows to specify additional information about the RequiredSomeipServiceInstance with the Capability Record that allows to transport arbitrary configuration strings (key/value pairs).

This allows to encode additional information like the name of a service or its configuration.

[TPS_MANI_03029] Client Capability Records ⌈ A Capability Record (key/value pair) on the Client side is configurable with the capabilityRecord and the two attributes key and value. ⌋*(RS_MANI_00024)*

6.2.1.2.4 Required Event Group

The RequiredSomeipServiceInstance aggregates a SomeipRequiredEventGroup in the role requiredEventGroup that allows to define service instance specific configuration settings for a SomeipEventGroup.

Class	**SomeipRequiredEventGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the required side for each EventGroup separately. **Tags:** atp.Status=draft			
Base	ARObject, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	0..1	ref	Reference to the SomeipEventGroup in the System Manifest for which the ServiceInstance related EventGroup settings are valid. **Tags:** atp.Status=draft
sdClientEventTimingConfig	SomeipSdClientEventGroupTimingConfig	0..1	aggr	Client Timing configuration settings that are EventGroup specific. **Tags:** atp.Status=draft

Table 6.40: SomeipRequiredEventGroup

Class	**SomeipSdClientEventGroupTimingConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This meta-class is used to specify configuration related to service discovery in the context of an event group on SOME/IP. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
requestResponseDelay	RequestResponseDelay	0..1	aggr	The Service Discovery shall delay answers to unicast messages triggered by multicast messages (e.g. Subscribe Eventgroup after Offer Service).

timeToLive	PositiveInteger	1	attr	Defines the time in seconds the subscription of this event is expected by the client. this value is send from the client to the server in the SD-subscribeEvent message.

Table 6.41: SomeipSdClientEventGroupTimingConfig

[TPS_MANI_03030] **`SomeipSdClientEventGroupTimingConfig.timeToLive` for `SubscribeEventGroup` Entries** ⌈ The lifetime of a event subscription is configurable with the `timeToLive` attribute of `SomeipSdClientEventGroupTimingConfig`.

If the time that is configured by `timeToLive` expires the event subscription is canceled. ⌋*(RS_MANI_00024)*

[TPS_MANI_03031] Clients `RequestResponseDelay` for received `ServiceOffer` entries ⌈ The Client will delay the `SubscribeEventGroup` answer to a received `ServiceOffer` message by the configured `SomeipSdClientEventGroupTimingConfig.requestResponseDelay`.

The actual delay will be randomly chosen between the `maxValue` and `minValue`. ⌋ *(RS_MANI_00024)*

6.2.2 User Defined Service Instance Deployment

[TPS_MANI_03032] Description of middleware technologies not standardized by AUTOSAR ⌈ The elements `ProvidedUserDefinedServiceInstance` and `RequiredUserDefinedServiceInstance` can be used to describe alternative middleware technologies that are not standardized by AUTOSAR. ⌋*(RS_MANI_00014)*

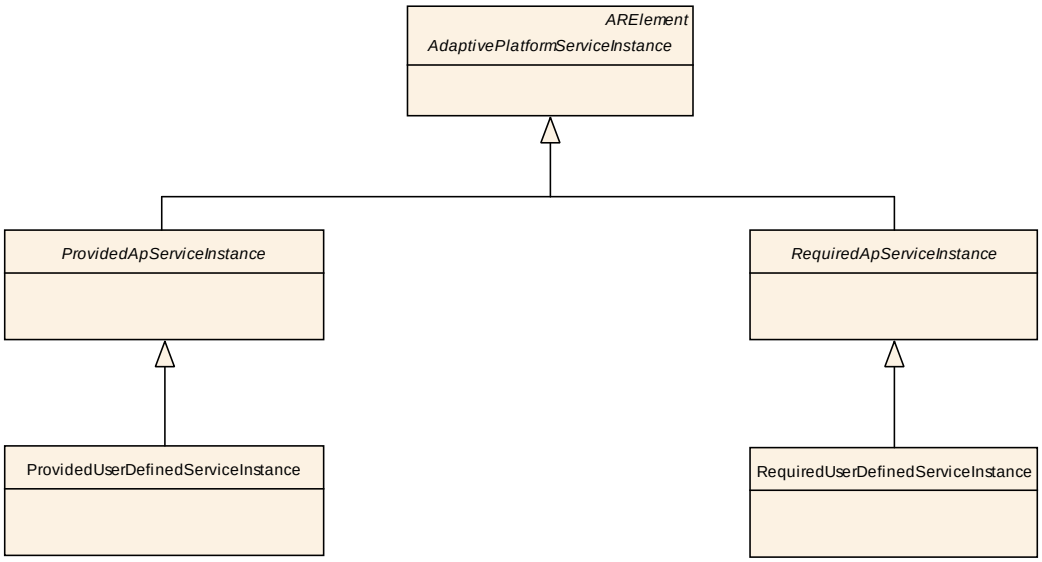

Figure 6.15: User Defined Service Instance Deployment

Class	**ProvidedUserDefinedServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation that is not standardized by AUTOSAR. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInstances			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, ProvidedApService Instance, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 6.42: ProvidedUserDefinedServiceInstance

Class	**RequiredUserDefinedServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation that is not standardized by AUTOSAR. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInstances			
Base	ARElement, ARObject, AdaptivePlatformServiceInstance, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, RequiredAp ServiceInstance			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 6.43: RequiredUserDefinedServiceInstance

Please note that both elements `ProvidedUserDefinedServiceInstance` and `RequiredUserDefinedServiceInstance` are `Identifiable` and therefore are able to describe special data (`sdg`) which is not represented by the standard model.

7 Machine Manifest

The `Machine` meta-class defines a computing resource on which the *Adaptive AUTOSAR Software Stack* is executed.

Therefore, the `Machine` meta-class allows to describe the computing capabilities, the available network connections, and storage resources that are available on the `Machine` in addition to the configuration settings of the *Adaptive AUTOSAR Software Stack* that is running on this `Machine`.

An overview of the `Machine` meta-class is sketched in Figure 7.1.

[TPS_MANI_03035] Content of the Machine configuration ⌈ The purpose of the `Machine` is to provide machine specific configuration settings. ⌋*(RS_MANI_00018, RS_MANI_00020, RS_MANI_00021, RS_MANI_00022, RS_MANI_00023)*

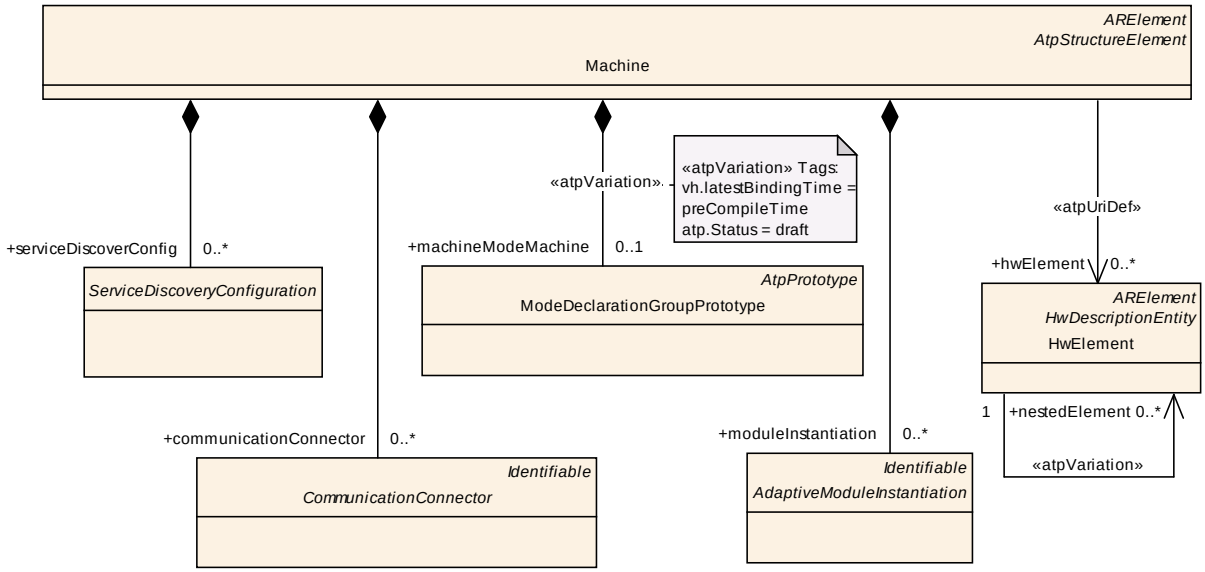

Figure 7.1: Overview about the content of the Machine configuration

Class	**Machine**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Machine			
Note	Machine that represents an Adaptive Autosar Software Stack. **Tags:** atp.Status=draft; atp.recommendedPackage=Machines			
Base	ARElement, ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Collectable Element, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
communicationConnector	CommunicationConnector	*	aggr	This aggregation defines the network connection of the machine. **Tags:** atp.Status=draft

hwElement	HwElement	*	ref	This reference is used to describe the hardware resources of the machine. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
machineModeMachine	ModeDeclarationGroupPrototype	0..1	aggr	Set of MachineStates (Modes) that are defined for the machine. **Stereotypes:** atpVariation **Tags:** atp.Status=draft vh.latestBindingTime=preCompileTime
moduleInstantiation	AdaptiveModuleInstantiation	*	aggr	Configuration of Adaptive Autosar module instances that are running on the machine. **Tags:** atp.Status=draft
serviceDiscoverConfig	ServiceDiscoveryConfiguration	*	aggr	Set of service discovery configuration settings that are defined on the machine for individual CommunicationConnectors. **Tags:** atp.Status=draft

Table 7.1: Machine

7.1 Network connection

One of the most prominent information defined in the context of the `Machine` is the network connectivity. Since the *AUTOSAR adaptive platform* focuses on the usage of Ethernet for communication, this boils down to the specification of IP addresses.

Specifically, the basic definition of the connectivity of a `Machine` is created by aggregating the abstract base-class `CommunicationConnector` in the role `communicationConnector`. The specific subclass of `CommunicationConnector` that is used in this context is the `EthernetCommunicationConnector`.

The `EthernetCommunicationConnector` is used to connect the `Machine` with a `VLAN` that is represented in AUTOSAR by a `EthernetPhysicalChannel` that is part of an `EthernetCluster`.

Class	**PhysicalChannel (abstract)**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	A physical channel is the transmission medium that is used to send and receive information between communicating ECUs. Each CommunicationCluster has at least one physical channel. Bus systems like CAN and LIN only have exactly one PhysicalChannel. A FlexRay cluster may have more than one PhysicalChannels that may be used in parallel for redundant communication. An ECU is part of a cluster if it contains at least one controller that is connected to at least one channel of the cluster.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
commConnector	CommunicationConnector	1..*	ref	Reference to the ECUInstance via a CommunicationConnector to which the channel is connected. atpVariation: Variable assignment of Physical Channels to different CommunicationConnectors is expressed with this variation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
frameTriggering	FrameTriggering	*	aggr	One frame triggering is defined for exactly one channel. Channels may have assigned an arbitrary number of frame triggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings must be variable, too. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
iSignalTriggering	ISignalTriggering	*	aggr	One ISignalTriggering is defined for exactly one channel. Channels may have assigned an arbitrary number of ISignaltriggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings must be variable, too. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild

pduTriggering	PduTriggering	*	aggr	One PduTriggering is defined for exactly one channel. Channels may have assigned an arbitrary number of I-Pdu triggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings must be variable, too. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild

Table 7.2: PhysicalChannel

Class	≪`atpVariation`≫ **EthernetCluster**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Ethernet-specific cluster attributes. **Tags:** atp.recommendedPackage=CommunicationClusters			
Base	ARObject, CollectableElement, CommunicationCluster, FibexElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
couplingPortConnection	CouplingPortConnection	*	aggr	Specification of connections between CouplingElements and EcuInstances. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=couplingPortConnection, variationPoint.shortLabel vh.latestBindingTime=postBuild
couplingPortSwitchoffDelay	TimeValue	0..1	attr	Switch off delay for CouplingPorts in seconds. It denotes the delay of switching off couplingPorts after the request to switch off a couplingPort was issued. (e.g. switch off of Ethernet switch ports).
macMulticastGroup	MacMulticastGroup	*	aggr	MacMulticastGroup that is defined for the Subnet (EthernetCluster).

Table 7.3: EthernetCluster

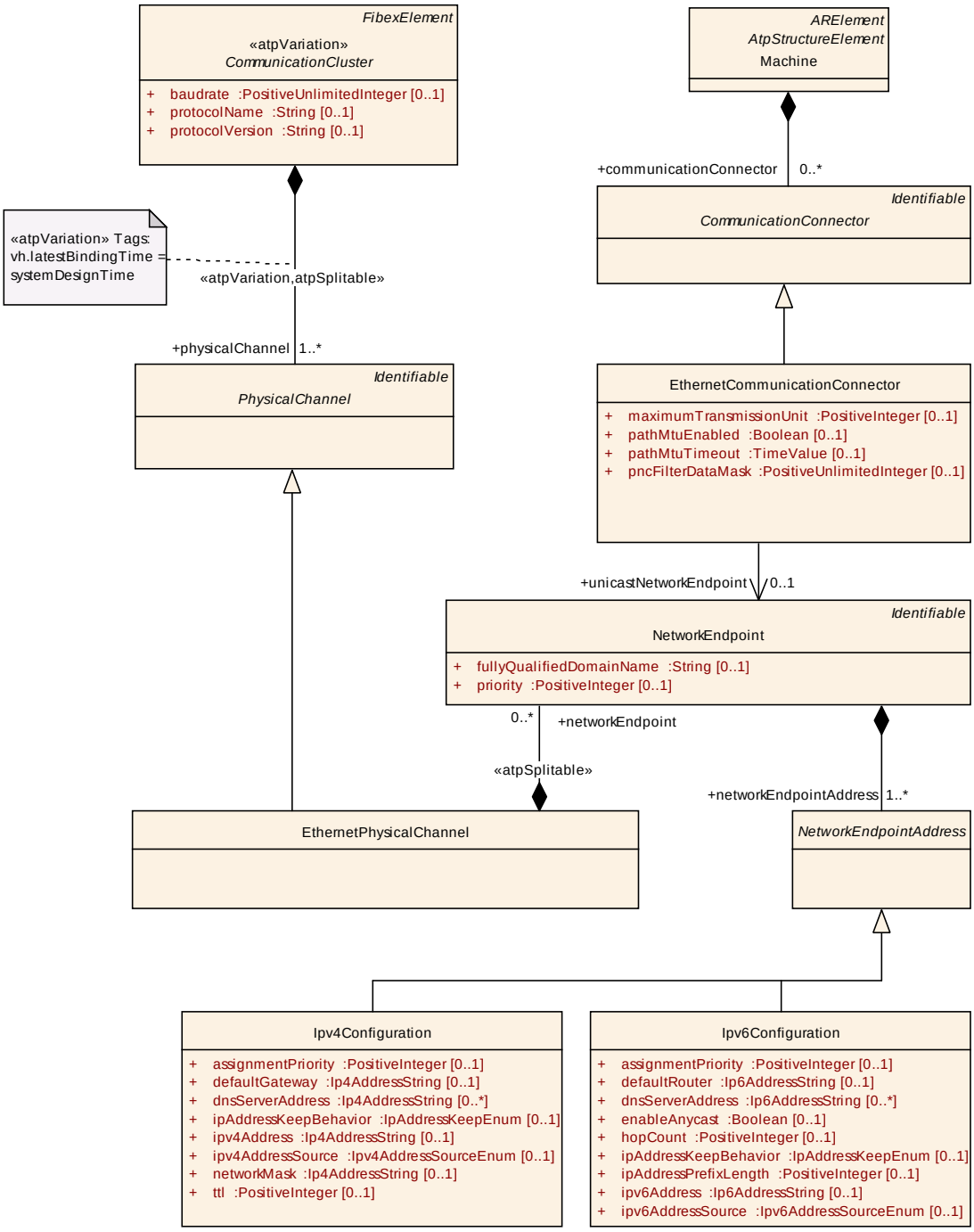

Figure 7.2: Network connection of a Machine

[constr_3320] Aggregation of `CommunicationConnector` by `Machine` ⌈ MetaClass `Machine` shall only aggregate `EthernetCommunicationConnector`s in the role `communicationConnector`. No other subclass of `CommunicationConnector` shall appear in this aggregation. ⌋*()*

The canonical way to specify an IP address is the modeling of a `NetworkEndpoint`, referenced from an `EthernetCommunicationConnector` that is aggregated by `Machine` in the role `communicationConnector`.

In addition to the IP address, the NetworkEndpoint may have a *Fully Qualified Domain Name* and a priority.

Class	**NetworkEndpoint**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	The network endpoint defines the network addressing (e.g. IP-Address or MAC multicast address).			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
fullyQualifiedDomainName	String	0..1	attr	Defines the fully qualified domain name (FQDN) e.g. some.example.host.
infrastructureServices	InfrastructureServices	0..1	aggr	Defines the network infrastructure services provided or consumed.
networkEndpointAddress	NetworkEndpointAddress	1..*	aggr	Definition of a Network Address. **Tags:** xml.namePlural=NETWORK-ENDPOINT-ADDRESSES
priority	PositiveInteger	0..1	attr	Priority of this Network-Endpoint.

Table 7.4: NetworkEndpoint

More precisely, the particular IP address is configured by means of the aggregation of Ipv4Configuration resp. Ipv6Configuration in the role networkEndpointAddress.

The NetworkEndpoint is aggregated by the EthernetPhysicalChannel that in turn is aggregated by the EthernetCluster.

Please note that the reference commConnector from the EthernetPhysicalChannel to the CommunicationConnector is optional although the lower multiplicity in the model is 1. The multiplicity of 1 is related to AUTOSAR Classic Platform and will be changed in future.

[TPS_MANI_03052] Static IPv4 configuration ⌈ If the value of attribute ipv4AddressSource of meta-class Ipv4Configuration is set to Ipv4AddressSourceEnum.fixed then the ipv4Address defines the static IPv4 Address. ⌋*(RS_MANI_00018)*

Class	**Ipv4Configuration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Internet Protocol version 4 (IPv4) configuration.			
Base	ARObject, NetworkEndpointAddress			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
assignmentPriority	PositiveInteger	0..1	attr	Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority.

defaultGateway	Ip4AddressString	0..1	attr	IP address of the default gateway.
dnsServerAddress	Ip4AddressString	*	attr	IP addresses of preconfigured DNS servers. **Tags:** xml.namePlural=DNS-SERVER-ADDRESSES
ipAddressKeepBehavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.
ipv4Address	Ip4AddressString	0..1	attr	IPv4 Address. Notation: 255.255.255.255. The IP Address shall be declared in case the ipv4AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv4AddressSource	Ipv4AddressSourceEnum	0..1	attr	Defines how the node obtains its IP address.
networkMask	Ip4AddressString	0..1	attr	Network mask. Notation 255.255.255.255
ttl	PositiveInteger	0..1	attr	Lifespan of data (0..255). The purpose of the TimeToLive field is to avoid a situation in which an undeliverable datagram keeps circulating on a system.

Table 7.5: Ipv4Configuration

Enumeration	**Ipv4AddressSourceEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Defines how the node obtains its IPv4-Address.
Literal	***Description***
autoIp	AutoIP is used to dynamically assign IP addresses at device startup. **Tags:** atp.EnumerationValue=0
autoIp_doip	Linklocal IPv4 Address Assignment using DoIP Parameters **Tags:** atp.EnumerationValue=2
dhcpv4	DHCP is a service for the automatic IP configuration of a client. **Tags:** atp.EnumerationValue=3
fixed	The IP Address shall be declared manually. **Tags:** atp.EnumerationValue=4

Table 7.6: Ipv4AddressSourceEnum

[TPS_MANI_03053] Static IPv6 configuration ⌈ If the value of attribute `ipv6AddressSource` of meta-class `Ipv6Configuration` is set to `Ipv6AddressSourceEnum.fixed` then the `ipv6Address` defines the static IPv6 Address. ⌋*(RS_MANI_00018)*

Class	**Ipv6Configuration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Internet Protocol version 6 (IPv6) configuration.			
Base	ARObject, NetworkEndpointAddress			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
assignmentPriority	PositiveInteger	0..1	attr	Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority.
defaultRouter	Ip6AddressString	0..1	attr	IP address of the default router.
dnsServerAddress	Ip6AddressString	*	attr	IP addresses of pre configured DNS servers. **Tags:** xml.namePlural=DNS-SERVER-ADDRESSES
enableAnycast	Boolean	0..1	attr	This attribute is used to enable anycast addressing (i.e. to one of multiple receivers).
hopCount	PositiveInteger	0..1	attr	The distance between two hosts. The hop count n means that n gateways separate the source host from the destination host (Range 0..255)
ipAddressKeepBehavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.
ipAddressPrefixLength	PositiveInteger	0..1	attr	IPv6 prefix length defines the part of the IPv6 address that is the network prefix.
ipv6Address	Ip6AddressString	0..1	attr	IPv6 Address. Notation: FFFF:...:FFFF. The IP Address shall be declared in case the ipv6AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv6AddressSource	Ipv6AddressSourceEnum	0..1	attr	Defines how the node obtains its IP address.

Table 7.7: Ipv6Configuration

Enumeration	**Ipv6AddressSourceEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Defines how the node obtains its IPv6-Address.
Literal	***Description***
dhcpv6	DHCP is a service for the automatic IP configuration of a client. **Tags:** atp.EnumerationValue=0
fixed	The IP Address shall be declared manually. **Tags:** atp.EnumerationValue=1

linkLocal	LinkLocal is intended only for communications within the segment of a local network (a link) or a point-to-point connection that a host is connected to. **Tags:** atp.EnumerationValue=2
linkLocal _doip	Linklocal IPv6 Address Assignment using DoIP Parameters **Tags:** atp.EnumerationValue=3
routerAdver-tisement	IPv6 Stateless Autoconfiguration. **Tags:** atp.EnumerationValue=4

Table 7.8: Ipv6AddressSourceEnum

Enumeration	**IpAddressKeepEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::Ethernet Topology
Note	Defines the behavior after a dynamic IP address has been assigned.
Literal	***Description***
forget	After a dynamic IP address has been assigned just use it for this session. **Tags:** atp.EnumerationValue=0
storePersis-tently	After a dynamic IP address has been assigned store the address persistently. **Tags:** atp.EnumerationValue=1

Table 7.9: IpAddressKeepEnum

7.2 Service Discovery Configuration

Service Discovery messages are exchanged between network nodes to announce and to discover available service instances. This chapter describes the configuration that is necessary to exchange service discovery messages for supported middleware transport layers.

Class	**ServiceDiscoveryConfiguration (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Machine			
Note	Service Discovery configuration settings for the middleware transport layer. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 7.10: ServiceDiscoveryConfiguration

7.2.1 SOME/IP Service Discovery Configuration

[TPS_MANI_03064] SOME/IP Service Discovery message exchange configuration ⌈ `ProvidedServiceInstance`s are announced in SOME/IP by the server with multicast addressing on a VLAN to a specifically designated IP multicast address (`SomeipServiceDiscovery.multicastSdIpAddress`) at a specific UDP port number (`SomeipServiceDiscovery.someipServiceDiscoveryPort`). ⌋ *(RS_MANI_00019)*

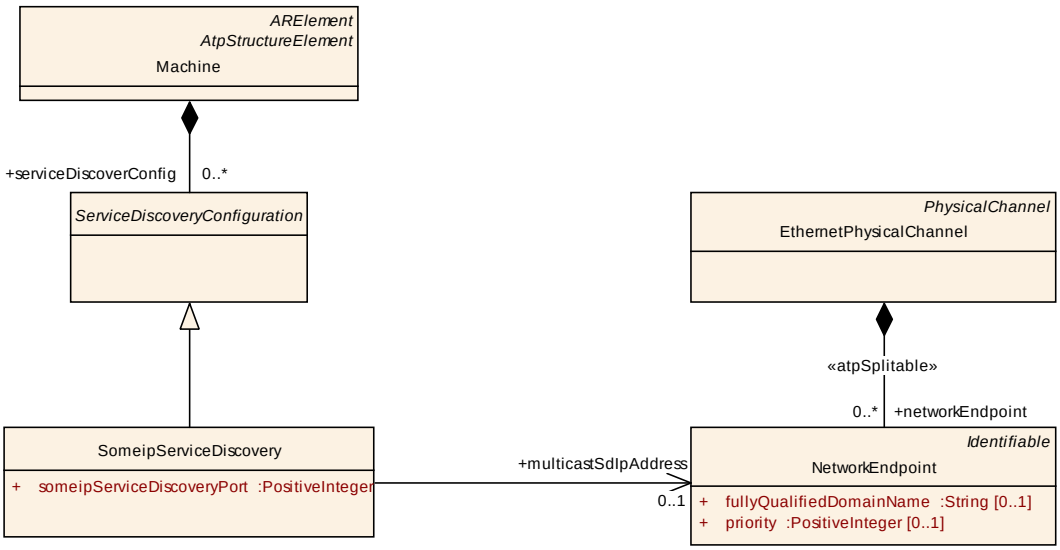

Figure 7.3: SOME/IP Service Discovery Configuration

Class	**SomeipServiceDiscovery**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInterfaceDeployment			
Note	This meta-class represents a specialization of the generic service discovery for the SOME/IP case. **Tags:** atp.Status=draft			
Base	ARObject, ServiceDiscoveryConfiguration			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
multicastSdIpAddress	NetworkEndpoint	0..1	ref	This reference identifies the multicast IP address used for service discovery. **Tags:** atp.Status=draft
someipServiceDiscoveryPort	PositiveInteger	1	attr	This attribute represents the port number reserved for service discovery.

Table 7.11: SomeipServiceDiscovery

7.3 Hardware Resources

[TPS_MANI_03065] Hardware resources of the machine ⌈ With the `Machine.hwElement` reference it is possible to formally describe the hardware of the machine. ⌋*(RS_MANI_00020)*

The `HwElement` is the main describing element that is used for example to describe Processing units, memory, peripherals and sensors/actuators.

The `HwCategory` that is referenced by the `HwElement` defines the hardware type and the applicable attribute definitions are defined by `HwAttributeDef`. An attribute value can be assigned to `HwAttributeDef` by `hwAttributeValue`.

Predefined categories and corresponding attributes are described in the Ecu Resource Template [10].

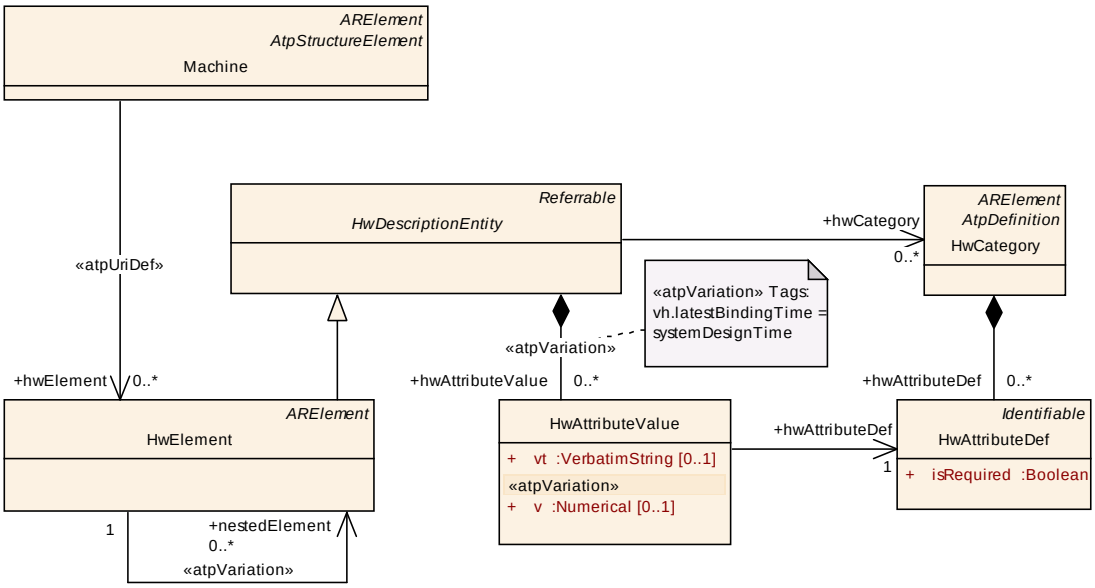

Figure 7.4: Description of hardware resources of the machine

Class	**HwElement**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
Note	This represents the ability to describe Hardware Elements on an instance level. The particular types of hardware are distinguished by the category. This category determines the applicable attributes. The possible categories and attributes are defined in HwCategory. **Tags:** atp.recommendedPackage=HwElements			
Base	ARElement, ARObject, CollectableElement, HwDescriptionEntity, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

hwElement Connection	HwElementConnector	*	aggr	This represents one particular connection between two hardware elements. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=110
hwPinGroup	HwPinGroup	*	aggr	This aggregation is used to describe the connection facilities of a hardware element. Note that hardware element has no pins but only pingroups. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=90
nestedElement	HwElement	*	ref	This association is used to establish hierarchies of hw elements. Note that one particular HwElement can be target of this association only once. I.e. multiple instantiation of the same HwElement is not supported (at any hierarchy level). **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=70

Table 7.12: HwElement

Class	**HwDescriptionEntity (abstract)**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
Note	This meta-class represents the ability to describe a hardware entity.			
Base	ARObject, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hwAttributeValue	HwAttributeValue	*	aggr	This aggregation represents a particular hardware attribute value. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime xml.sequenceOffset=50
hwCategory	HwCategory	*	ref	One of the associations representing one particular category of the hardware entity. **Tags:** xml.sequenceOffset=30
hwType	HwType	0..1	ref	This association is used to assign an optional HwType which contains the common attribute values for all occurences of this HwDescriptionEntity. Note that HwTypes can not be redefined and therefore shall not have a hwType reference.

Table 7.13: HwDescriptionEntity

Class	**HwAttributeValue**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
Note	This metaclass represents the ability to assign a hardware attribute value. Note that v and vt are mutually exclusive.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
annotation	Annotation	0..1	aggr	Optional annotation that can be added to each HwAttributeValue.
hwAttribute Def	HwAttributeDef	1	ref	This association represents the definition of the particular hardware attribute value.
v	Numerical	0..1	attr	This represents a numerical hardware attribute value. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=systemDesignTime
vt	VerbatimString	0..1	attr	This represents a textual hardware attribute value.

Table 7.14: HwAttributeValue

Class	**HwCategory**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
Note	This metaclass represents the ability to declare hardware categories and its particular attributes. **Tags:** atp.recommendedPackage=HwCategorys			
Base	ARElement, ARObject, AtpDefinition, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hwAttribute Def	HwAttributeDef	*	aggr	This aggregation describes particular hardware attribute definition.

Table 7.15: HwCategory

Class	**HwAttributeDef**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
Note	This metaclass represents the ability to define a particular hardware attribute. The category of this element defines the type of the attributeValue. If the category is Enumeration the hwAttributeEnumerationLiterals specify the available literals.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hwAttribute Literal	HwAttributeLiteralDef	*	aggr	The available EnumerationLiterals of the Enumeration definition. Only applicable if the category of the HwAttributeDef equals Enumeration.
isRequired	Boolean	1	attr	This attribute specifies if the defined attribute value is required to be provided.
unit	Unit	0..1	ref	This association specifies the physical unit of the defined hardware attribute. This is optional due to the fact that there are textual attributes.

Table 7.16: HwAttributeDef

7.4 Machine States

[TPS_MANI_03066] Description of machine states ⌈ With the machineModeMachine aggregation it is possible to define a set of Modes (States) as ModeDeclarationGroupPrototype in the context of a Machine.

The ModeDeclarationGroupPrototype points to a reusable ModeDeclarationGroup in the role type that contains the different modes as ModeDeclarations and a designated initialMode. ⌋*(RS_MANI_00021)*

Please note that the startup of a Process may depend on Modes that are defined in the context of a Machine. The ModeDependentStartupConfig is described in chapter 5.2.

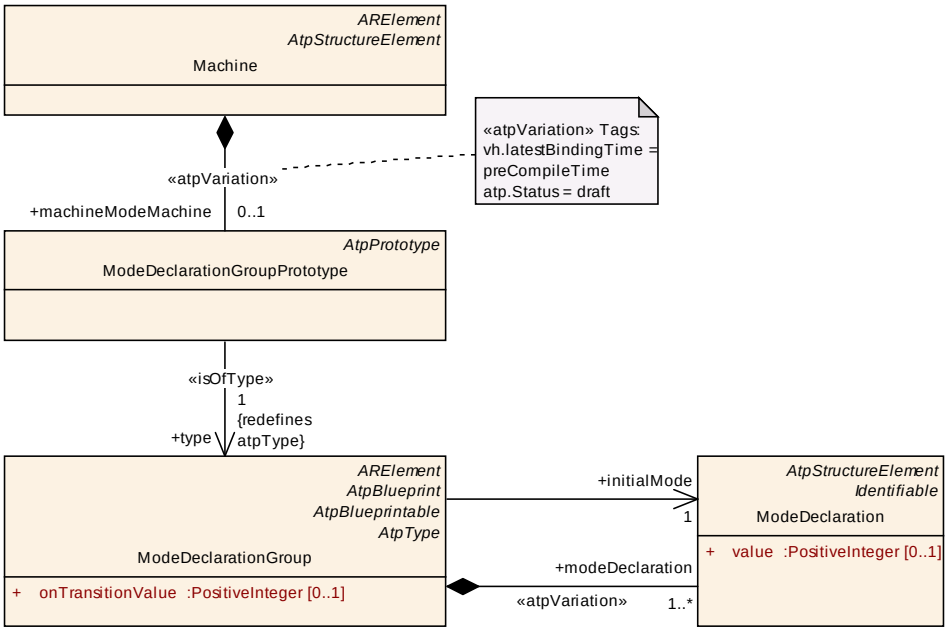

Figure 7.5: Configuration of Machine States

Class	**ModeDeclarationGroupPrototype**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context.			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

type	ModeDeclarationGroup	1	tref	The "collection of ModeDeclarations" (= ModeDeclarationGroup) supported by a component **Stereotypes:** isOfType

Table 7.17: ModeDeclarationGroupPrototype

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

Class	**ModeDeclaration**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this ModeDeclaration.

Table 7.19: ModeDeclaration

7.5 Adaptive Autosar Module and Platform Configuration

The configuration settings for individual Adaptive Autosar modules are covered by specializations of the abstract class `AdaptiveModuleInstantiation`.

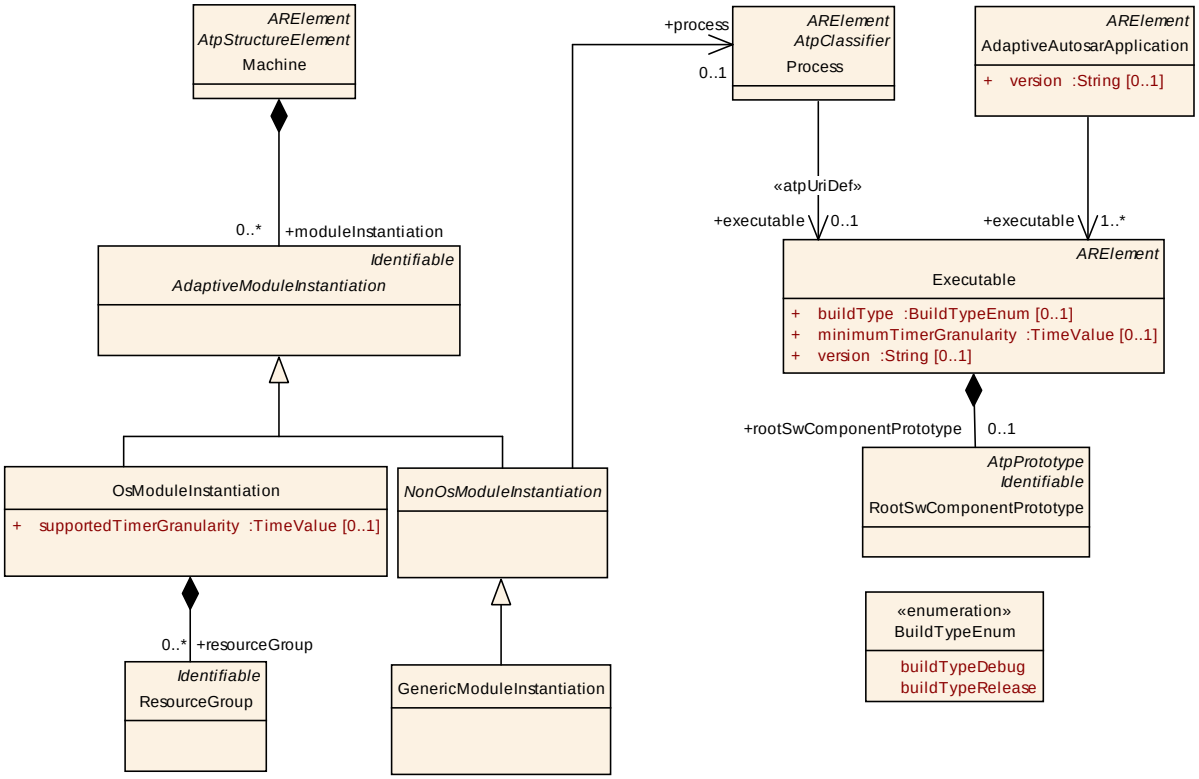

Figure 7.6: Adaptive Autosar Module Configuration

Class	**AdaptiveModuleInstantiation (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::AdaptiveModuleImplementation			
Note	This meta-class defines the abstract attributes for the configuration of an adaptive autosar module instance on a specific machine. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 7.20: AdaptiveModuleInstantiation

Each Adaptive Autosar module other than OS can be assigned to a `Process` with the `NonOsModuleInstantiation.process` reference.

[constr_1490] Allowed value of `category` for reference `NonOsModuleInstantiation.process.executable` ⌈ The value of `category` of an `Executable` referenced in the role `NonOsModuleInstantiation.process.executable` shall **only** be set to `PLATFORM_LEVEL` (see [TPS_MANI_01009]). ⌋*()*

The meta-class `GenericModuleInstantiation` can be used to define configuration settings of generic modules and modules that are not standardized by AUTOSAR. Different modules are distinguishable by the `category` attribute.

Please note that both elements are `Identifiable` and therefore are able to describe special data (`sdg`), by which means it is possible to define generic custom settings that are not represented by the standard model. For more information, please refer to the AUTOSAR Generic Structure Template [5].

[TPS_MANI_03096] `Machine`-specific configuration settings for a generic module ⌈ The `Machine`-specific configuration settings for a generic module are collected in `GenericModuleInstantiation` where the value of attribute `category` value denotes the module. ⌋*(RS_MANI_00023)*

Class	**GenericModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::AdaptiveModuleImplementation			
Note	This meta-class defines the attributes for the generic module configuration on a specific machine. Different modules are distinguishable by the category attribute. This element can also be used to describe modules that are not standardized by AUTOSAR. **Tags:** atp.Status=draft			
Base	ARObject, AdaptiveModuleInstantiation, Identifiable, MultilanguageReferrable, NonOsModuleInstantiation, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 7.21: GenericModuleInstantiation

7.5.1 OS Module configuration

[TPS_MANI_03098] `Machine`-specific configuration settings for the OS module ⌈ The `Machine`-specific configuration settings for the OS module are collected in `OsModuleInstantiation`. ⌋*(RS_MANI_00023)*

Class	**OsModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::AdaptiveModuleImplementation			
Note	This meta-class defines the attributes for the OS configuration on a specific machine. **Tags:** atp.Status=draft			
Base	ARObject, AdaptiveModuleInstantiation, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
resourceGroup	ResourceGroup	*	aggr	This represents the collection of ResourceGroups owned by the enclosing OsModuleImplementation. **Tags:** atp.Status=draft
supportedTimerGranularity	TimeValue	0..1	attr	This attribute describes the supported timer granularity (TimeValue of one tick).

Table 7.22: OsModuleInstantiation

Class	**ResourceGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::AdaptiveModuleImplementation			
Note	This meta-class represents a resource group. **Tags:** atp.Status=draft			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 7.23: ResourceGroup

A Examples

This chapter contains a collection of examples that reflect concepts described in different chapters of this document. The content of the chapter provides mere explanation and does not add anything to the model semantics.

A.1 Service Instance Deployment by Service Interface Mapping

The example in Figure A.2 sketches the modeling of a `ProvidedSomeipServiceInstance` in the presence of a `ServiceInterfaceMapping`, that references two `ServiceInterface`s in the role `sourceServiceInterface`.

For support, Figure A.1 contains an excerpt from the meta-model that contains the relevant meta-classes that have been instantiated to create the example sketched in Figure A.2.

Note further that the example depicted in Figure A.2 is not limited to the explanation of the actual `ServiceInterfaceMapping`.

As the main use case for this is the usage of `ServiceInterface`s for the definition of an "outside" communication binding the example also contains the modeling of such a binding, in this case to SOME/IP.

Please note that the modeling of the binding requires the existence of a `PortPrototype`, which in turn is aggregated by an `SwComponentType` (not depicted).

This approach still contains some degrees of freedom with respect to the role of the `SwComponentType` that aggregates the mentioned `PortPrototype`. This document does not go further in discussing the nature of such a configuration.

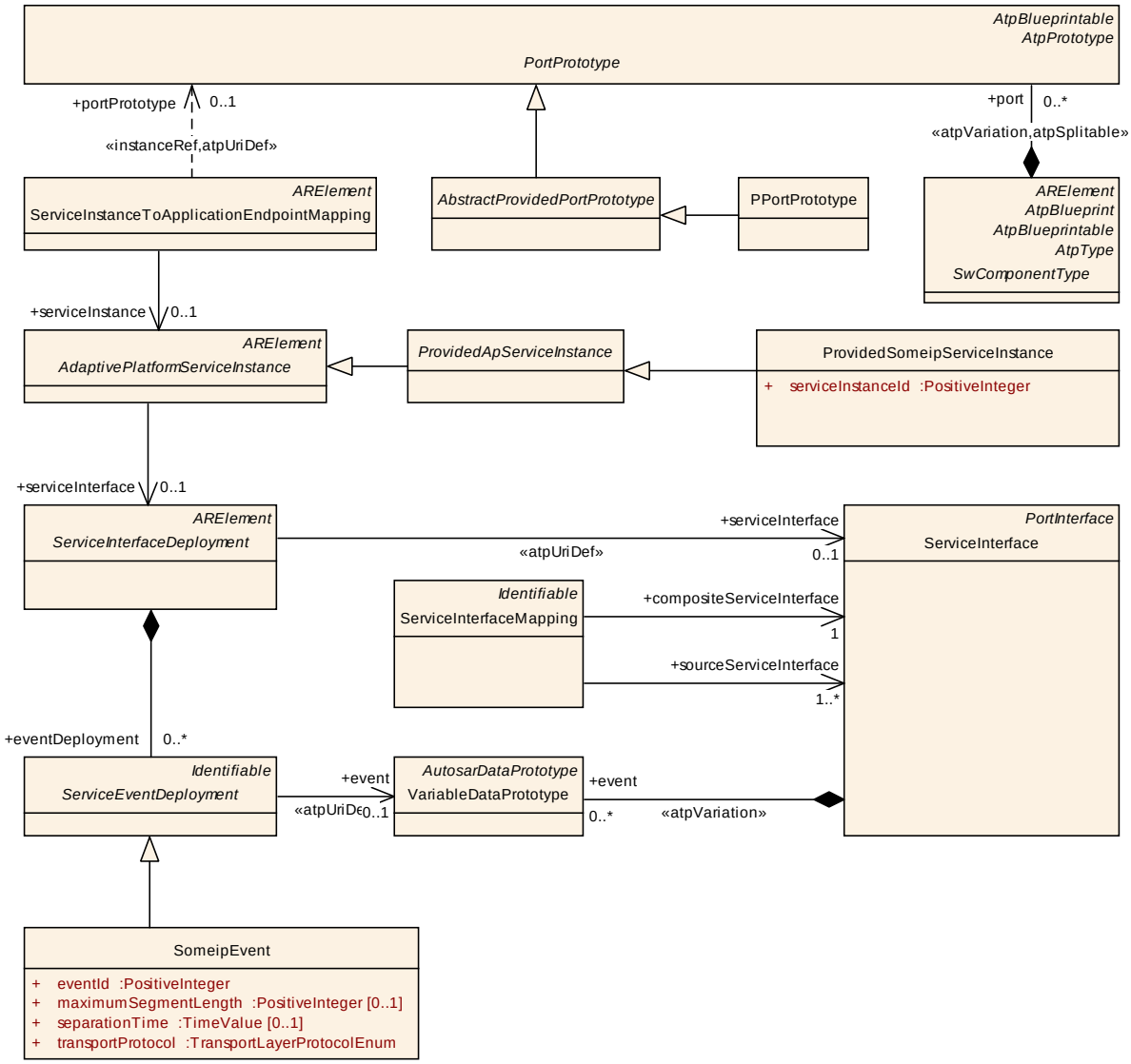

Figure A.1: Meta-model excerpt relevant for the example

For reasons of keeping the example as simple as possible, each of the `ServiceInterfaces` in the role `sourceServiceInterface` aggregate a single `event`.

The `ServiceInterface` referenced in the role `compositeServiceInterface` aggregates two `event` with `shortNames` that match the mentioned `event` of the source `ServiceInterfaces` (see [TPS_MANI_01022]).

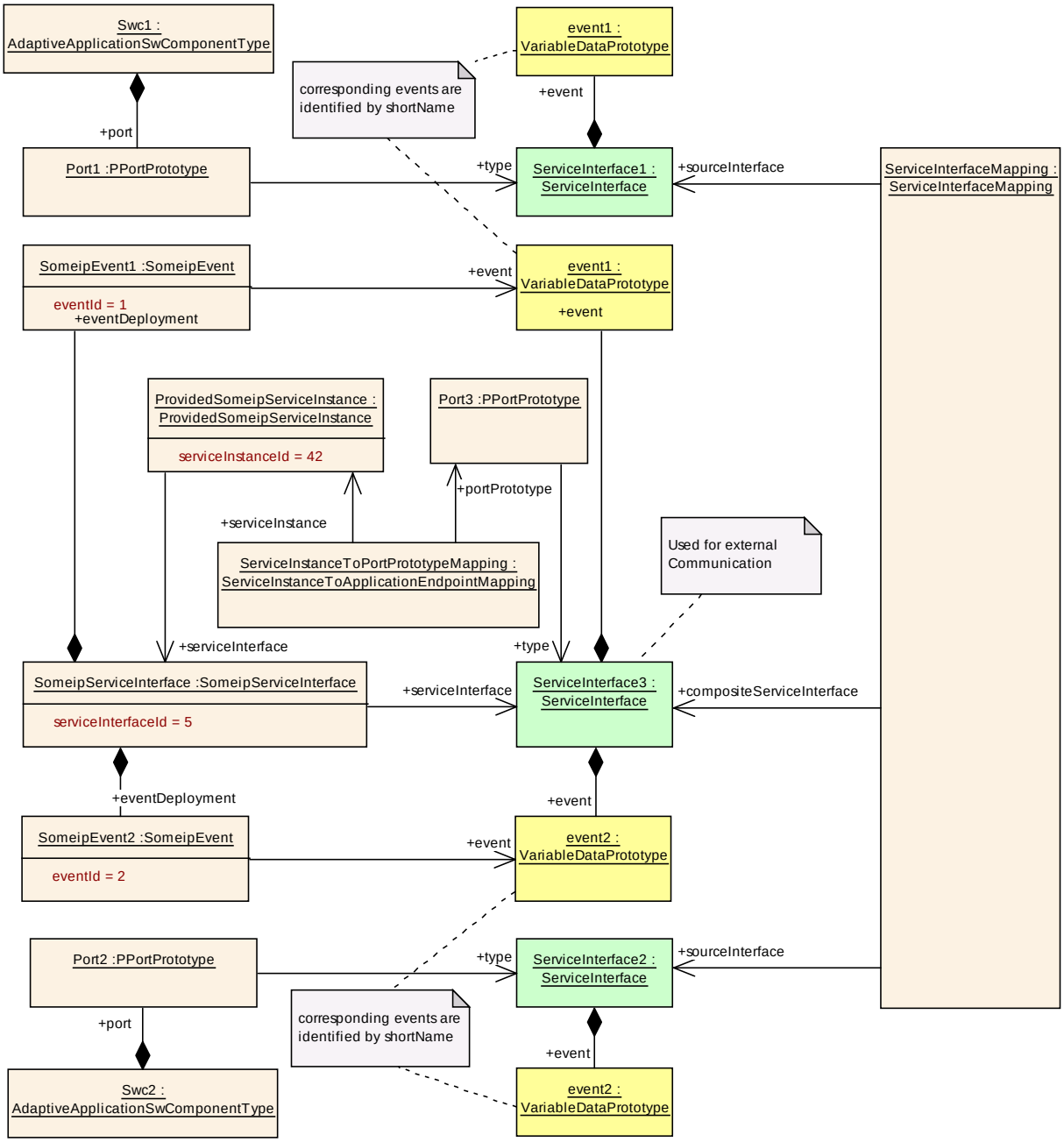

Figure A.2: Example for the deployments of a service in the presence of a `ServiceInterfaceMapping`

A.2 Service Instance Deployment by Service Interface Element Mapping

The example in Figure A.4 sketches the modeling of a `ProvidedSomeipServiceInstance` in the presence of a `ServiceInterfaceEventMapping`s. In principle, this example is very close to the example described in Figure A.2.

In contrast to the example sketched in Figure A.2, the example depicted in Figure A.4 uses a mapping to individual elements of a ServiceInterface instead of the entire ServiceInterface.

Please find the corresponding excerpt of relevant meta-classes for the utilization of ServiceInterfaceEventMapping sketched in Figure A.3.

Note further that the example depicted in Figure A.3 is not limited to the explanation of the actual ServiceInterfaceElementMapping.

As the main use case for this is the usage of ServiceInterfaces for the definition of an "outside" communication binding the example also contains the modeling of such a binding, in this case to SOME/IP.

Please note that the modeling of the binding requires the existence of a PortPrototype, which in turn is aggregated by an SwComponentType (not depicted).

This approach still contains some degrees of freedom with respect to the role of the SwComponentType that aggregates the mentioned PortPrototype. This document does not go further in discussing the nature of such a configuration.

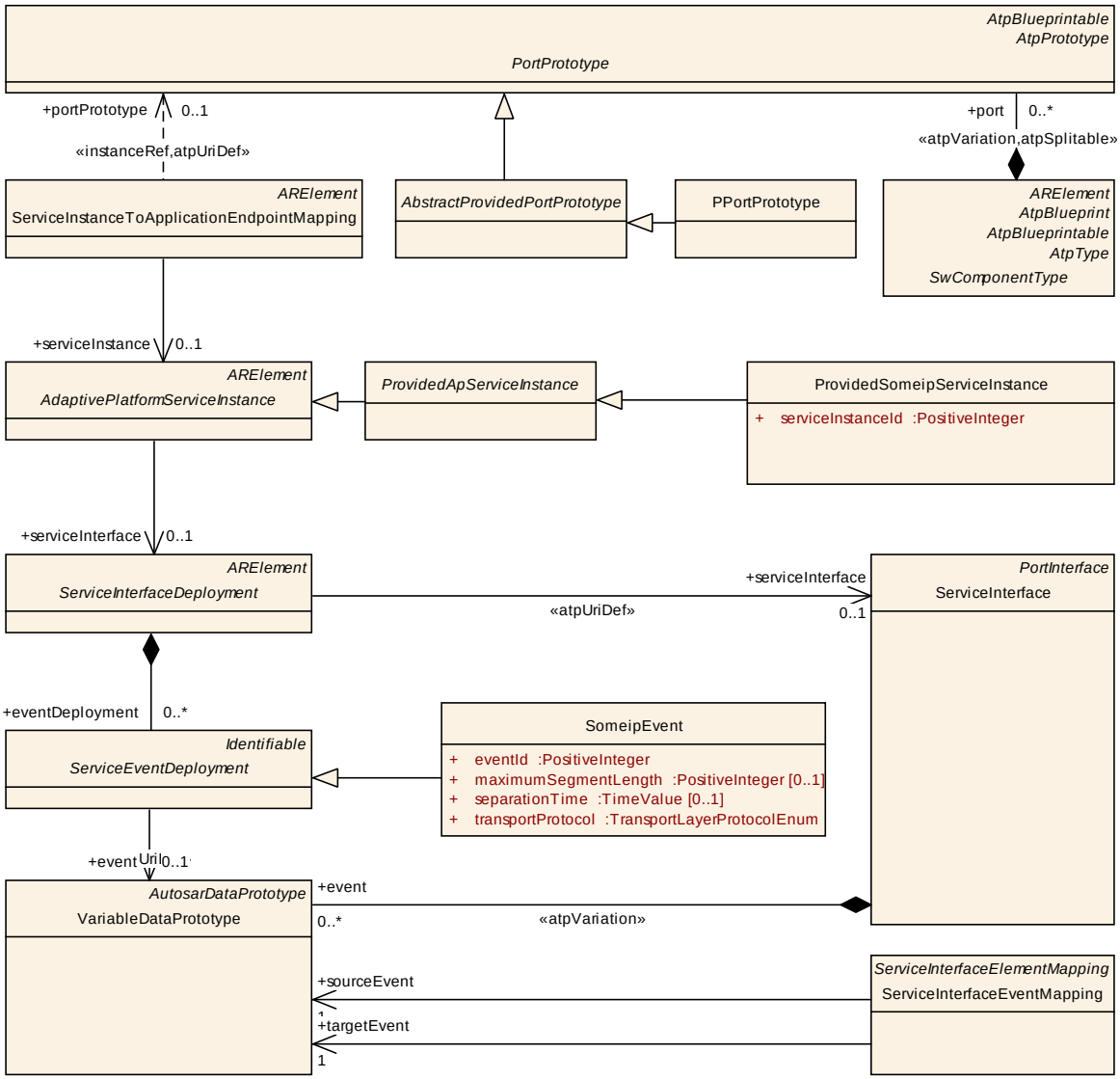

Figure A.3: Excerpt of the relevant meta-classes for the `ServiceInterfaceEventMapping` example

By mapping individual elements of `ServiceInterface`s, it is possible to map element with different `shortName`s to each other. In this example, the `event` with the `shortName` `event1` is mapped to another `event` with the `shortName` `eventLeft`.

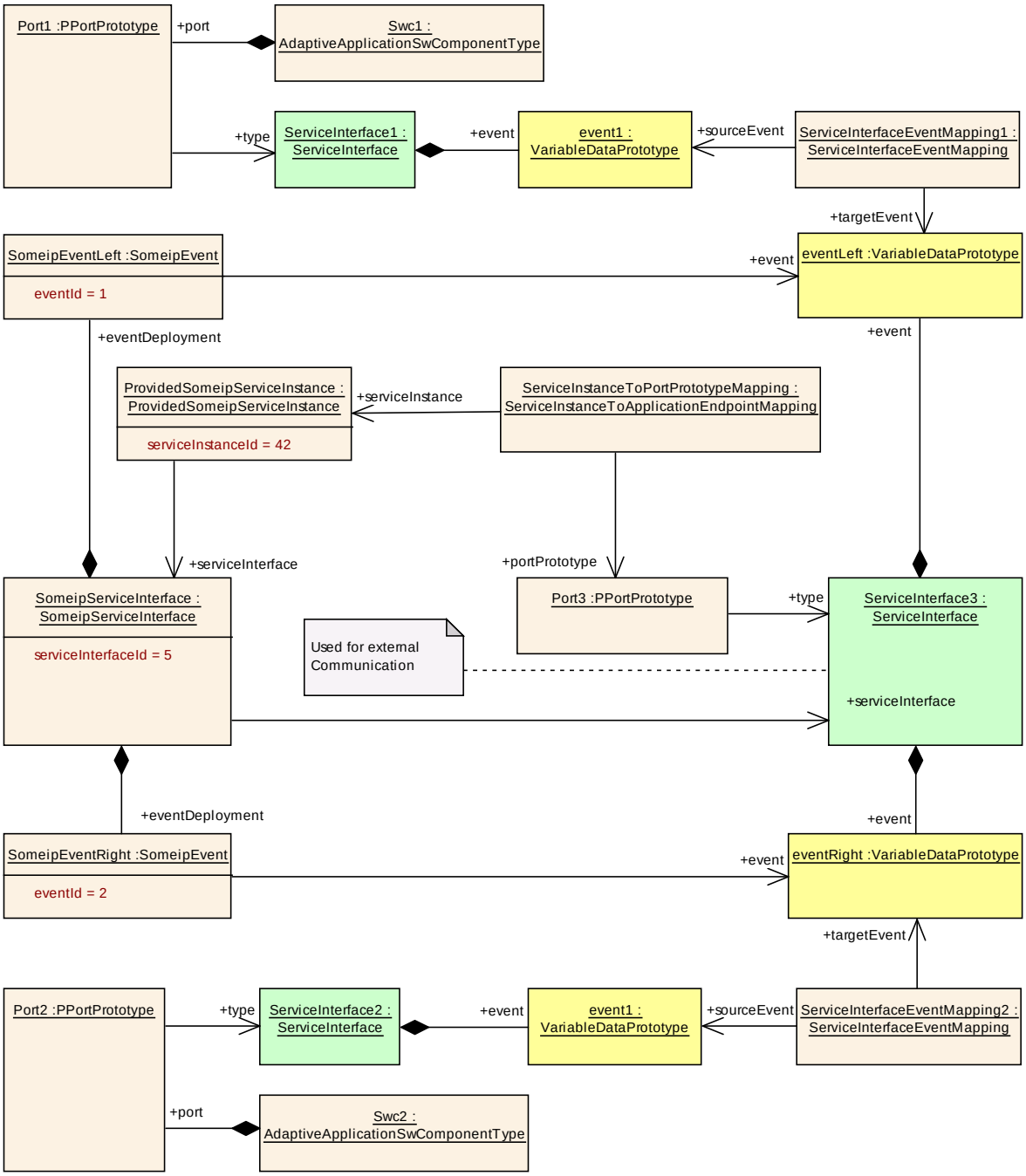

Figure A.4: Example for the deployment of a service in the presence of a `ServiceInterfaceEventMapping`

In Figure A.4, two different `ServiceInterface`s exist that each aggregate an `event` with the identical `shortName`. This scenario **requires** the existence of `ServiceInterfaceElementMapping`s.

As an extension to the scenario depicted in Figure A.4, Figure A.5 describes a model where the **same** `event` of a `ServiceInterface` is used in two different event deployments by means of two `ServiceInterfaceEventMapping`s that each refer to said `event` in the role `ServiceInterfaceEventMapping.sourceEvent`.

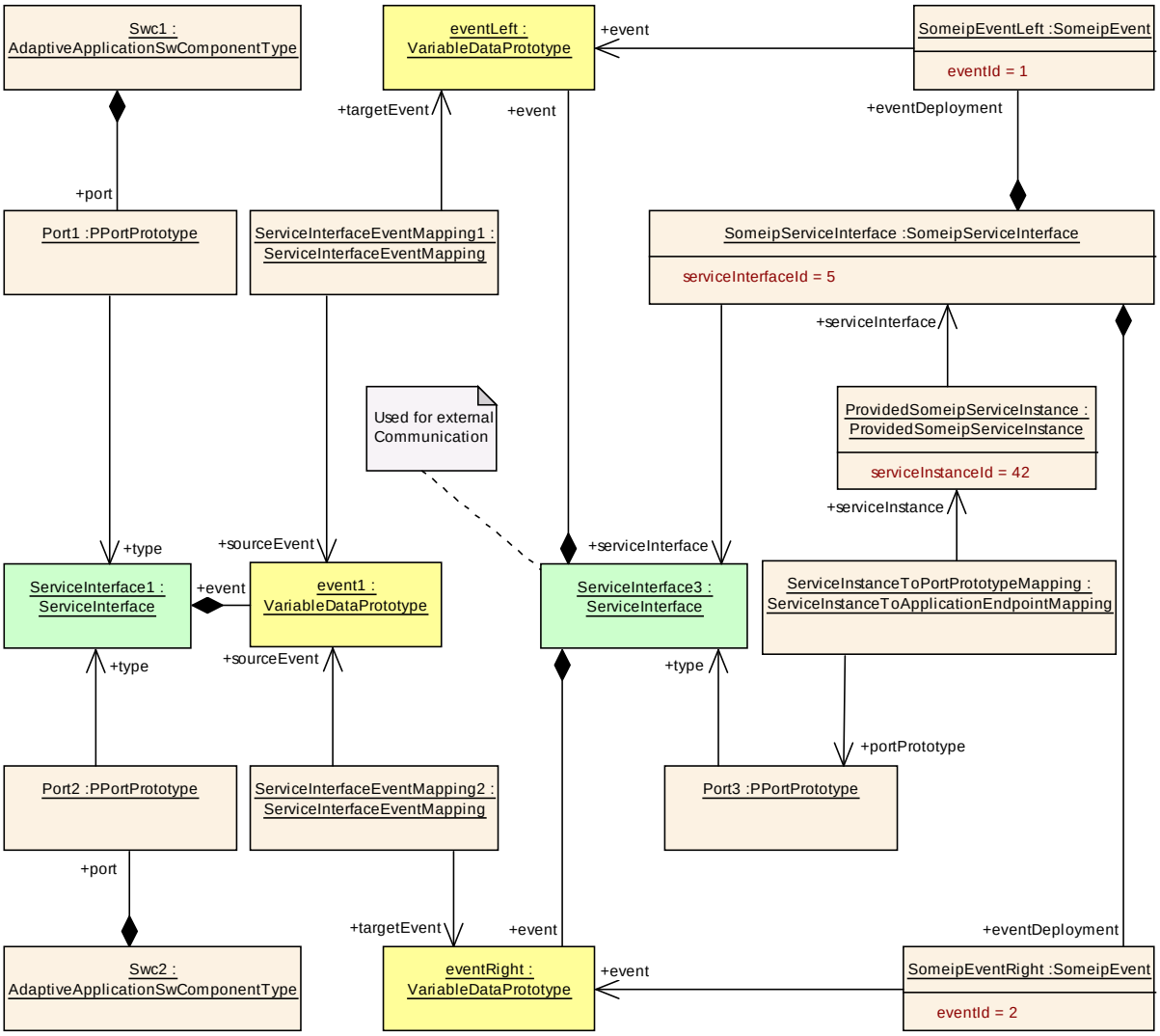

Figure A.5: Example for the deployment of a service in the presence of a `ServiceInterfaceEventMapping` to the same source `ServiceInterface`

Again, this scenario **requires** the existence of appropriately configured `ServiceInterfaceElementMapping`s.

A.3 Definition of Startup Configuration

As already mentioned, the startup configuration is directly aggregated by the definition of a `Process`:

```
<PROCESS>
  <SHORT-NAME>AA1</SHORT-NAME>
  <MODE-DEPENDENT-STARTUP-CONFIGS>
    <MODE-DEPENDENT-STARTUP-CONFIG>
      <EXECUTION-DEPENDENCYS>
        <EXECUTION-DEPENDENCY>
          <APPLICATION-MODE-IREF>
```

```
          <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
    DECLARATION-GROUP-PROTOTYPE">/Processes/MWC/ApplicationStateMachine</
    CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
          <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
    ModeDeclarationGroups/ApplicationStateMachine/Running</TARGET-MODE-
    DECLARATION-REF>
        </APPLICATION-MODE-IREF>
      </EXECUTION-DEPENDENCY>
      <EXECUTION-DEPENDENCY>
        <APPLICATION-MODE-IREF>
          <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
    DECLARATION-GROUP-PROTOTYPE">/Processes/MSM/ApplicationStateMachine</
    CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
          <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
    ModeDeclarationGroups/ApplicationStateMachine/Running</TARGET-MODE-
    DECLARATION-REF>
        </APPLICATION-MODE-IREF>
      </EXECUTION-DEPENDENCY>
    </EXECUTION-DEPENDENCYS>
    <MACHINE-MODE-IREFS>
      <MACHINE-MODE-IREF>
        <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
    DECLARATION-GROUP-PROTOTYPE">/Machines/ExampleMachine/
    ExampleMachine_StateMachine</CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-
    REF>
        <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
    ModeDeclarationGroups/VehicleStateMachine/Driving</TARGET-MODE-
    DECLARATION-REF>
      </MACHINE-MODE-IREF>
    </MACHINE-MODE-IREFS>
    <STARTUP-CONFIG-REF DEST="STARTUP-CONFIG">/StartupConfigSets/
    StartupConfigSet_AA/AA1_Startup</STARTUP-CONFIG-REF>
   </MODE-DEPENDENT-STARTUP-CONFIG>
  </MODE-DEPENDENT-STARTUP-CONFIGS>
</PROCESS>
```

Listing A.1: Example for the definition of the `ModeDependentStartupConfig` owned by a `Process`

In this example, launch dependencies exist on two other `Process`es. Both `Process`es MWC and MSM need to be in the ApplicationState "Running" before AA1 is started.

The reference `ModeDependentStartupConfig.machineMode` refers to a `ModeDeclaration` with the `shortName` `Driving` within the state machine of the underlying `Machine`.

The referenced `StartupConfig` is defined in Listing A.2.

```
<STARTUP-CONFIG>
  <SHORT-NAME>AA1_Startup</SHORT-NAME>
  <RESOURCE-GROUP-REFS>
    <RESOURCE-GROUP-REF DEST="RESOURCE-GROUP">/Machines/ExampleMachine/
    Linux/limitcpu</RESOURCE-GROUP-REF>
    <RESOURCE-GROUP-REF DEST="RESOURCE-GROUP">/Machines/ExampleMachine/
    Linux/limitmem</RESOURCE-GROUP-REF>
  </RESOURCE-GROUP-REFS>
```

```
  <SCHEDULING-POLICY>SCHEDULING-POLICY-FIFO</SCHEDULING-POLICY>
  <SCHEDULING-PRIORITY>20</SCHEDULING-PRIORITY>
  <STARTUP-OPTIONS>
    <STARTUP-OPTION>
      <OPTION-ARGUMENT>inputfile_1</OPTION-ARGUMENT>
      <OPTION-KIND>COMMAND-LINE-LONG-FORM</OPTION-KIND>
      <OPTION-NAME>filename</OPTION-NAME>
    </STARTUP-OPTION>
  </STARTUP-OPTIONS>
</STARTUP-CONFIG>
```

Listing A.2: Example for a `StartupConfig`

Please note that the definition of the `StartupOption` in the example yields an actual command-line option that reads `--filename=inputfile_1`.

The corresponding definition of a `Machine` contains a `OsModuleInstantiation` that in turn owns the two `ResourceGroup`s named `limitcpu` and `limitmem`. This aspect can be found in Listing A.3.

```
<MACHINE>
  <SHORT-NAME>ExampleMachine</SHORT-NAME>
  <MACHINE-MODE-MACHINES>
    <MODE-DECLARATION-GROUP-PROTOTYPE>
      <SHORT-NAME>ExampleMachine_StateMachine</SHORT-NAME>
      <TYPE-TREF DEST="MODE-DECLARATION-GROUP">/ModeDeclarationGroups/
   VehicleStateMachine</TYPE-TREF>
    </MODE-DECLARATION-GROUP-PROTOTYPE>
  </MACHINE-MODE-MACHINES>
  <MODULE-INSTANTIATIONS>
    <OS-MODULE-INSTANTIATION>
      <SHORT-NAME>Linux</SHORT-NAME>
      <RESOURCE-GROUPS>
        <RESOURCE-GROUP>
          <SHORT-NAME>limitcpu</SHORT-NAME>
          <DESC>
            <L-2 L="EN">Limits the cpu shares available to processes in
   this cgroup to 10.</L-2>
          </DESC>
        </RESOURCE-GROUP>
        <RESOURCE-GROUP>
          <SHORT-NAME>limitmem</SHORT-NAME>
          <DESC>
            <L-2 L="EN">Limits memory available to the cgroup processes to
   50MB. </L-2>
          </DESC>
        </RESOURCE-GROUP>
      </RESOURCE-GROUPS>
    </OS-MODULE-INSTANTIATION>
  </MODULE-INSTANTIATIONS>
</MACHINE>
```

Listing A.3: Example for the definition of a `Machine`

A.4 Service Instance Mapping

This section contains some examples that explain the modeling of a mapping between a service instance and the application. The examples have been created to show both the "find" and the "offer" side of the service binding.

In the first example, depicted in Figure A.6 shows the binding of PortPrototypes to a SOME/IP-based transport layer. The left part of the diagram contains the modeling of the “find” aspect and the right part contains the modeling of the “offer” aspect.

Please note that the shortNames of the two affected PortPrototypes are different. In other words, the shortNames of the PortPrototypes are not used as a way to identify the opposite end of the service binding.

Instead, the existence of a ServiceInstanceToApplicationEndpointMapping that maps a PortPrototype to a ProvidedSomeipServiceInstance resp. RequiredSomeipServiceInstance with the **identical value** of attribute serviceInstanceId creates the actual binding between the “find” and the “offer” end.

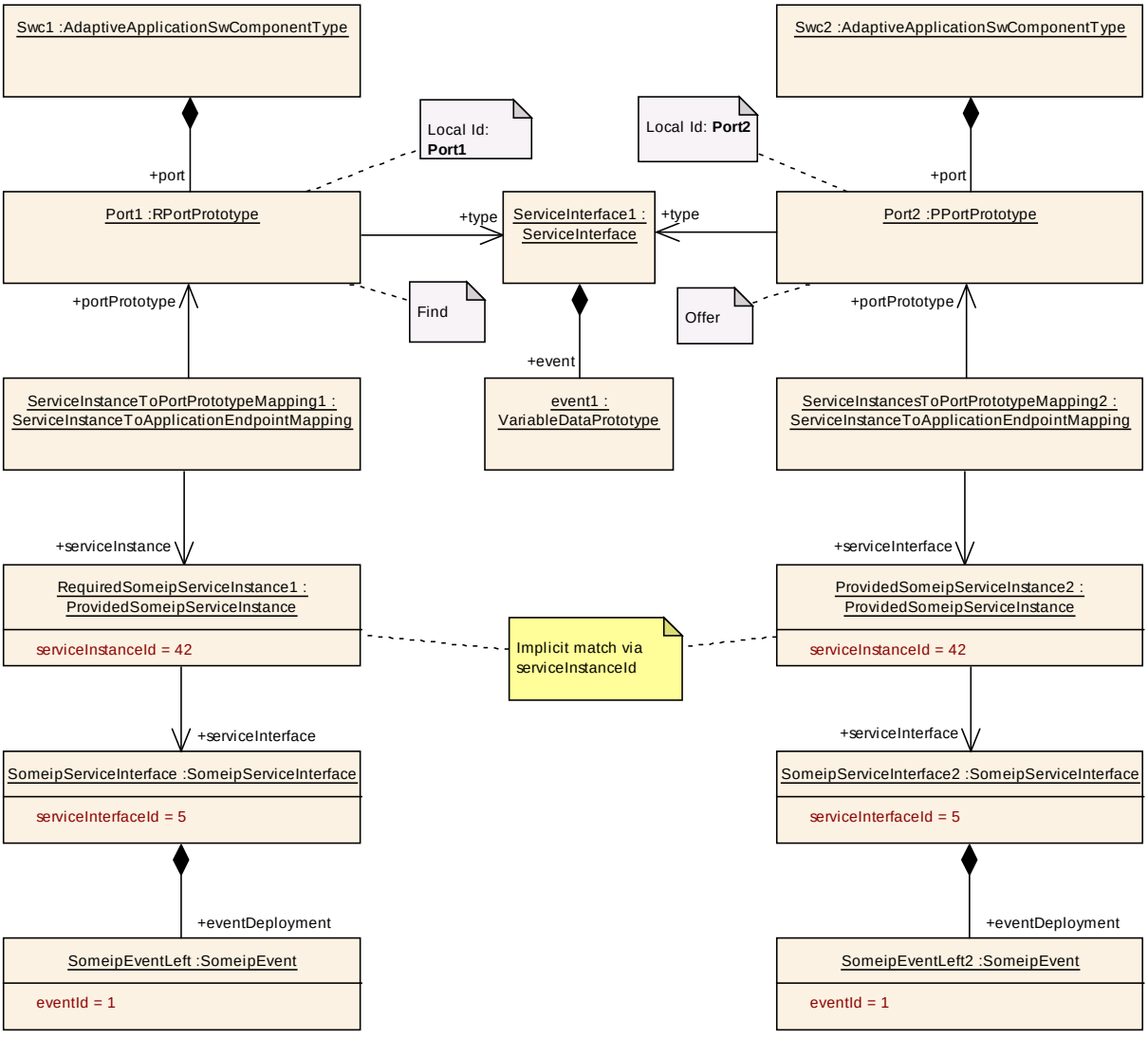

Figure A.6: Port-based binding of a service instance to the application using SOME/IP

The next example (depicted in Figure A.7) shows a binding of PortPrototypes to a user-defined transport layer. The left part of the diagram contains the modeling of the “find” aspect and the right part contains the modeling of the “offer” aspect.

Because the binding is user-defined, there are no attributes modeled on the level of the meta-model available to identify an instance according to the user-defined service implementation. There is just no way to define attributes that are “needed anyway” for a user-defined binding.

Therefore, the only option in this case it the usage of AdminData, Sdg, and Sd to define an identification of the user-defined transport layer.

In order to support the comparison to the example depicted in Figure A.6, the example described in Figure A.7 uses a simple identification based on a numerical value. Again, this is an arbitrary scenario created just for the sake of explanation.

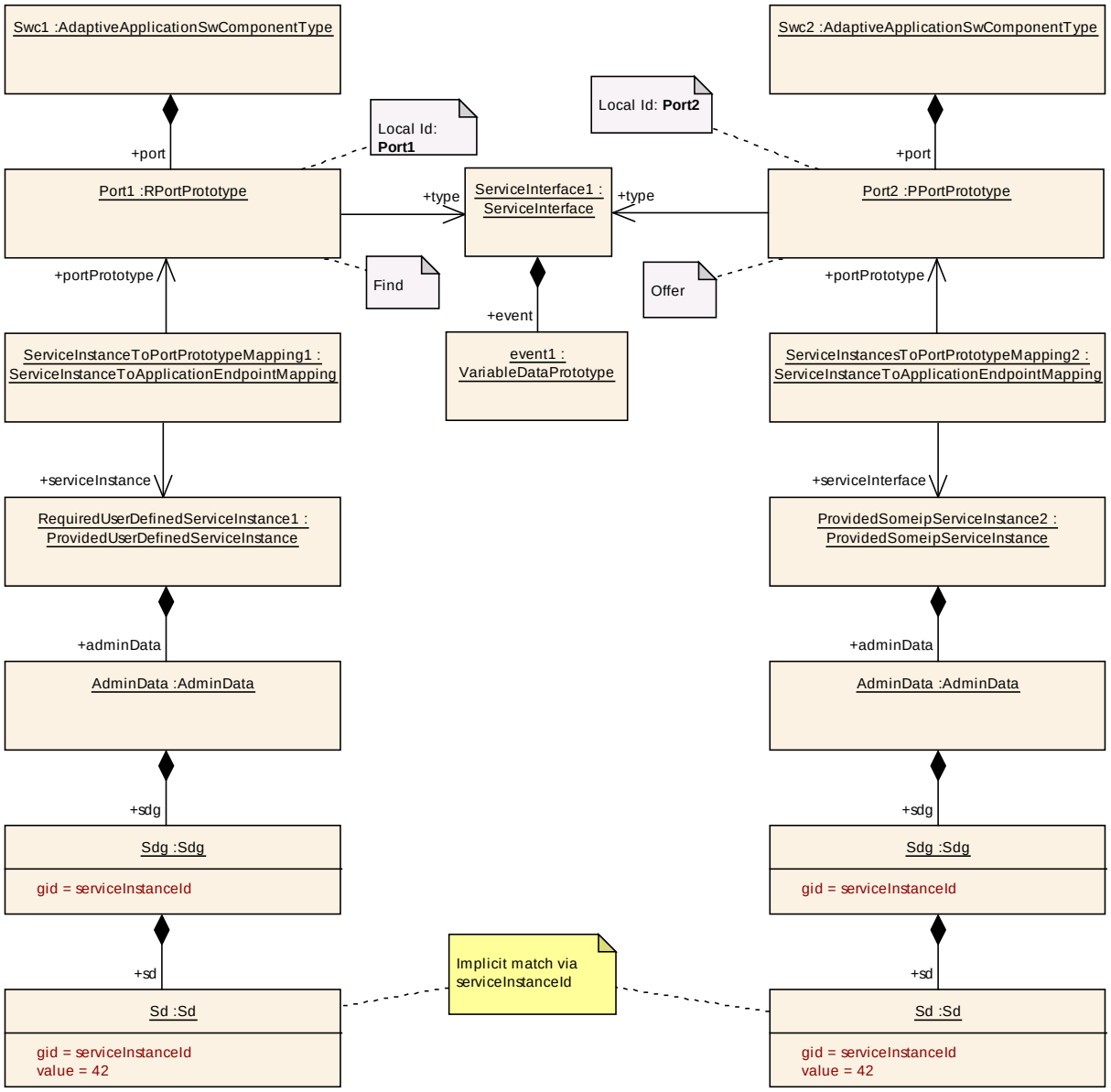

Figure A.7: Port-based binding of a service instance to the application using a user-defined binding

The following example (see Figure A.8) describes the binding of a `TransportLayerIndependentInstanceId` to a SOME/IP-based transport layer. The left part of the diagram contains the modeling of the “find” aspect and the right part contains the modeling of the “offer” aspect.

The basic modeling approach is comparable to the example depicted in Figure A.6 and thereby demonstrates the fact that the `TransportLayerIndependentInstanceId`, at least to some extent, represents a “port without the software-component”.

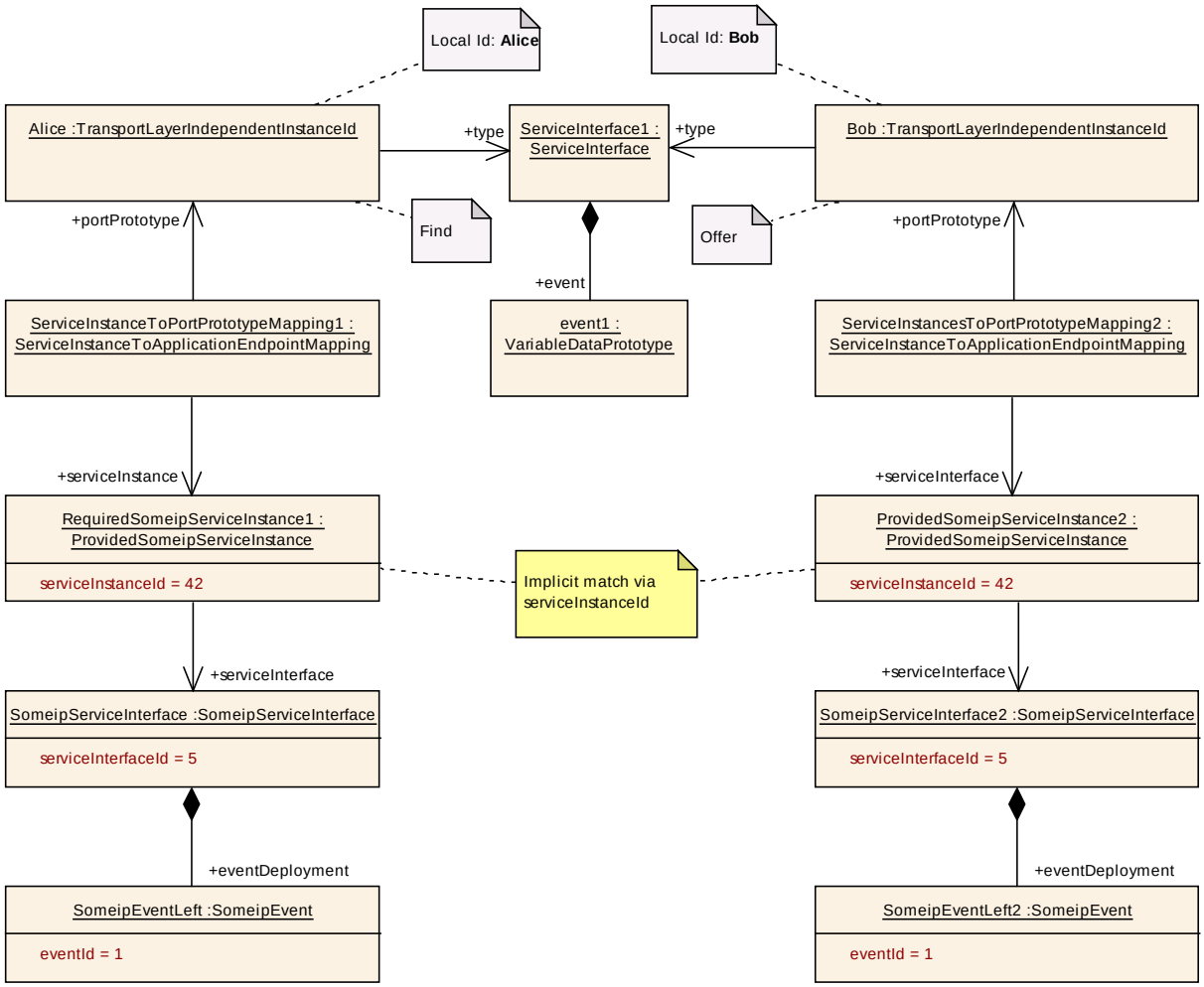

Figure A.8: Binding of a service instance to `TransportLayerIndependentInstanceId` using SOME/IP

For the sake of completion, there is another example that addresses the binding of a `TransportLayerIndependentInstanceId` to a user-defined transport layer. The left part of the diagram contains the modeling of the “find” aspect and the right part contains the modeling of the “offer” aspect. Please find more details in Figure A.9.

Please note that the modeling of the user-defined service identification is (again) completely heuristic and most likely does not represent any realistic approach to a user-defined binding.

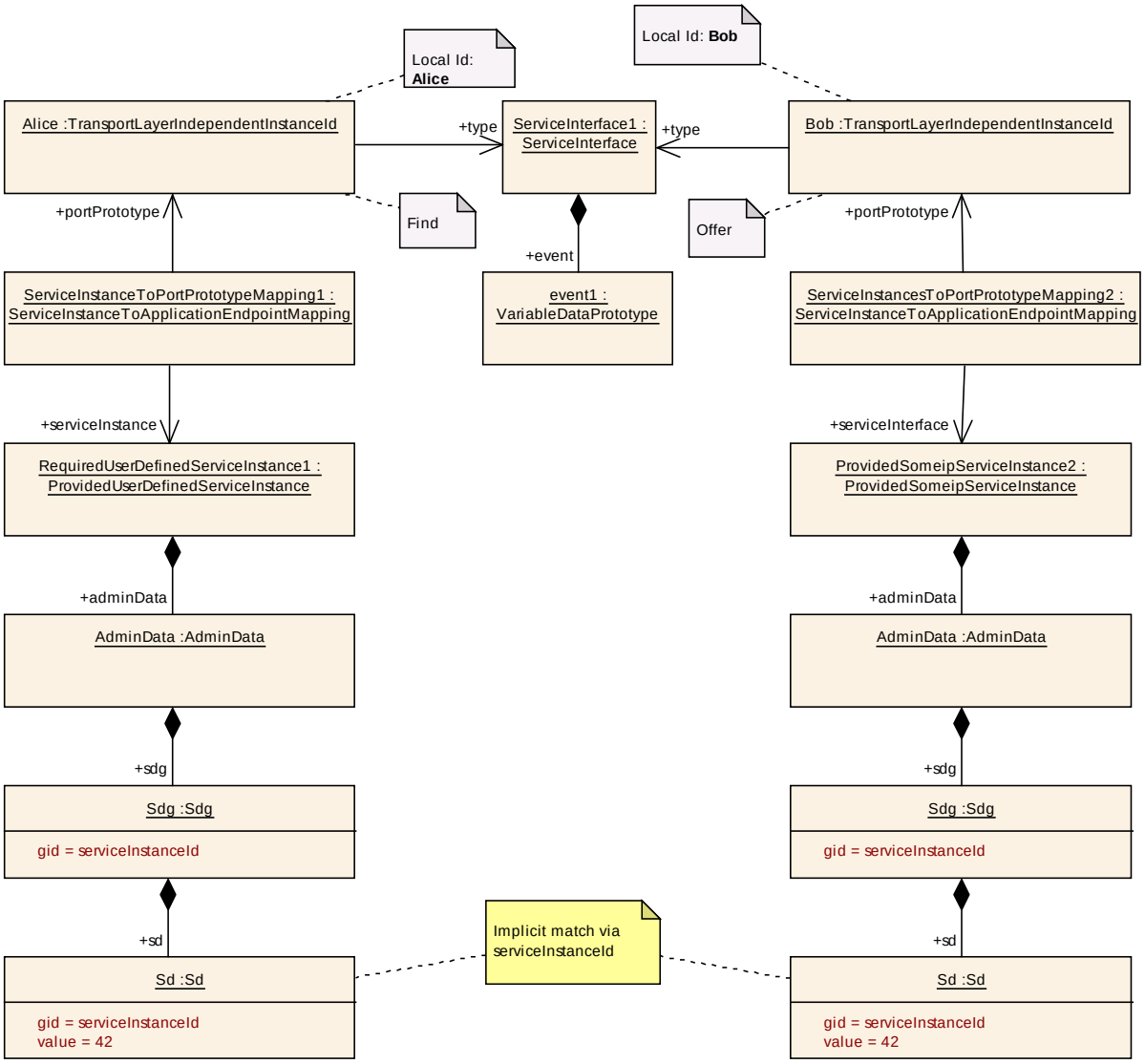

Figure A.9: Binding of a service instance to to `TransportLayerIndependentInstanceId` using a user-defined binding

A.5 Radar and Camera ServiceInterface example

The example in figure A.10 shows a *Radar* `ServiceInterface` with a *BrakeEvent* and two `method`s: *Calibrate* and *Adjust*. The *Camera* `ServiceInterface` shown in figure A.11 has two events: *LaneEvent* and *SpeedLimitEvent* and one *Calibrate* `method`.

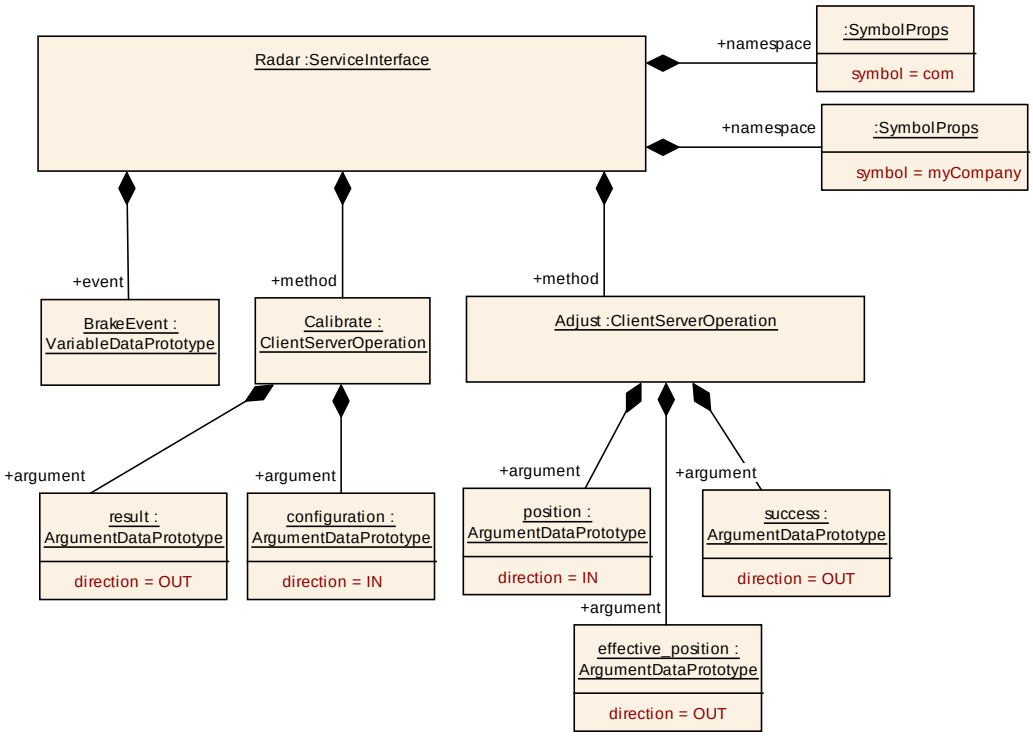

Figure A.10: Radar Service Interface

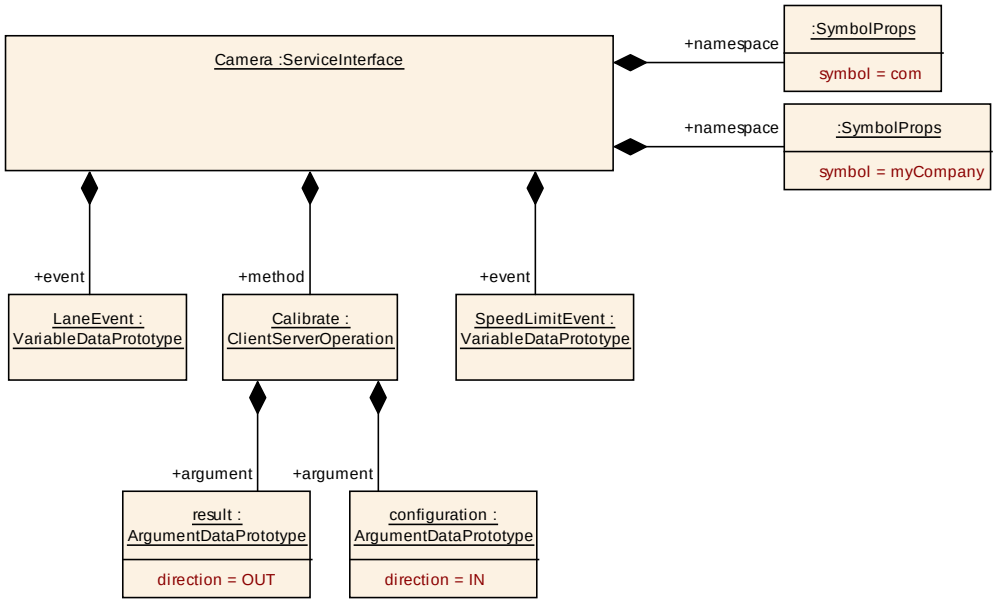

Figure A.11: Camera Service Interface

Both ServiceInterfaces *Radar* and *Camera* are mapped to a combined *RadarAndCamera* ServiceInterface with an Service Interface Element Mapping since both ServiceInterfaces have a method with the same name: *Calibrate*.

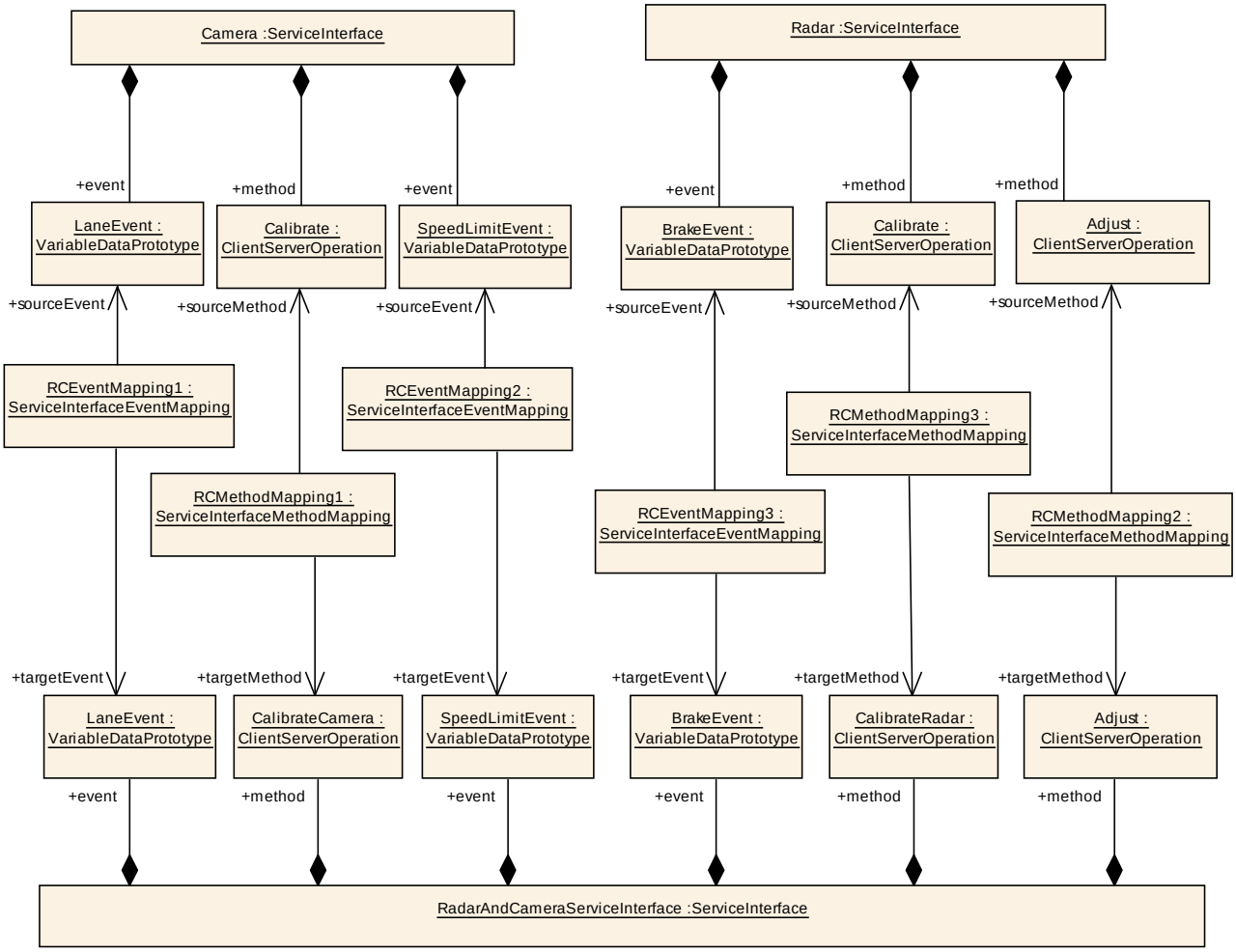

Figure A.12: Service Interface Element Mapping example

The combined ServiceInterface is offered over the network as a SOME/IP Service. Figure A.13 shows the assignment of the SOME/IP serviceInterfaceId to 31.

In addition SOME/IP eventIds are assigned to the events and methodIds are assigned to the methods. Furthermore a single SomeipEventGroup is defined to which all SomeipEvents of the *RadarAndCamera* ServiceInterface are assigned.

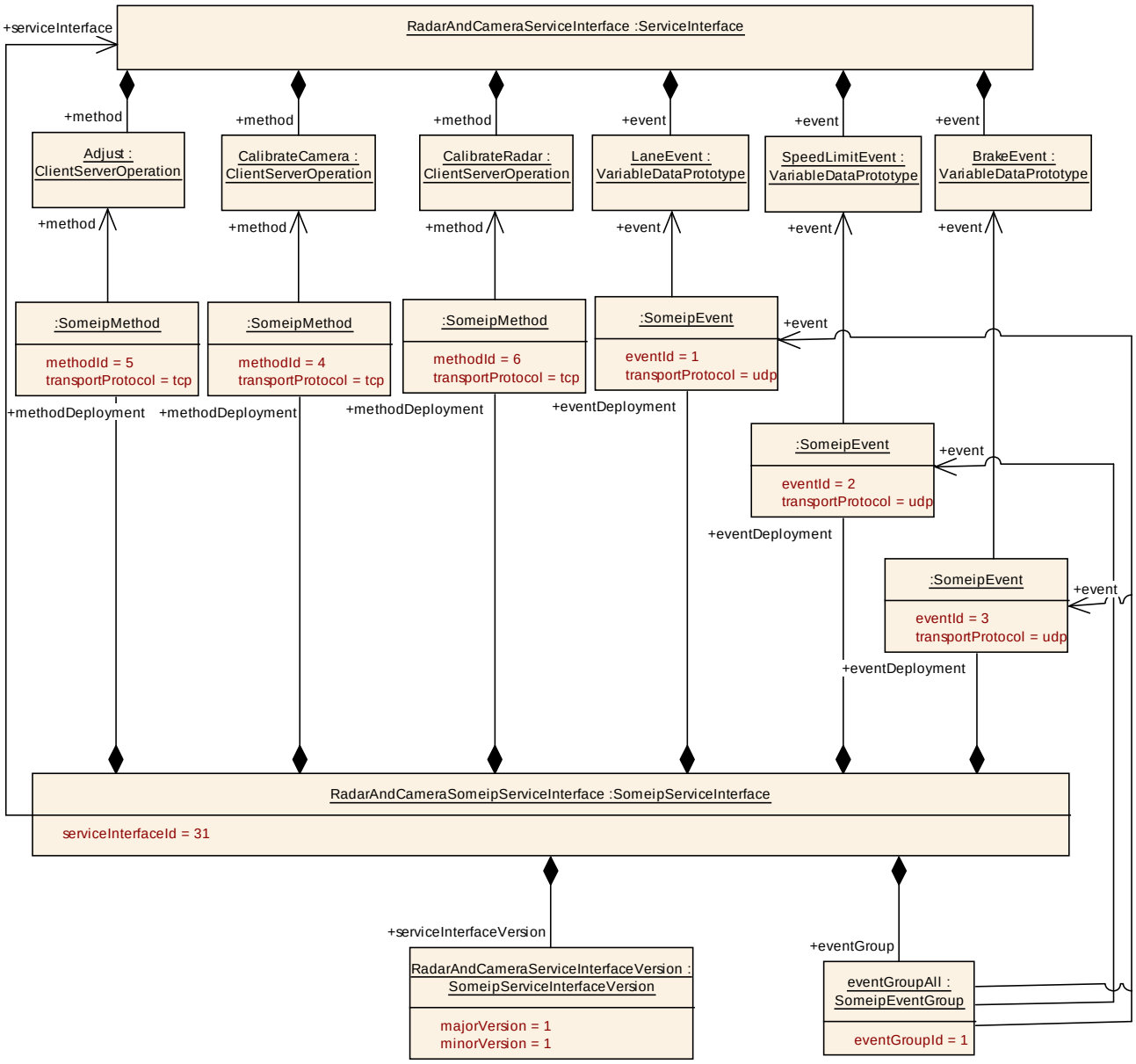

Figure A.13: SOME/IP Deployment

Figure A.14 shows a modeled `ProvidedSomeipServiceInstance` that is mapped to a `Machine`.

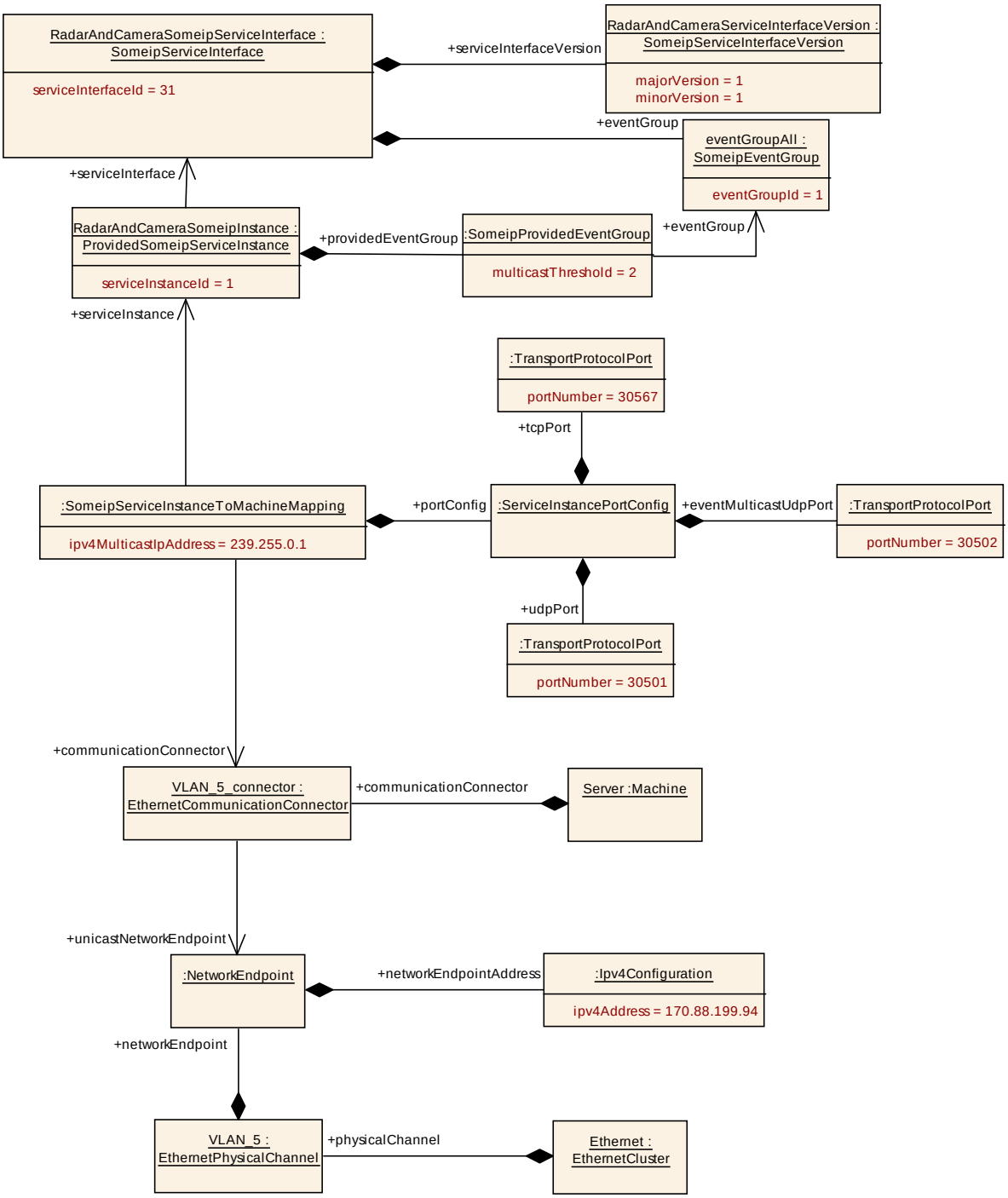

Figure A.14: SOME/IP Provided Service Instance

The displayed configuration in figure A.14 leads to a SOME/IP OfferService Message with the following content:

- ServiceId => `serviceInterfaceId` = 31
- InstanceId => `serviceInstanceId` = 1
- MajorVersion => 1

- MinorVersion => 1
- TTL => 3
- IPv4 Endpoint Option with IPv4 Address (170.88.199.94), Protocol (TCP), PortNumber (30567)
- IPv4 Endpoint Option with IPv4 Address (170.88.199.94), Protocol (UDP), PortNumber (30501)
- IP Multicast Endpoint Option with IPv4 Address (239.255.0.1), Protocol (UDP), PortNumber (30502)

An example of a `RequiredSomeipServiceInstance` is shown in Figure A.15.

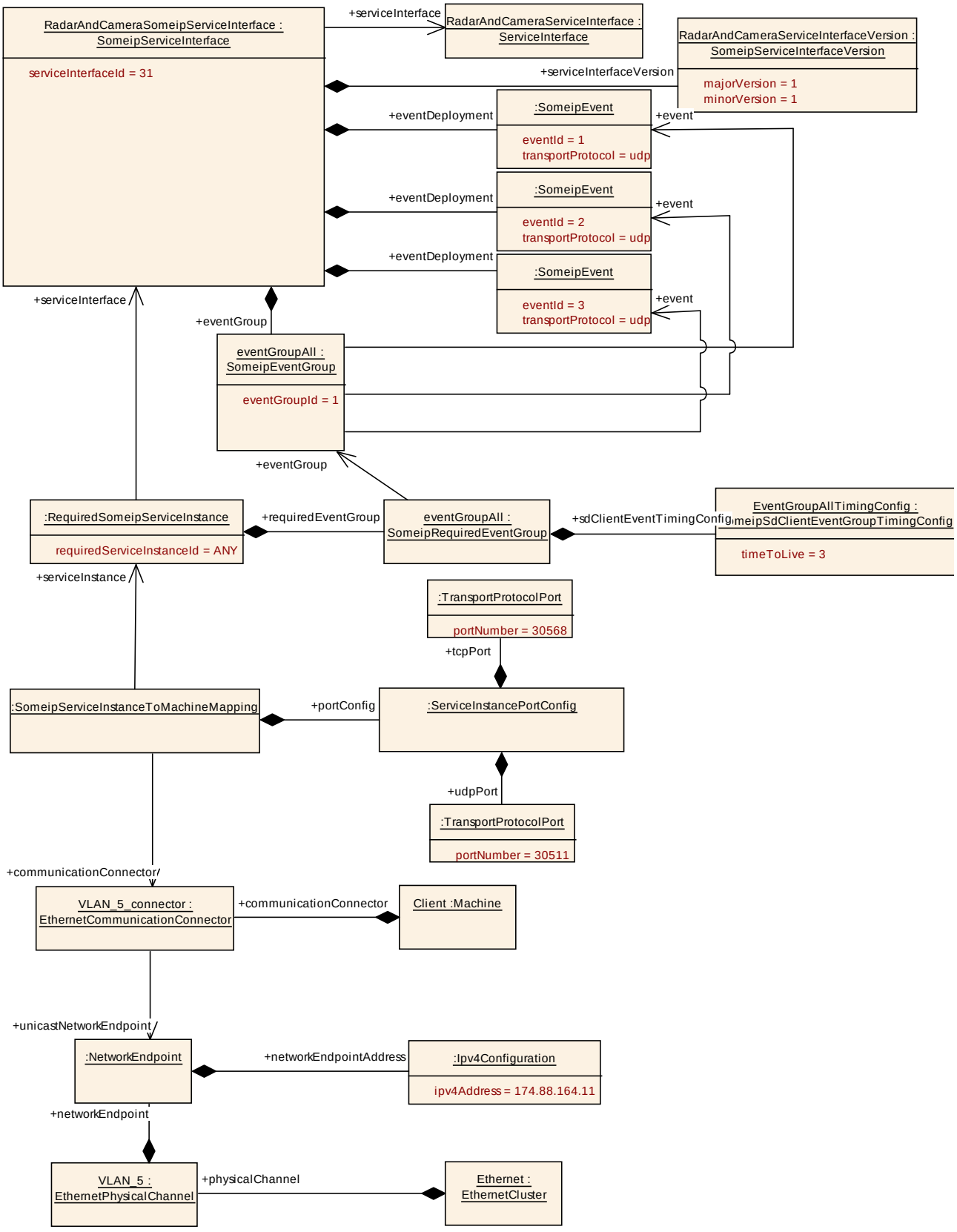

Figure A.15: SOME/IP Required Service Instance

The displayed configuration in figure A.15 leads to a SOME/IP Find Service Message with the following content:

- ServiceId => `serviceInterfaceId` = 31
- InstanceId => `RequiredSomeipServiceInstance.requiredServiceInstanceId` = ANY
- MajorVersion => `majorVersion` = 1
- MinorVersion => `minorVersion` = 1
- TTL => `RequiredSomeipServiceInstance.sdClientConfig.serviceFindTimeToLive` = 3

The displayed configuration in figure A.14 also leads to a SOME/IP SubscribeEventGroup Message content that is send from the Service Requester to the Service Provider:

- ServiceId => taken from the OfferMessage
- InstanceId => taken from the OfferMessage
- MajorVersion => taken from the OfferMessage
- MinorVersion => taken from the OfferMessage
- Eventgroup ID => `RequiredSomeipServiceInstance.requiredEventGroup.eventGroup.eventGroupId` = 1
- TTL => `RequiredSomeipServiceInstance.requiredEventGroup.sdClientEventTimingConfig.timeToLive` = 3
- IPv4 Endpoint Option with IPv4 Address (170.88.164.11), Protocol (UDP), PortNumber (30511)

B General Modeling

This chapter has been created to explain model elements that are not directly related to specific design or deployment usage but have a more general scope. In other words, this chapter describes the structure and usage of some widely reusable modeling content.

B.1 Reference to a DataPrototype in a CompositionSwComponentType

[TPS_MANI_01031] Semantics of `CompositionDataPrototypeRef` ⌈ The meta-class `CompositionDataPrototypeRef` has been created for the following purposes:

- Create a reference to a `DataPrototype` in the context of a `CompositionSwComponentType`. In this case it is not relevant whether the applicable sub-class of `DataPrototype` is typed by an `ApplicationDataType` or an `ImplementationDataType`. The aggregation `CompositionDataPrototypeRef.dataPrototype` shall be used.
- Create a reference to a `DataPrototype` located in a nested `AutosarDataPrototype` in the context of a `CompositionSwComponentType`. In this case it is technically relevant whether the applicable sub-class of `DataPrototype` is typed by an `ApplicationDataType` or an `ImplementationDataType`:
  - If the applicable sub-class of `DataPrototype` is typed by an `ApplicationDataType` then the aggregation in the role `CompositionDataPrototypeRef.dataPrototype` shall be used.
  - If the applicable sub-class of `DataPrototype` is typed by an `ImplementationDataType` then the aggregation in the role `CompositionDataPrototypeRef.elementInImplDatatype` **in addition** to `CompositionDataPrototypeRef.dataPrototype` shall be used.

⌋()

For referencing into the inside of a `ImplementationDataType` it is therefore necessary to use the aggregation `CompositionDataPrototypeRef.dataPrototype` to “get to” the root `DataPrototype` and **then proceed into the guts of the composite `ImplementationDataType`** by means of using `CompositionDataPrototypeRef.elementInImplDatatype`.

[constr_1480] Mutual existence of `CompositionDataPrototypeRef.elementInImplDatatype` vs. attributes of `CompositionDataPrototypeRef.dataPrototype` ⌈ If the aggregation `CompositionDataPrototypeRef.elementInImplDatatype` exists then the following attributes shall not exist:

- `CompositionDataPrototypeRef.dataPrototype.rootDataPrototype`

- CompositionDataPrototypeRef.dataPrototype.contextDataPrototype

⌋()

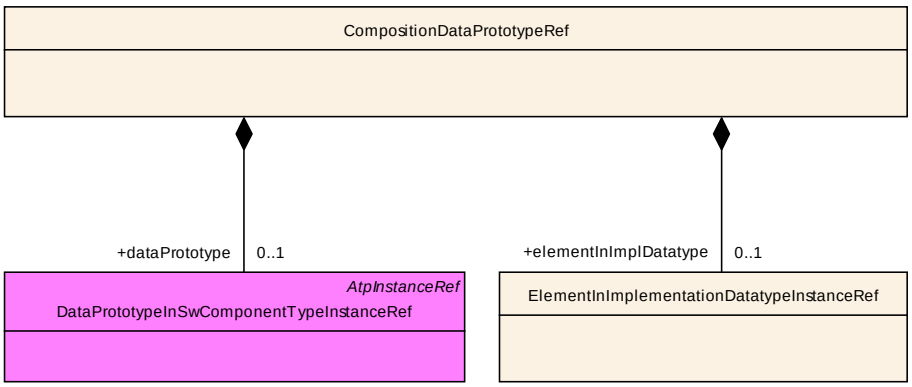

Figure B.1: Modeling of CompositionDataPrototypeRef

[constr_1481] Usage of CompositionDataPrototypeRef in the *AUTOSAR adaptive platform* ⌈ If CompositionDataPrototypeRef is used in the context of the *AUTOSAR adaptive platform* then the actual DataPrototypeInSwComponentTypeInstanceRef.targetDataPrototype shall be either a VariableDataPrototype or an ArgumentDataPrototype. ⌋()

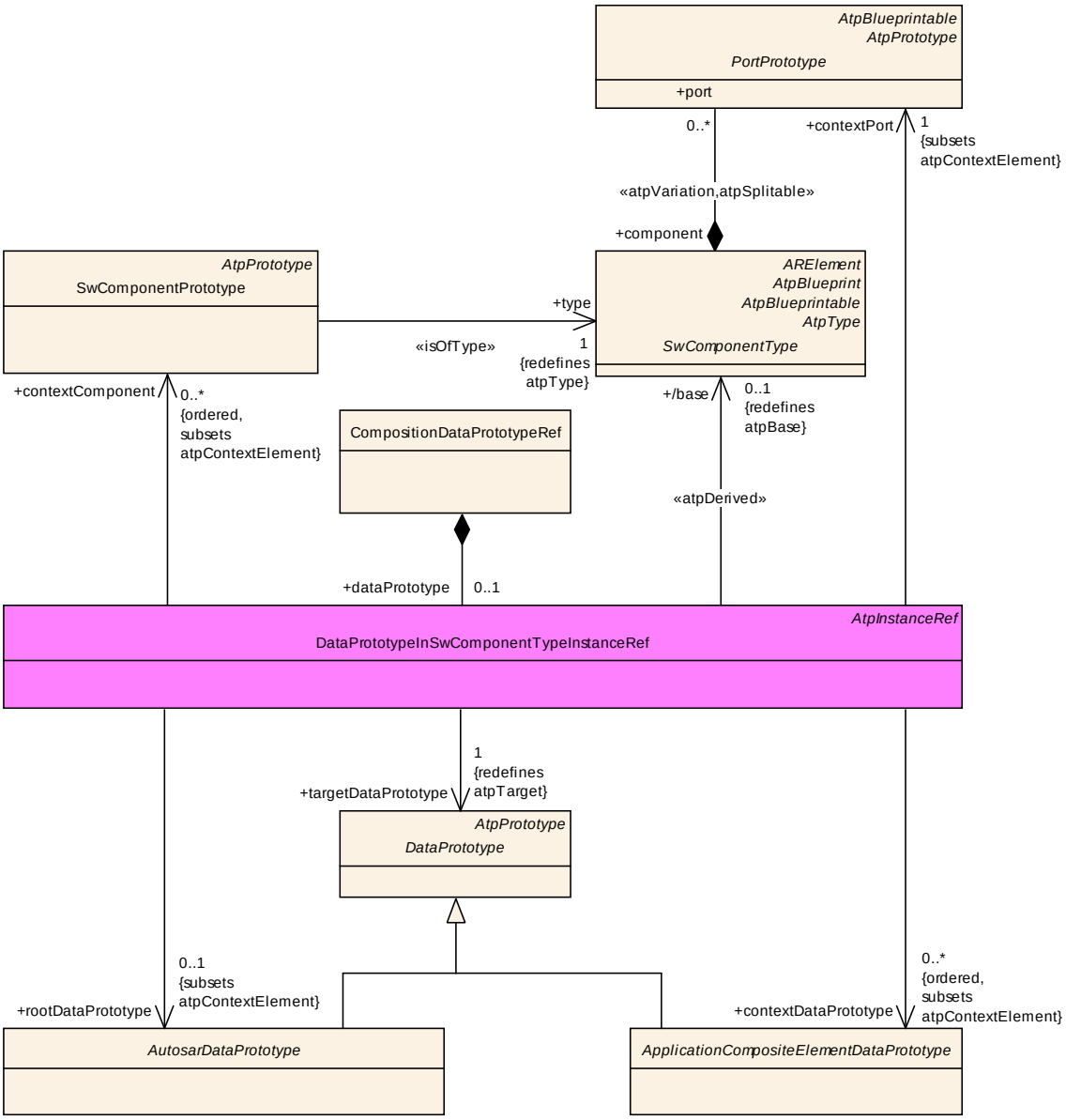

Figure B.2: Modeling of `DataPrototypeInSwComponentTypeInstanceRef`

Class	**CompositionDataPrototypeRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class represents the ability to refer to an AUTOSAR DataPrototype in the context of a CompositionSwComponentType. **Tags:** atp.Status=draft			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataPrototype	DataPrototype	0..1	iref	This attribute shall exist if the InstanceRef points to a DataPrototype typed by an ApplicationDataType. **Tags:** atp.Status=draft

elementInImplDatatype	ElementInImplementationDatatypeInstanceRef	0..1	aggr	This attribute shall exist if the InstanceRef points to a DataPrototype typed by an ImplementationDataType. **Tags:** atp.Status=draft

Table B.1: CompositionDataPrototypeRef

Class	**DataPrototypeInSwComponentTypeInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class represents the ability to: • refer to a DataPrototype in the context of a CompositionSwComponentType. • refer to the internal structure of a DataPrototype in the context of a CompositionSwComponentType. **Tags:** atp.Status=draft			
Base	ARObject,AtpInstanceRef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	SwComponentType	0..1	ref	**Stereotypes:** atpDerived**Tags:** xml.sequenceOffset=10
contextComponent (ordered)	SwComponentPrototype	*	ref	**Tags:** xml.sequenceOffset=20
contextDataPrototype (ordered)	ApplicationCompositeElementDataPrototype	*	ref	**Tags:** xml.sequenceOffset=50
contextPort	PortPrototype	1	ref	**Tags:** xml.sequenceOffset=30
rootDataPrototype	AutosarDataPrototype	0..1	ref	**Tags:** xml.sequenceOffset=40
targetDataPrototype	DataPrototype	1	ref	**Tags:** xml.sequenceOffset=60

Table B.2: DataPrototypeInSwComponentTypeInstanceRef

Class	**ArVariableInImplementationDataInstanceRef**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Data Elements			
Note	This class represents the ability to navigate into a data element inside of an VariableDataPrototype which is typed by an ImplementationDatatype. Note that it shall not be used if the target is the VariableDataPrototype itself (e.g. if its a primitive). Note that this class follows the pattern of an InstanceRef but is not implemented based on the abstract classes because the ImplementationDataType isn't either, especially because ImplementationDataTypeElement isn't derived from AtpPrototype.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
contextDataPrototype (ordered)	ImplementationDataTypeElement	*	ref	This is a context in case there are subelements with explicit types. The reference has to be ordered to properly reflect the nested structure. **Tags:** xml.sequenceOffset=30
portPrototype	PortPrototype	0..1	ref	This is the port providing/receiving the root of the variable **Tags:** xml.sequenceOffset=10
rootVariableDataPrototype	VariableDataPrototype	0..1	ref	This refers to the variableDataPrototype which is typed by the implementationDatatype in which which the target can be found. **Tags:** xml.sequenceOffset=20
targetDataPrototype	ImplementationDataTypeElement	1	ref	This is a context in case there are subelements with explicit types. **Tags:** xml.sequenceOffset=40

Table B.3: ArVariableInImplementationDataInstanceRef

B.2 Modeling of InstanceRefs

This section illustrates the concrete modeling of the instance references used in the previous parts of this document.

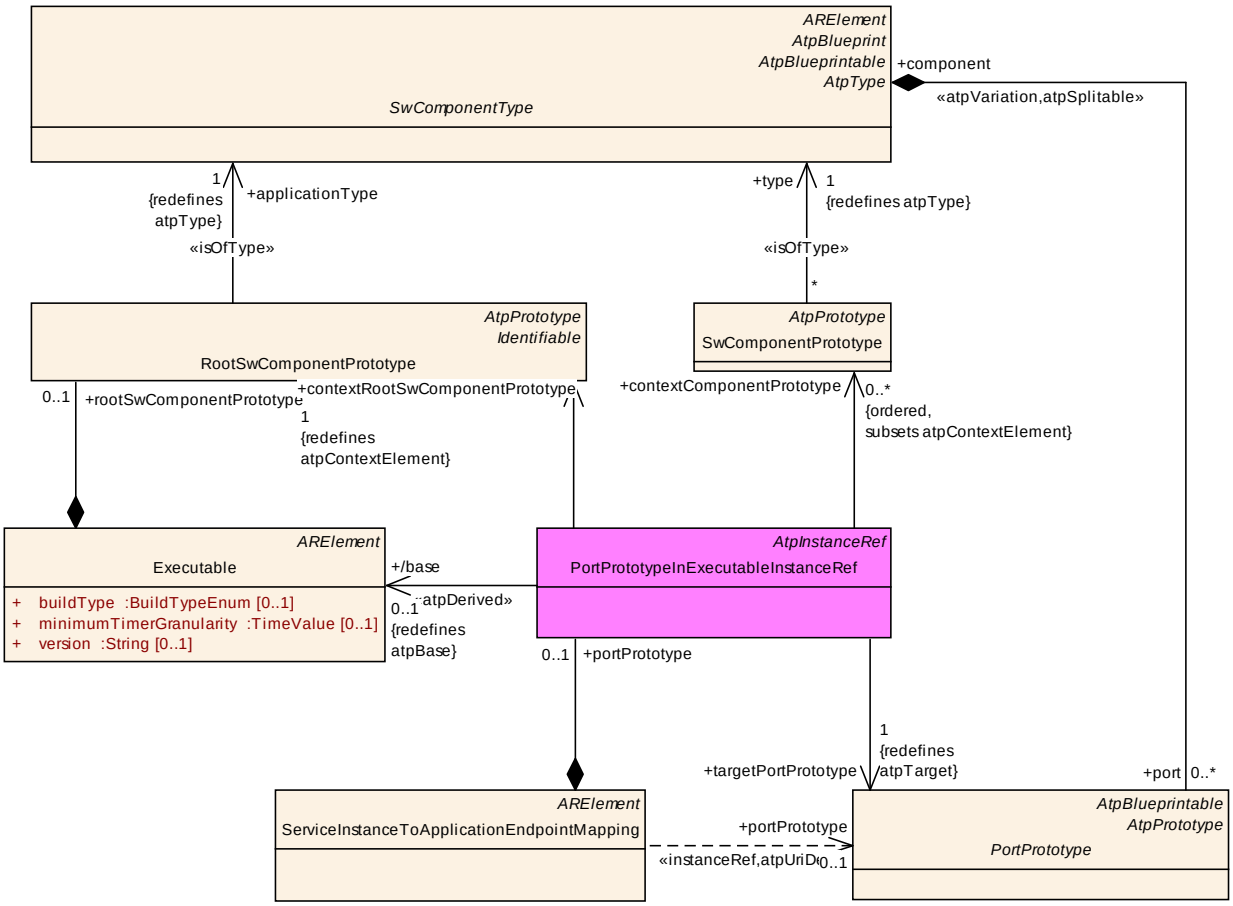

Figure B.3: Modeling of `PortPrototypeInExecutableInstanceRef`

Class	**PortPrototypeInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Process::InstanceRefs			
Note	**Tags:** atp.Status=draft			
Base	ARObject,AtpInstanceRef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived**Tags:** atp.Status=draft xml.sequenceOffset=10
contextComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootSwComponentPrototype	RootSwComponentPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetPortPrototype	PortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table B.4: PortPrototypeInExecutableInstanceRef

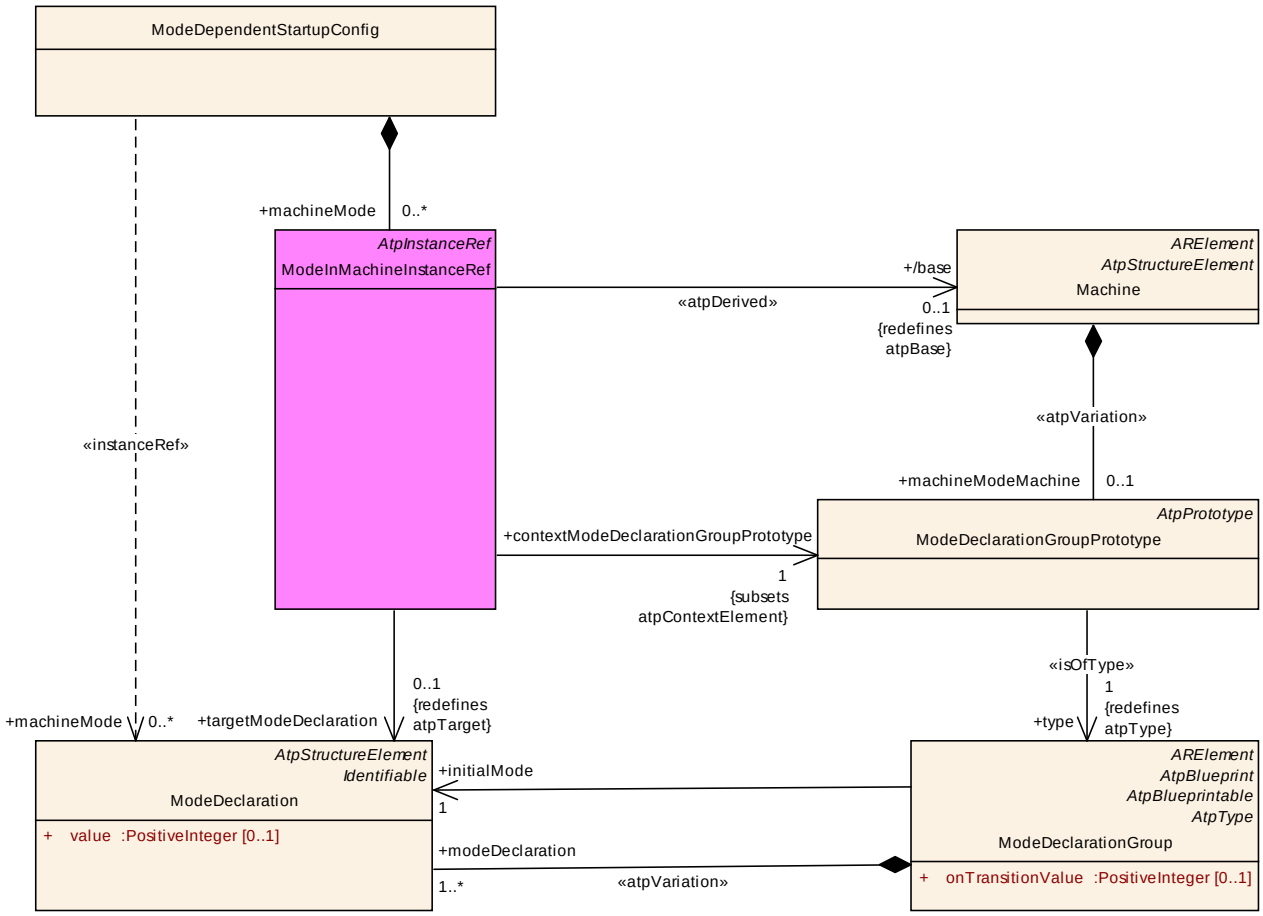

Figure B.4: Modeling of `ModeInMachineInstanceRef`

Class	**ModeInMachineInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Process::InstanceRefs			
Note	**Tags:** atp.Status=draft			
Base	ARObject,AtpInstanceRef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	Machine	0..1	ref	**Stereotypes:** atpDerived**Tags:** atp.Status=draft xml.sequenceOffset=10
contextModeDeclarationGroupPrototype	ModeDeclarationGroupPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetModeDeclaration	ModeDeclaration	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

Table B.5: ModeInMachineInstanceRef

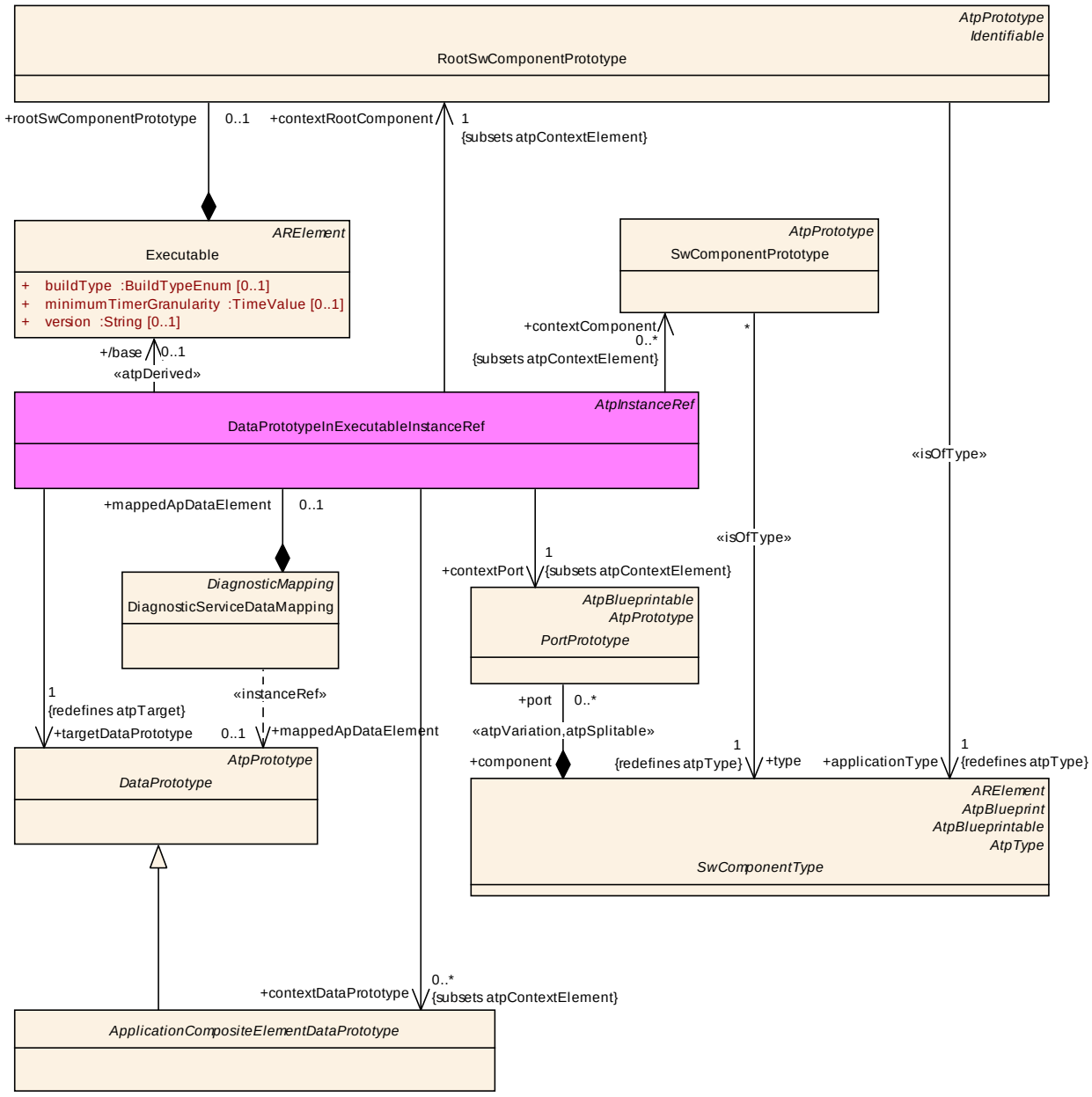

Figure B.5: Modeling of `DiagnosticServiceDataMapping` via `DataPrototypeInExecutableInstanceRef`

Class	**DataPrototypeInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticMapping			
Note	**Tags:** atp.Status=draft			
Base	ARObject,AtpInstanceRef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived**Tags:** atp.Status=draft xml.sequenceOffset=10
contextComponent	SwComponentPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30

contextDataPrototype	ApplicationCompositeElementDataPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=50
contextPort	PortPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40
contextRootComponent	RootSwComponentPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetDataPrototype	DataPrototype	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=60

Table B.6: DataPrototypeInExecutableInstanceRef

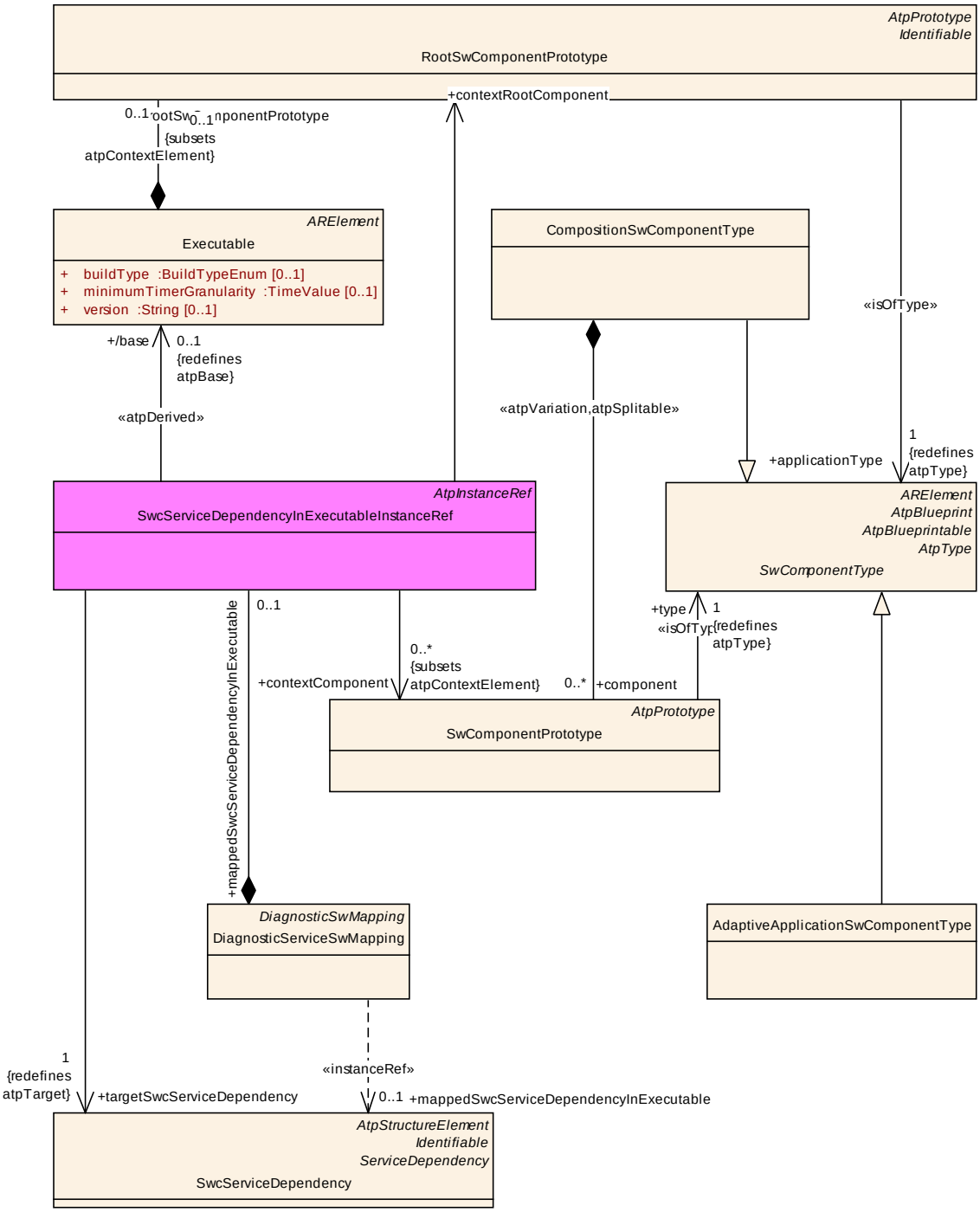

Figure B.6: Modeling of `DiagnosticServiceSwMapping` via `SwcServiceDependencyInExecutableInstanceRef`

Class	**SwcServiceDependencyInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticMapping			
Note	**Tags:** atp.Status=draft			
Base	ARObject,AtpInstanceRef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived**Tags:** atp.Status=draft xml.sequenceOffset=10

contextComponent	SwComponentPrototype	*	ref	**Tags:** atp.Status=draft xml.sequenceOffset=30
contextRootComponent	RootSwComponentPrototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20
targetSwcServiceDependency	SwcServiceDependency	1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=40

Table B.7: SwcServiceDependencyInExecutableInstanceRef

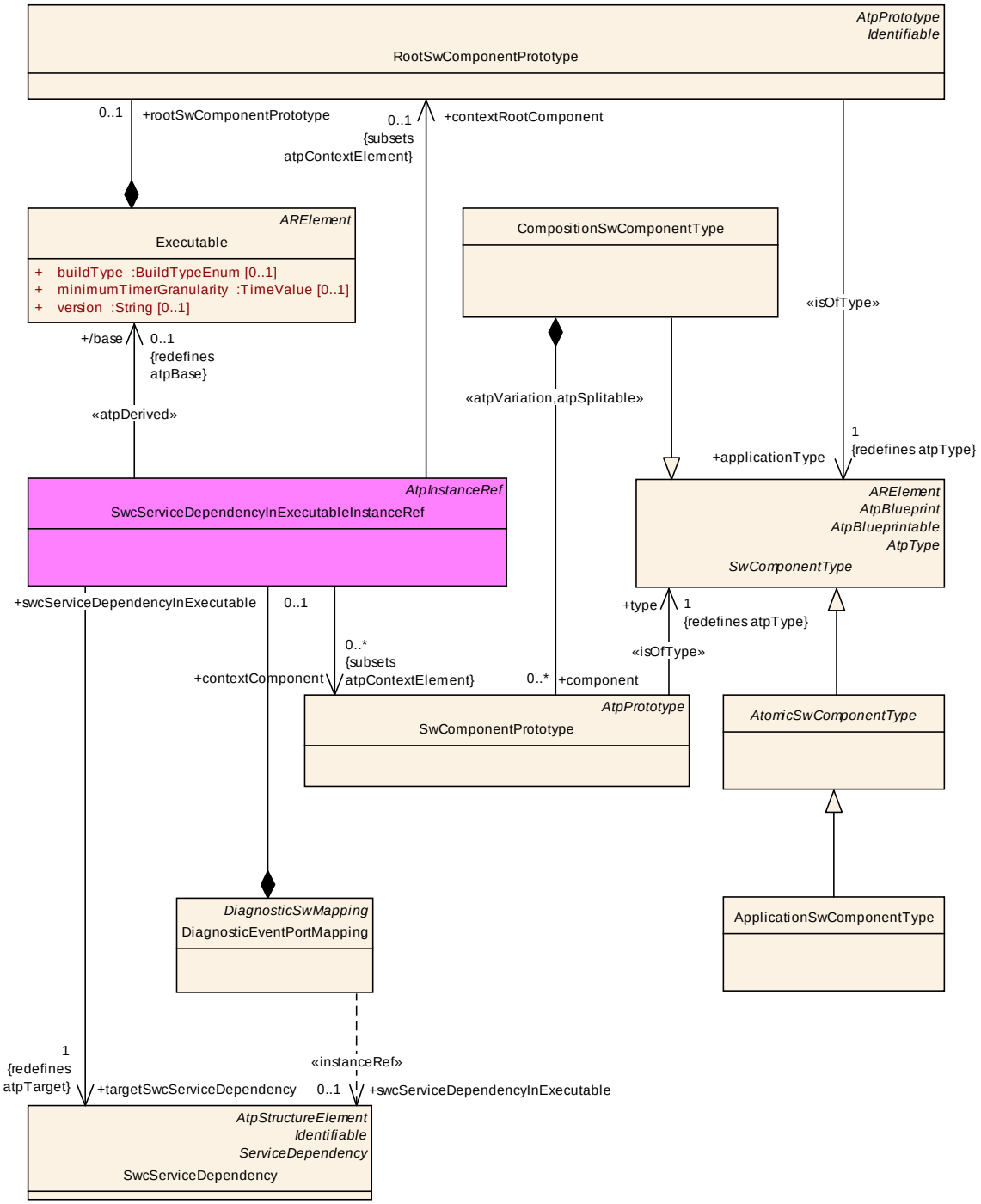

Figure B.7: Modeling of `DiagnosticEventPortMapping` via `SwcServiceDependencyInExecutableInstanceRef`

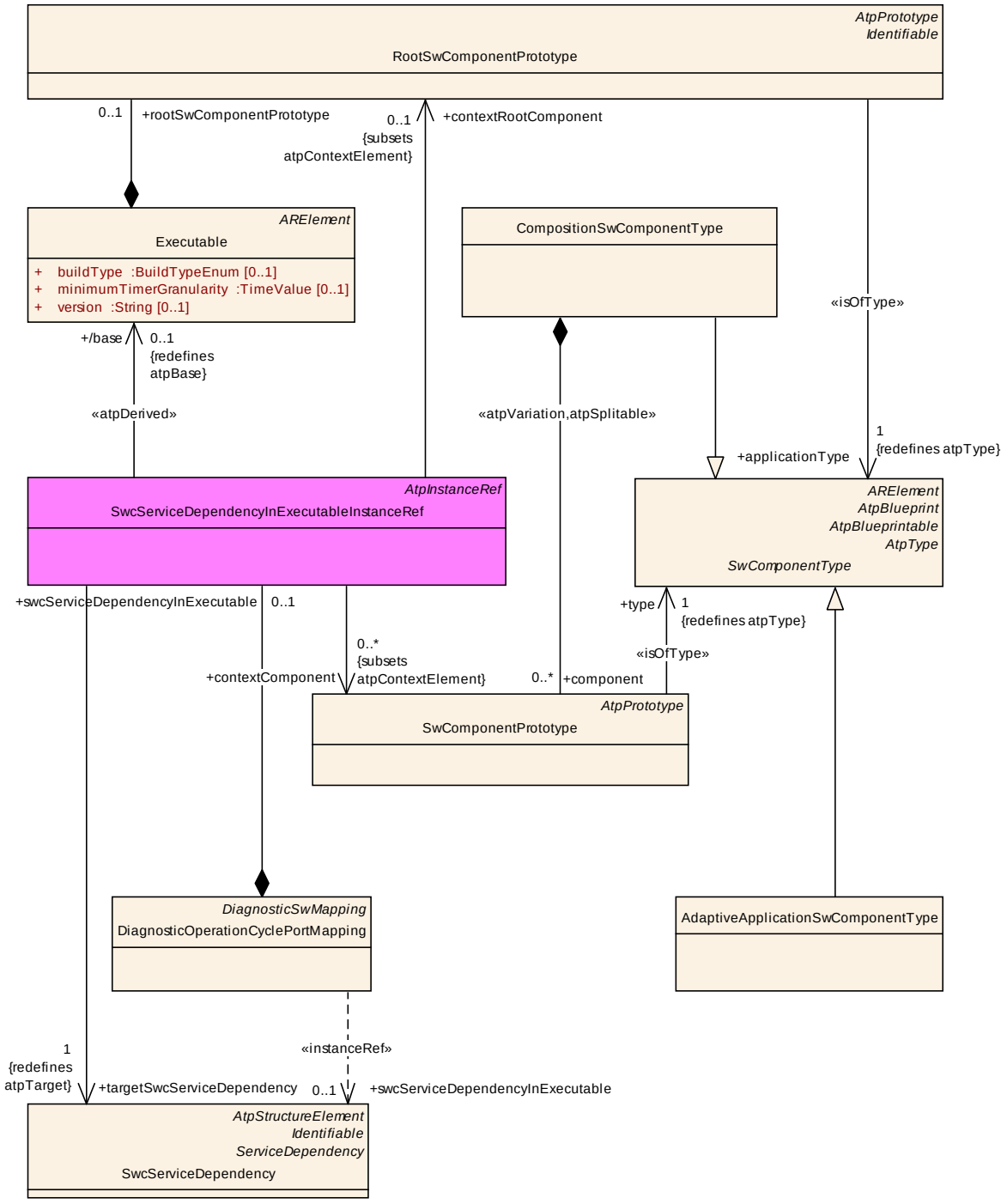

Figure B.8: Modeling of `DiagnosticOperationCyclePortMapping` via `SwcServiceDependencyInExecutableInstanceRef`

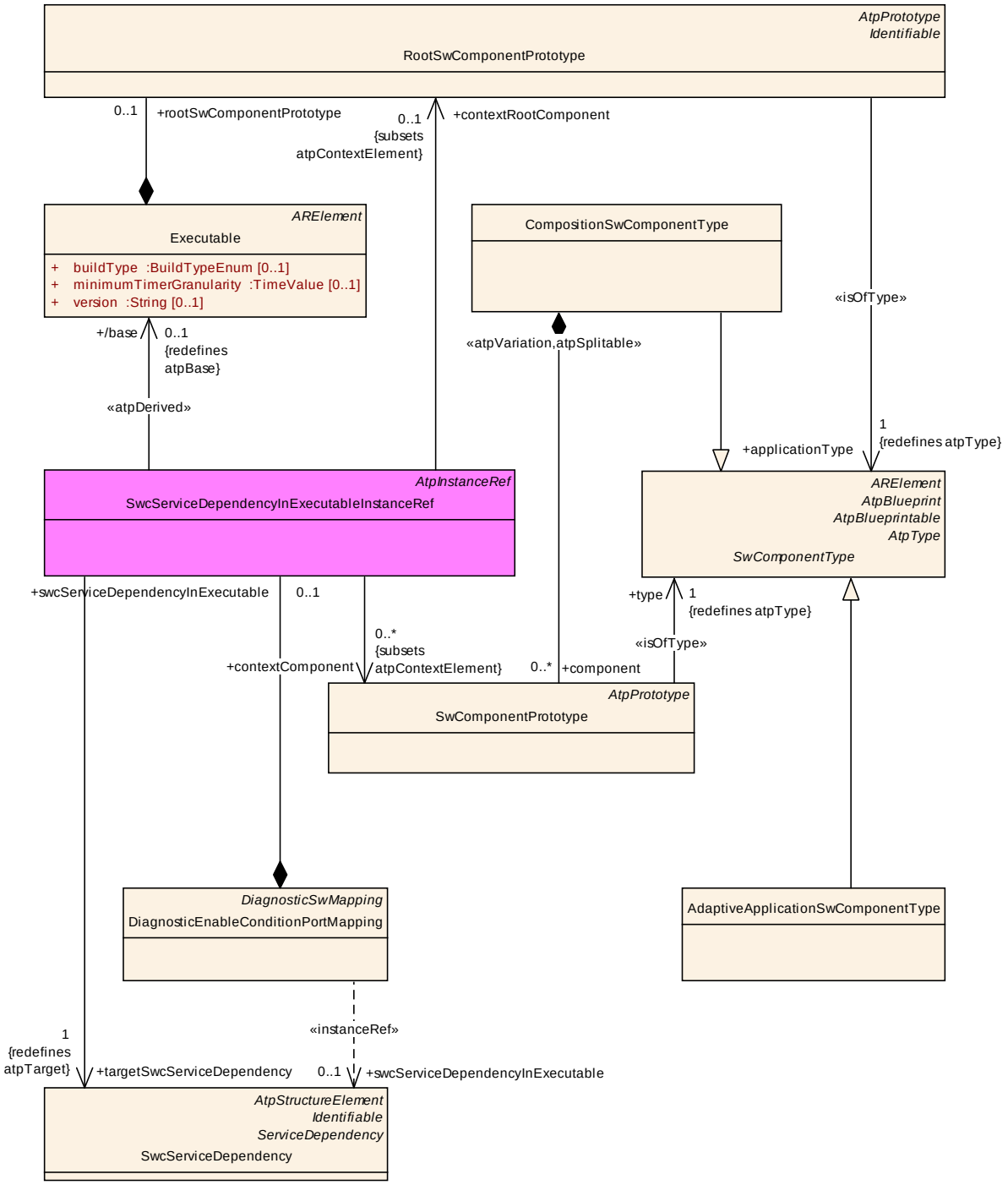

Figure B.9: Modeling of `DiagnosticEnableConditionPortMapping` via `SwcServiceDependencyInExecutableInstanceRef`

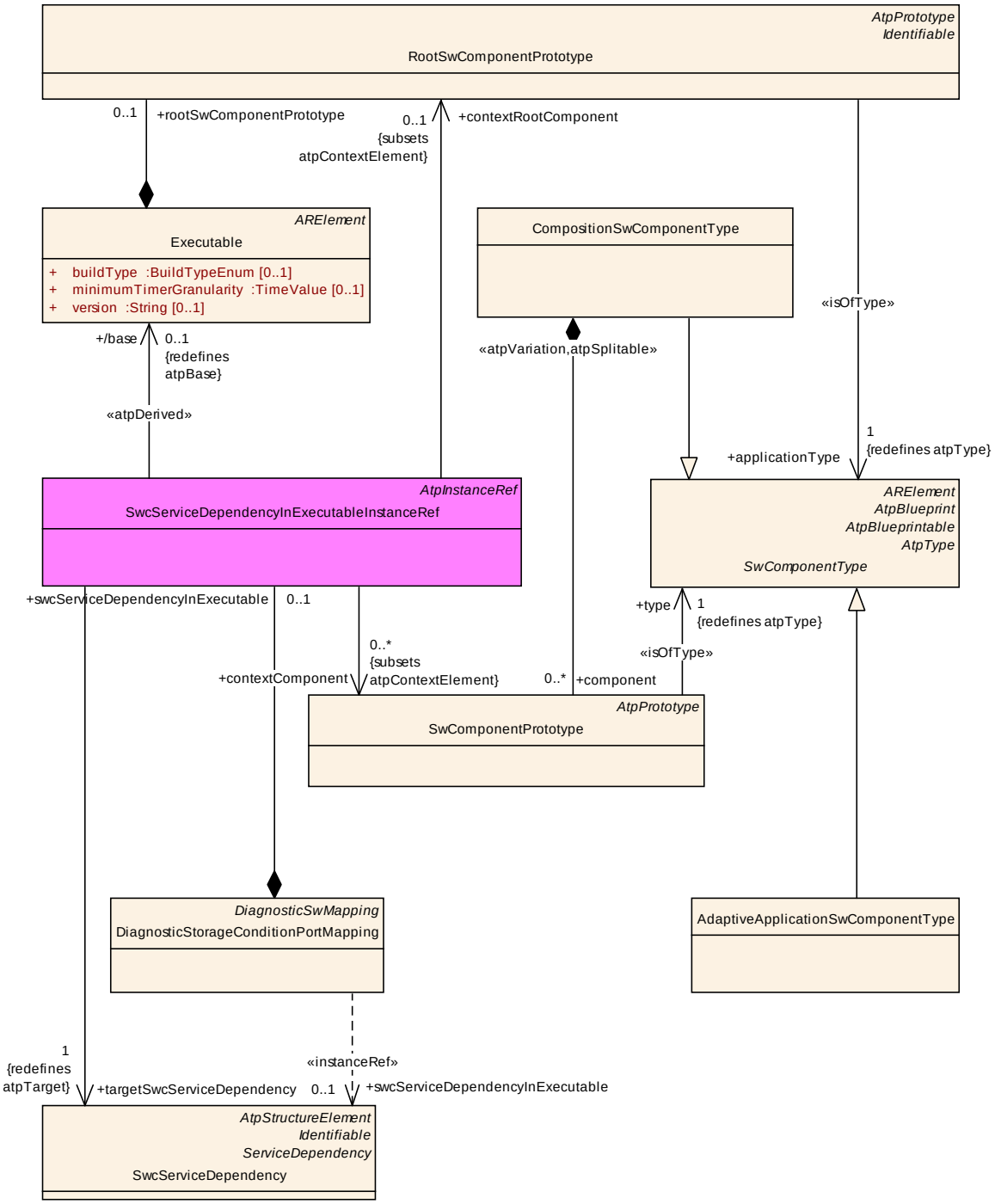

Figure B.10: Modeling of `DiagnosticStorageConditionPortMapping` via `SwcServiceDependencyInExecutableInstanceRef`

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	***Type***	***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, CollectableElement, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***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

referenceBase	ReferenceBase	*	aggr	This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortLabel xml.sequenceOffset=10

Table C.2: ARPackage

Class	**AdminData**			
Package	M2::MSR::AsamHdo::AdminData			
Note	AdminData represents the ability to express administrative information for an element. This administration information is to be treated as meta-data such as revision id or state of the file. There are basically four kinds of meta-data • The language and/or used languages. • Revision information covering e.g. revision number, state, release date, changes. Note that this information can be given in general as well as related to a particular company. • Document meta-data specific for a company			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
docRevision (ordered)	DocRevision	*	aggr	This allows to denote information about the current revision of the object. Note that information about previous revisions can also be logged here. The entries shall be sorted descendant by date in order to reflect the history. Therefore the most recent entry representing the current version is denoted first. **Tags:** xml.roleElement=true; xml.roleWrapper Element=true; xml.sequenceOffset=50; xml.type Element=false; xml.typeWrapperElement=false
language	LEnum	0..1	attr	This attribute specifies the master language of the document or the document fragment. The master language is the one in which the document is maintained and from which the other languages are derived from. In particular in case of inconsistencies, the information in the master language is priority. **Tags:** xml.sequenceOffset=20

sdg	Sdg	*	aggr	This property allows to keep special data which is not represented by the standard model. It can be utilized to keep e.g. tool specific data. **Tags:** xml.roleElement=true; xml.roleWrapper Element=true; xml.sequenceOffset=60; xml.type Element=false; xml.typeWrapperElement=false
usedLangu ages	MultiLanguageP lainText	0..1	aggr	This property specifies the languages which are provided in the document. Therefore it should only be specified in the top level admin data. For each language provided in the document there is one entry in MultilanguagePlainText. The content of each entry can be used for illustration of the language. The used language itself depends on the language attribute in the entry. **Tags:** xml.sequenceOffset=30

Table C.3: AdminData

Class	**ApplicationSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The ApplicationSwComponentType is used to represent the application software. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	ARElement, ARObject, AtomicSwComponentType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.4: ApplicationSwComponentType

Class	**ArrayValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies the values for an array.			
Base	ARObject, CompositeValueSpecification, ValueSpecification			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
element (ordered)	ValueSpecificati on	1..*	aggr	The value for a single array element. All ValueSpecifications aggregated by ArrayValueSpecification shall have the same structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table C.5: ArrayValueSpecification

Class	**AssemblySwConnector**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	AssemblySwConnectors are exclusively used to connect SwComponentPrototypes in the context of a CompositionSwComponentType.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable, SwConnector			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
provider	AbstractProvidedPortPrototype	0..1	iref	Instance of providing port.
requester	AbstractRequiredPortPrototype	0..1	iref	Instance of requiring port.

Table C.6: AssemblySwConnector

Class	**AtomicSwComponentType (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	An atomic software component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs.			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
internalBehavior	SwcInternalBehavior	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, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the AtomicSwComponentType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName

Table C.7: AtomicSwComponentType

Class	**AutosarDataPrototype (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of an AutosarDataType.			
Base	ARObject, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
type	AutosarDataType	1	tref	This represents the corresponding data type. **Stereotypes:** isOfType

Table C.8: AutosarDataPrototype

Class	**AutosarDataType (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for user defined AUTOSAR data types for ECU software.			
Base	ARElement, ARObject, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of this AutosarDataType.

Table C.9: AutosarDataType

Class	**BaseType (abstract)**			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This abstract meta-class represents the ability to specify a platform dependant base type.			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
baseTypeDefinition	BaseTypeDefinition	1	aggr	This is the actual definition of the base type. **Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false

Table C.10: BaseType

Class	**CompuMethod**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:** atp.recommendedPackage=CompuMethods			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
compuInternalToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Tags:** xml.sequenceOffset=80
compuPhysToInternal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Tags:** xml.sequenceOffset=90

displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:** xml.sequenceOffset=20
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:** xml.sequenceOffset=30

Table C.11: CompuMethod

Class	**DataPrototype (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of any data type.			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
swDataDefProps	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties which apply on data prototype level.

Table C.12: DataPrototype

Class	**DiagnosticServiceInstance (abstract)**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService			
Note	This represents a concrete instance of a diagnostic service.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
accessPermission	DiagnosticAccessPermission	0..1	ref	This represents the collection of DiagnosticAccessPermissions that allow for the execution of the referencing DiagnosticServiceInstance..
serviceClass	DiagnosticServiceClass	0..1	ref	This represents the corresponding "class", i.e. this meta-class provides properties that are shared among all instances of applicable sub-classes of DiagnosticServiceInstance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference. **Stereotypes:** atpAbstract

Table C.13: DiagnosticServiceInstance

Class	**EthernetPhysicalChannel**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	The EthernetPhysicalChannel represents a VLAN or an untagged channel. An untagged channel is modeled as an EthernetPhysicalChannel without an aggregated VLAN.			
Base	ARObject, Identifiable, MultilanguageReferrable, PhysicalChannel, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
networkEndpoint	NetworkEndpoint	*	aggr	Collection of NetworkEndpoints that are used in the VLan. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName
soAdConfig	SoAdConfig	0..1	aggr	SoAd Configuration for one specific Physical Channel.
vlan	VlanConfig	0..1	aggr	VLAN Configuration.

Table C.14: EthernetPhysicalChannel

Class	**ISignal**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Signal of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal is sent in different SignalIPdus to multiple receivers. To support the RTE "signal fan-out" each SignalIPdu contains ISignals. If the same System Signal is to be mapped into several SignalIPdus there is one ISignal needed for each ISignalToIPduMapping. ISignals describe the Interface between the Precompile configured RTE and the potentially Postbuild configured Com Stack (see ECUC Parameter Mapping). In case of the SystemSignalGroup an ISignal must be created for each SystemSignal contained in the SystemSignalGroup. **Tags:** atp.recommendedPackage=ISignals			
Base	ARObject, CollectableElement, FibexElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataTransformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignal. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataTransformation, variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime

dataTypePolicy	DataTypePolicyEnum	1	attr	With the aggregation of SwDataDefProps an ISignal specifies how it is represented on the network. This representation follows a particular policy. Note that this causes some redundancy which is intended and can be used to support flexible development methodology as well as subsequent integrity checks. If the policy "networkRepresentationFromComSpec" is chosen the network representation from the ComSpec that is aggregated by the PortPrototype shall be used. If the "override" policy is chosen the requirements specified in the PortInterface and in the ComSpec are not fulfilled by the networkRepresentationProps. In case the System Description doesn't use a complete Software Component Description (VFB View) the "legacy" policy can be chosen.
iSignalProps	ISignalProps	0..1	aggr	Additional optional ISignal properties that may be stored in different files. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=iSignalProps
iSignalType	ISignalTypeEnum	0..1	attr	This attribute defines whether this iSignal is an array that results in an UINT8_N / UINT8_DYN ComSignalType in the COM configuration or a primitive type.
initValue	ValueSpecification	0..1	aggr	Optional definition of a ISignal's initValue in case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. This value can be used to configure the Signal's "InitValue". If a full DataMapping exist for the SystemSignal this information may be available from a configured SenderComSpec and ReceiverComSpec. In this case the initvalues in SenderComSpec and/or ReceiverComSpec override this optional value specification. Further restrictions apply from the RTE specification.
length	Integer	1	attr	Size of the signal in bits. The size needs to be derived from the mapped VariableDataPrototype according to the mapping of primitive DataTypes to BaseTypes as used in the RTE. Indicates maximum size for dynamic length signals. The ISignal length of zero bits is allowed.

networkRepresentationProps	SwDataDefProps	0..1	aggr	Specification of the actual network representation. The usage of SwDataDefProps for this purpose is restricted to the attributes compuMethod and baseType. The optional baseType attributes "memAllignment" and "byteOrder" shall not be used. The attribute "dataTypePolicy" in the SystemTemplate element defines whether this network representation shall be ignored and the information shall be taken over from the network representation of the ComSpec. If "override" is chosen by the system integrator the network representation can violate against the requirements defined in the PortInterface and in the network representation of the ComSpec. In case that the System Description doesn't use a complete Software Component Description (VFB View) this element is used to configure "ComSignalDataInvalidValue" and the Data Semantics.
systemSignal	SystemSignal	1	ref	Reference to the System Signal that is supposed to be transmitted in the ISignal.
timeoutSubstitutionValue	ValueSpecification	0..1	aggr	Defines and enables the ComTimeoutSubstituition for this ISignal.
transformationISignalProps	TransformationISignalProps	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignal specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignals are described in the TransformationTechnology class.

Table C.15: ISignal

Class	**ISignalIPdu**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Represents the IPdus handled by Com. The ISignalIPdu assembled and disassembled in AUTOSAR COM consists of one or more signals. In case no multiplexing is performed this IPdu is routed to/from the Interface Layer. A maximum of one dynamic length signal per IPdu is allowed. **Tags:** atp.recommendedPackage=Pdus			
Base	ARObject, CollectableElement, FibexElement, IPdu, Identifiable, MultilanguageReferrable, PackageableElement, Pdu, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

iPduTiming Specification	IPduTiming	0..1	aggr	Timing specification for Com IPdus (Transmission Modes). This information is mandatory for the sender in a System Extract. This information may be omitted on receivers in a System Extract. atpVariation: The timing of a Pdu can vary. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
iSignalToPduMapping	ISignalToIPduMapping	*	aggr	Definition of SignalToIPduMappings included in the SignalIPdu. atpVariation: The content of a PDU can be variable. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
pduCounter	SignalIPduCounter	0..1	aggr	An included Pdu counter is used to ensure that a sequence of Pdus is maintained. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
pduReplication	SignalIPduReplication	0..1	aggr	Pdu Replication is a form of redundancy where the data content of one ISignalIPdu (source) is transmitted inside a set of replica ISignalIPdus. These ISignalIPdus (copies) have different Pdu IDs, identical PduCounters, identical data content and are transmitted with the same frequency. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
unusedBitPattern	Integer	1	attr	AUTOSAR COM and AUTOSAR IPDUM are filling not used areas of an IPDU with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the IPdu.

Table C.16: ISignalIPdu

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	***Type***	***Mul.***	***Kind***	***Note***

desc	MultiLanguage OverviewParagraph	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	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:** xml.sequenceOffset=-50
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Tags:** xml.sequenceOffset=-40
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** xml.sequenceOffset=-25
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:** xml.sequenceOffset=-30

uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:** xml.attribute=true

Table C.17: Identifiable

Class	**NonOsModuleInstantiation (abstract)**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::AdaptiveModuleImplementation			
Note	This meta-class defines the abstract attributes for the configuration of an adaptive autosar module other than the OS module. **Tags:** atp.Status=draft			
Base	ARObject, AdaptiveModuleInstantiation, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
process	Process	0..1	ref	Reference to the process where the AdaptiveModuleInstantiation is executed. **Tags:** atp.Status=draft

Table C.18: NonOsModuleInstantiation

Class	**PPortComSpec (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of a provided PortPrototype. This class will contain attributes that are valid for all kinds of provide ports, independent of client-server or sender-receiver communication patterns.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.19: PPortComSpec

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

Table C.20: PPortPrototype

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	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.21: PortInterface

Class	**PortInterfaceToDataTypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign			
Note	This meta-class represents the ability to associate a PortInterface with a DataTypeMappingSet. This association is needed for the generation of header files in the scope of a single PortInterface. The association is intentionally made outside the scope of the PortInterface itself because the designers of a PortInterface most likely will not want to add details about the level of ImplementationDataType. **Tags:** atp.Status=draft; atp.recommendedPackage=ServiceInterfaceToDataType Mappings			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
dataTypeMappingSet	DataTypeMappingSet	1..*	ref	This represents the reference to the applicable dataTypemappingSet **Tags:** atp.Status=draft; atp.Status Comment=Reserved for adaptive platform
portInterface	PortInterface	1	ref	This represents the reference to the applicable PortInterface **Tags:** atp.Status=draft; atp.Status Comment=Reserved for adaptive platform

Table C.22: PortInterfaceToDataTypeMapping

Class	**PortPrototype (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	ARObject, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	*Type*	*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.
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.
portPrototypeProps	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype. **Tags:** atp.Status=draft
senderReceiverAnnotation	SenderReceiverAnnotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPortAnnotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table C.23: PortPrototype

Class	**ProvidedServiceInstance**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Service instances that are provided by the ECU that is connected via the ApplicationEndpoint to a CommunicationConnector.			
Base	ARObject, AbstractServiceInstance, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
EventHandler	EventHandler	*	aggr	Collection of event callback configurations.
instanceIdentifier	PositiveInteger	0..1	attr	Instance identifier. Can be used for e.g. service discovery to identify the instance of the service.
priority	PositiveInteger	0..1	attr	Priority defined per provided ServiceInstance.
sdServerConfig	SdServerConfig	0..1	aggr	Service Discovery Server configuration.
serviceIdentifier	PositiveInteger	0..1	attr	Service ID. Shall be unique within one system to allow service discovery.

Table C.24: ProvidedServiceInstance

Class	**RPortComSpec (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of a required PortPrototype. This class will contain attributes that are valid for all kinds of require-ports, independent of client-server or sender-receiver communication patterns.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.25: RPortComSpec

Class	**RPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	ARObject, AbstractRequiredPortPrototype, AtpBlueprintable, AtpFeature, Atp Prototype, Identifiable, MultilanguageReferrable, PortPrototype, Referrable			
Attribute	***Type***	***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.26: RPortPrototype

Class	**RecordValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies the values for a record.			
Base	ARObject, CompositeValueSpecification, ValueSpecification			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
field (ordered)	ValueSpecification	1..*	aggr	The value for a single record field. This could also be mapped explicitly to a record element of the data type using the shortName of the ValueSpecification. But this would introduce a relationship to the data type that is too strong. As of now, it is only important that the structure of the data type matches the structure of the ValueSpecification indepenently of the shortNames. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table C.27: RecordValueSpecification

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	***Type***	***Mul.***	***Kind***	***Note***
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Tags:** xml.enforceMinMultiplicity=true; xml.sequenceOffset=-100

shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** xml.sequenceOffset=-90

Table C.28: Referrable

Class	**Sd**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	This class represents a primitive element in a special data group.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
gid	NameToken	1	attr	This attributes specifies an identifier. Gid comes from the SGML/XML-Term "Generic Identifier" which is the element name in XML. The role of this attribute is the same as the name of an XML - element. **Tags:** xml.attribute=true
value	VerbatimStringPlain	1	attr	This is the value of the special data. **Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.typeElement=false; xml.typeWrapperElement=false
xmlSpace	XmlSpaceEnum	0..1	attr	This attribute is used to signal an intention that in that element, white space should be preserved by applications. It is defined according to xml:space as declared by W3C. **Tags:** xml.attribute=true; xml.attributeRef=true; xml.enforceMinMultiplicity=true; xml.name=space; xml.nsPrefix=xml

Table C.29: Sd

Class	**Sdg**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	Sdg (SpecialDataGroup) is a generic model which can be used to keep arbitrary information which is not explicitly modeled in the meta-model. Sdg can have various contents as defined by sdgContentsType. Special Data should only be used moderately since all elements should be defined in the meta-model. Thereby SDG should be considered as a temporary solution when no explicit model is available. If an sdgCaption is available, it is possible to establish a reference to the sdg structure.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

gid	NameToken	1	attr	This attributes specifies an identifier. Gid comes from the SGML/XML-Term "Generic Identifier" which is the element name in XML. The role of this attribute is the same as the name of an XML - element. **Tags:** xml.attribute=true
sdgCaption	SdgCaption	0..1	aggr	This aggregation allows to assign the properties of Identifiable to the sdg. By this, a shortName etc. can be assigned to the Sdg. **Tags:** xml.sequenceOffset=20
sdgCaptionRef	SdgCaption	0..1	ref	This association allows to reuse an already existing caption. **Tags:** xml.name=SDG-CAPTION-REF; xml.sequenceOffset=25
sdgContentsType	SdgContents	0..1	aggr	This is the content of the Sdg. **Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false

Table C.30: Sdg

Class	**ServiceNeeds (abstract)**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This expresses the abstract needs that a Software Component or Basic Software Module has on the configuration of an AUTOSAR Service to which it will be connected. "Abstract needs" means that the model abstracts from the Configuration Parameters of the underlying Basic Software.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.31: ServiceNeeds

Class	**ServiceSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	ServiceSwComponentType is used for configuring services for a given ECU. Instances of this class are only to be created in ECU Configuration phase for the specific purpose of the service configuration. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	ARElement, ARObject, AtomicSwComponentType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.32: ServiceSwComponentType

Class	**SwComponentPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	Role of a software component within a composition.			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
type	SwComponentType	1	tref	Type of the instance. **Stereotypes:** isOfType

Table C.33: SwComponentPrototype

Class	**SwConnector (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	The base class for connectors between ports. Connectors have to be identifiable to allow references from the system constraint template.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
mapping	PortInterfaceMapping	0..1	ref	Reference to a PortInterfaceMapping specifying the mapping of unequal named PortInterface elements of the two different PortInterfaces typing the two PortPrototypes which are referenced by the ConnectorPrototype.

Table C.34: SwConnector

Class	≪atpVariation≫ **SwDataDefProps**			
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	This class is a collection of properties relevant for data objects under various aspects. One could consider this class as a "pattern of inheritance by aggregation". The properties can be applied to all objects of all classes in which SwDataDefProps is aggregated. Note that not all of the attributes or associated elements are useful all of the time. Hence, the process definition (e.g. expressed with an OCL or a Document Control Instance MSR-DCI) has the task of implementing limitations. SwDataDefProps covers various aspects: • Structure of the data element for calibration use cases: is it a single value, a curve, or a map, but also the recordLayouts which specify how such elements are mapped/converted to the DataTypes in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet • Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, swAddrMethod, swPointerTagetProps, baseType, implementationDataType and additionalNativeTypeQualifier • Access policy for the MCD system, mainly expressed by swCalibrationAccess • Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalidValue • Code generation policy provided by swRecordLayout **Tags:** vh.latestBindingTime=codeGenerationTime			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
additionalNativeTypeQualifier	NativeDeclarationString	0..1	attr	This attribute is used to declare native qualifiers of the programming language which can neither be deduced from the baseType (e.g. because the data object describes a pointer) nor from other more abstract attributes. Examples are qualifiers like "volatile", "strict" or "enum" of the C-language. All such declarations have to be put into one string. **Tags:** xml.sequenceOffset=235
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. **Tags:** xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. **Tags:** xml.sequenceOffset=50

compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. **Tags:** xml.sequenceOffset=180
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. **Tags:** xml.sequenceOffset=190
displayFormat	DisplayFormatString	0..1	attr	This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system. **Tags:** xml.sequenceOffset=210
implementationDataType	ImplementationDataType	0..1	ref	This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially • redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype • the target type of a pointer (see SwPointerTargetProps), if it does not refer to a base type directly • the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly • the data type of an SwServiceArg, if it does not refer to a base type directly **Tags:** xml.sequenceOffset=215
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. **Tags:** xml.sequenceOffset=255
stepSize	Float	0..1	attr	This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating.
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself. **Tags:** xml.sequenceOffset=30

swAlignment	AlignmentType	0..1	attr	The attribute describes the intended alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memoryAllocationKeywordPolicy of the referenced SwAddrMethod. **Tags:** xml.sequenceOffset=33
swBitRepresentation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. **Tags:** xml.sequenceOffset=60
swCalibrationAccess	SwCalibrationAccessEnum	0..1	attr	Specifies the read or write access by MCD tools for this data object. **Tags:** xml.sequenceOffset=70
swCalprmAxisSet	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. **Tags:** xml.sequenceOffset=90
swComparisonVariable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. **Tags:** xml.sequenceOffset=170; xml.typeElement=false
swDataDependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). **Tags:** xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. **Tags:** xml.sequenceOffset=220; xml.typeElement=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. **Tags:** xml.sequenceOffset=230

swIntendedResolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". **Tags:** xml.sequenceOffset=240
swInterpolationMethod	Identifier	0..1	attr	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. **Tags:** xml.sequenceOffset=250
swIsVirtual	Boolean	0..1	attr	This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency . **Tags:** xml.sequenceOffset=260
swPointerTargetProps	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. **Tags:** xml.sequenceOffset=280
swRecordLayout	SwRecordLayout	0..1	ref	Record layout for this data object. **Tags:** xml.sequenceOffset=290
swRefreshTiming	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system. So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing. **Tags:** xml.sequenceOffset=300

swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. **Tags:** xml.sequenceOffset=120
swValueBlockSize	Numerical	0..1	attr	This represents the size of a Value Block **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. **Tags:** xml.sequenceOffset=350
valueAxisDataType	ApplicationPrimitiveDataType	0..1	ref	The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType. **Tags:** xml.sequenceOffset=355

Table C.35: SwDataDefProps

Class	**SwPointerTargetProps**			
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	This element defines, that the data object (which is specified by the aggregating element) contains a reference to another data object or to a function in the CPU code. This corresponds to a pointer in the C-language. The attributes of this element describe the category and the detailed properties of the target which is either a data description or a function signature.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
functionPointerSignature	BswModuleEntry	0..1	ref	The referenced BswModuleEntry serves as the signature of a function pointer definition. Primary use case: function pointer passed as argument to other function. **Tags:** xml.sequenceOffset=40
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of the target data type. **Tags:** xml.sequenceOffset=30

targetCategory	Identifier	0..1	attr	This specifies the category of the target: • In case of a data pointer, it shall specify the category of the referenced data. • In case of a function pointer, it could be used to denote the category of the referenced BswModuleEntry. Since currently no categories for BswModuleEntry are defined it will be empty. **Tags:** xml.sequenceOffset=5

Table C.36: SwPointerTargetProps

Class	**SwRecordLayout**			
Package	M2::MSR::DataDictionary::RecordLayout			
Note	Defines how the data objects (variables, calibration parameters etc.) are to be stored in the ECU memory. As an example, this definition specifies the sequence of axis points in the ECU memory. Iterations through axis values are stored within the sub-elements swRecordLayoutGroup. **Tags:** atp.recommendedPackage=SwRecordLayouts			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
swRecordLayoutGroup	SwRecordLayoutGroup	1	aggr	This is the top level record layout group. **Tags:** xml.roleElement=true; xml.roleWrapperElement=false; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false

Table C.37: SwRecordLayout

Class	**System**			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The top level element of the System Description. The System description defines five major elements: Topology, Software, Communication, Mapping and Mapping Constraints. The System element directly aggregates the elements describing the Software, Mapping and Mapping Constraints; it contains a reference to an ASAM FIBEX description specifying Communication and Topology. **Tags:** atp.recommendedPackage=Systems			
Base	ARElement, ARObject, AtpClassifier, AtpFeature, AtpStructureElement, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
clientIdDefinitionSet	ClientIdDefinitionSet	*	ref	Set of Client Identifiers that are used for inter-ECU client-server communication in the System.

containerIPduHeaderByteOrder	ByteOrderEnum	0..1	attr	Defines the byteOrder of the header in ContainerIPdus.
ecuExtractVersion	RevisionLabelString	0..1	attr	Version number of the Ecu Extract.
fibexElement	FibexElement	*	ref	Reference to ASAM FIBEX elements specifying Communication and Topology. All Fibex Elements used within a System Description shall be referenced from the System Element. atpVariation: In order to describe a product-line, all FibexElements can be optional. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
j1939SharedAddressCluster	J1939SharedAddressCluster	*	aggr	Collection of J1939Clusters that share a common address space for the routing of messages. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects (mapping of SW components to ECUs, mapping of data elements to signals, and mapping constraints). In order to support OEM / Tier 1 interaction and shared development for one common System this aggregation is atpSplitable and atpVariation. The content of SystemMapping can be provided by several parties using different names for the SystemMapping. This element is not required when the System description is used for a network-only use-case. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
pncVectorLength	PositiveInteger	0..1	attr	Length of the partial networking request release information vector (in bytes).
pncVectorOffset	PositiveInteger	0..1	attr	Absolute offset (with respect to the NM-PDU) of the partial networking request release information vector that is defined in bytes as an index starting with 0.

rootSoftwareComposition	RootSwCompositionPrototype	0..1	aggr	Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case. atpVariation: The RootSwCompositionPrototype can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
systemDocumentation	Chapter	*	aggr	Possibility to provide additional documentation while defining the System. The System documentation can be composed of several chapters. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=-10
systemVersion	RevisionLabelString	1	attr	Version number of the System Description.

Table C.38: System

Class	**SystemSignal**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The system signal represents the communication system's view of data exchanged between SW components which reside on different ECUs. The system signals allow to represent this communication in a flattened structure, with exactly one system signal defined for each data element prototype sent and received by connected SW component instances. **Tags:** atp.recommendedPackage=SystemSignals			
Base	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dynamicLength	Boolean	1	attr	The length of dynamic length signals is variable in run-time. Only a maximum length of such a signal is specified in the configuration (attribute length in ISignal element).
physicalProps	SwDataDefProps	0..1	aggr	Specification of the physical representation.

Table C.39: SystemSignal

Class	**TransformationProps (abstract)**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	This meta-class represents a abstract base class for transformation settings.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table C.40: TransformationProps

Enumeration	**TransportLayerProtocolEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstance
Note	This enumeration allows to choose a TCP/IP transport layer protocol. **Tags:** atp.Status=draft
Literal	***Description***
tcp	Transmission control protocol **Tags:** atp.EnumerationValue=1
udp	User datagram protocol **Tags:** atp.EnumerationValue=0

Table C.41: TransportLayerProtocolEnum

Class	**ValueSpecification (abstract)**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Base class for expressions leading to a value which can be used to initialize a data object.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
shortLabel	Identifier	0..1	attr	This can be used to identify particular value specifications for human readers, for example elements of a record type.

Table C.42: ValueSpecification

D History of Constraints and Specification Items

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

D.1 Constraint History of this Document according to the original version of the Document

D.1.1 Created Constraints

Number	Heading
[constr_1473]	No support for PRPortPrototype
[constr_1474]	SwDataDefProps applicable to ImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1475]	ImplementationDataType of category STRING is limited
[constr_1476]	ImplementationDataType of category VECTOR is limited
[constr_1477]	ImplementationDataType of category ASSOCIATIVE_MAP is limited
[constr_1478]	SwDataDefProps applicable to ApplicationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1479]	No support for certain values of ImplementationDataType.category
[constr_1480]	Mutual existence of CompositionDataPrototypeRef.elementInImplDatatype vs. attributes of CompositionDataPrototypeRef.dataPrototype
[constr_1481]	Usage of CompositionDataPrototypeRef in the *AUTOSAR adaptive platform*
[constr_1482]	Mapping of service interfaces vs. mapping of service interface elements
[constr_1483]	Applicability of a ServiceInterface
[constr_1484]	Applicability of ModeDependentStartupConfig.executionDependency
[constr_1485]	No subElement for ImplementationDataType of category STRING
[constr_1486]	ImplementationDataType of category STRING and SwBaseType
[constr_1487]	Number of subElements of an ImplementationDataType of category ASSOCIATIVE_MAP
[constr_1488]	Initialization of a DataPrototype typed by an ApplicationAssocMapDataType
[constr_1489]	Uniqueness of ApplicationAssocMapValueSpecification.mapElementTuple.key
[constr_1490]	Allowed value of category for reference AdaptiveModuleInstantiation.process.executable
[constr_1491]	Reference to ApplicationError
[constr_1492]	SwComponentType referenced as Executable.rootSwComponentPrototype.applicationType
[constr_1493]	ArgumentDataPrototype referenced in the role ApplicationError.errorContext
[constr_1494]	Initial value for event
[constr_1495]	Initial value for field
[constr_1496]	DiagnosticServiceDataMapping.mappedApDataElement shall only refer to specific sub-classes of DataPrototype
[constr_1497]	Attribute optionKind set to commandLineSimpleForm
[constr_1498]	Attribute optionKind set to commandLineShortForm or commandLineLongForm
[constr_1499]	Target SwcServiceDependency of DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable
[constr_1500]	Target SwcServiceDependency of DiagnosticEventPortMapping.swcServiceDependencyInExecutable

Number	Heading
[constr_1501]	Target SwcServiceDependency of DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable
[constr_1502]	Target SwcServiceDependency of DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable
[constr_1503]	Target SwcServiceDependency of DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable
[constr_1504]	Number of Process.modeDependentStartupConfig that refer to the same ModeDeclaration
[constr_1505]	Number of Process.modeDependentStartupConfig that do not refer to a ModeDeclaration
[constr_1507]	PortInterfaceToDatatypeMapping is only applicable to ServiceInterface
[constr_1508]	BaseTypeDirectDefinition.nativeDeclaration shall not be set to the value enum
[constr_3320]	Aggregation of CommunicationConnector by Machine
[constr_3287]	Mandatory information of a ProvidedSomeipServiceInstance
[constr_3288]	IP configuration restriction for unicastNetworkEndpoints
[constr_3290]	Usage of ServiceInstancePortConfig defined for a ProvidedSomeipServiceInstance
[constr_3291]	SomeipServiceInstanceToMachineMapping.portConfig aggregation restriction
[constr_3293]	Mandatory information of a RequiredSomeipServiceInstance
[constr_3296]	Usage of ServiceInstancePortConfig defined for a RequiredSomeipServiceInstance
[constr_3297]	SomeipServiceInstanceToMachineMapping only supports a single Address Family
[constr_3300]	Allowed ServiceMethodDeployment.method references
[constr_3301]	Allowed ServiceEventDeployment.event references
[constr_3302]	Allowed ServiceFieldDeployment.field references
[constr_3303]	ANY not allowed for SomeipServiceInterface.serviceInterfaceVersion
[constr_3304]	Value of attribute SomeipEventGroup.eventGroupId shall be unique
[constr_3305]	Value of attribute SomeipEvent.eventId shall be unique
[constr_3306]	Value of attribute SomeipMethod.methodId shall be unique
[constr_3307]	SomeipEvent.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3308]	SomeipEvent.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3309]	SomeipMethod.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3310]	SomeipMethod.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3320]	Aggregation of CommunicationConnector by Machine
[constr_3349]	Usage of ApplicationAssocMapDataType is limited
[constr_3350]	Consistent value of category for AdaptiveAutosarApplications referencing an Executable
[constr_3351]	SOME/IP segmentation allowed for udp SomeipEvents
[constr_3352]	SOME/IP segmentation allowed for udp SomeipMethods

Number	Heading
[constr_3353]	Restriction in usage of ApSomeipTransformationProps.sizeOfArrayLengthField
[constr_3354]	Restriction in usage of ApSomeipTransformationProps.sizeOfStructLengthField
[constr_3355]	Restriction in usage of ApSomeipTransformationProps.sizeOfUnionLengthField
[constr_3356]	Restriction in usage of ApSomeipTransformationProps.alignment
[constr_3357]	Restriction in usage of ApSomeipTransformationProps.sizeOfUnionTypeSelectorField
[constr_3358]	Usage of PortPrototype and TransportLayerIndependentInstanceId to define the same Service Instance is not allowed.
[constr_3359]	RPortPrototypeProps are related only to RPortPrototypes.
[constr_3360]	RPortPrototypeProps are related only to TransportLayerIndependentInstanceIds representing a consumer Service Instance.
[constr_3361]	Selective definition of serialization settings.
[constr_3362]	SomeipEvents aggregated by a SomeipField
[constr_3363]	SomeipMethods aggregated by a SomeipField

Table D.1: Added Constraints in original version

D.1.2 Created Specification Items

Number	Heading
[TPS_MANI_01000]	Definition of the term Manifest
[TPS_MANI_01001]	Meaning of ServiceInterface
[TPS_MANI_01002]	Semantics of a ServiceInterfaceMapping
[TPS_MANI_01003]	Limitations of the applicability of ServiceInterfaceMapping
[TPS_MANI_01004]	Semantics of ServiceInterface.namespace
[TPS_MANI_01005]	The definition of the namespace of a ServiceInterface may follow a hierarchical pattern
[TPS_MANI_01006]	Ordered definition of ServiceInterface.namespace
[TPS_MANI_01007]	Service-oriented **communication** and service **discovery**
[TPS_MANI_01008]	Semantics of AdaptiveAutosarApplication
[TPS_MANI_01009]	Standardized values of AdaptiveAutosarApplication.category
[TPS_MANI_01010]	Root element for a hierarchical software-component
[TPS_MANI_01011]	Connection between application design and application deployment
[TPS_MANI_01012]	Formal modeling of application startup behavior
[TPS_MANI_01013]	Semantics of meta-class ModeDependentStartupConfig
[TPS_MANI_01014]	Semantics of meta-class StartupConfigSet
[TPS_MANI_01015]	Semantics of meta-class StartupOption
[TPS_MANI_01016]	Category of ApplicationAssocMapDataType
[TPS_MANI_01017]	Relation of startup configuration to resource groups
[TPS_MANI_01018]	ImplementationDataType of category VECTOR

Number	Heading
[TPS_MANI_01019]	`Manifest` content may apply to different aspects of the *AUTOSAR adaptive platform*
[TPS_MANI_01020]	Serialization format of the *Manifest* in AUTOSAR
[TPS_MANI_01021]	Serialization format of *Manifest* content on a machine
[TPS_MANI_01022]	Concept behind `ServiceInterfaceMapping`
[TPS_MANI_01024]	Semantics of `ServiceInterfaceEventMapping`
[TPS_MANI_01025]	Semantics of `ServiceInterfaceFieldMapping`
[TPS_MANI_01026]	Semantics of `ServiceInterfaceMethodMapping`
[TPS_MANI_01027]	Semantics of `ApplicationAssocMapDataType`
[TPS_MANI_01028]	`ImplementationDataType` of `category` ASSOCIATIVE_MAP
[TPS_MANI_01029]	Usage of `ImplementationDataType`
[TPS_MANI_01030]	`ImplementationDataType` of `category` STRING
[TPS_MANI_01031]	Semantics of `CompositionDataPrototypeRef`
[TPS_MANI_01032]	Usage of `ServiceInterfaceMapping`
[TPS_MANI_01033]	Semantics of `ServiceInterface.event`
[TPS_MANI_01034]	Semantics of `ServiceInterface.field`
[TPS_MANI_01035]	Semantics of `ServiceInterface.method`
[TPS_MANI_01037]	Diagnostic data mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01038]	Diagnostic software mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01039]	Representation of provided service
[TPS_MANI_01040]	Representation of required service
[TPS_MANI_01041]	Startup configuration supports the definition of a launch dependency
[TPS_MANI_01042]	Definition of a linear `ImplementationDataType` of `category` VECTOR
[TPS_MANI_01043]	Definition of a rectangular `ImplementationDataType` of `category` VECTOR
[TPS_MANI_01044]	Structure of an `ImplementationDataType` of `category` ASSOCIATIVE_MAP
[TPS_MANI_01045]	`Process.modeDependentStartupConfig` that does not refer to a `ModeDeclaration`
[TPS_MANI_01046]	Semantics of `ModeDependentStartupConfig.machineMode`
[TPS_MANI_01047]	Existence of `SwRecordLayout` for an `ApplicationPrimitiveDataType` of `category` STRING
[TPS_MANI_01048]	Mapping of `DiagnosticEvent` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01049]	Mapping of `DiagnosticOperationCycle` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01050]	Mapping of `DiagnosticEnableCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01051]	Mapping of `DiagnosticStorageCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01052]	Semantics of `RPortPrototypeProps.portInstantiationBehavior`
[TPS_MANI_01053]	Usage of `ComSpec`s on the *AUTOSAR adaptive platform*
[TPS_MANI_01054]	Definition of the queue length of an `event`

Number	Heading
[TPS_MANI_01055]	Semantics of ServiceInterface.possibleError
[TPS_MANI_01056]	Semantics of ApplicationError.errorContext
[TPS_MANI_01057]	Semantics of RPortPrototypeProps.searchBehavior
[TPS_MANI_01058]	Ability to create a mapping of ApplicationErrors aggregated in the role possibleError
[TPS_MANI_01059]	Different values of optionKind within a StartupConfig.startupOption
[TPS_MANI_01060]	Use cases for the application of DiagnosticServiceDataMapping
[TPS_MANI_01061]	Requirements on scheduling
[TPS_MANI_01062]	ImplementationDataType to generate a C++ enum
[TPS_MANI_01063]	Sharing of ImplementationDataType with enumeration semantics
[TPS_MANI_03000]	Mapping of AdaptivePlatformServiceInstance to PortPrototypes
[TPS_MANI_03001]	Mapping of AdaptivePlatformServiceInstance to a Machine
[TPS_MANI_03002]	IP configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03003]	ProvidedSomeipServiceInstance Fanout
[TPS_MANI_03004]	IPv4 Multicast event destination address
[TPS_MANI_03005]	IPv4 Multicast address range
[TPS_MANI_03006]	IPv6 Multicast address range
[TPS_MANI_03007]	Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03008]	Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03009]	Tcp and Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03011]	Server Timing configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03012]	Initial Wait Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03013]	Repetition Wait Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03014]	Main Phase configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03015]	TTL for Offer Service Entries
[TPS_MANI_03016]	Servers RequestResponseDelay for received FindService entries
[TPS_MANI_03017]	Server Capability Records
[TPS_MANI_03018]	Usage of SomeipProvidedEventGroup.multicastThreshold
[TPS_MANI_03019]	TTL for SubscribeEventGroupAck Entries
[TPS_MANI_03020]	Servers RequestResponseDelay for received SubscribeEventGroup entries
[TPS_MANI_03021]	Requirements on the service version from the client's point of view
[TPS_MANI_03022]	Context of RequiredSomeipServiceInstance
[TPS_MANI_03023]	Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03024]	Tcp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03025]	Client Timing configuration for a RequiredSomeipServiceInstance

Number	Heading
[TPS_MANI_03026]	Initial Wait Phase configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03027]	Repetition Wait Phase configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03028]	TTL for Find Service Entries
[TPS_MANI_03029]	Client Capability Records
[TPS_MANI_03030]	SomeipSdClientEventGroupTimingConfig.timeToLive for SubscribeEventGroup Entries
[TPS_MANI_03031]	Clients RequestResponseDelay for received ServiceOffer entries
[TPS_MANI_03032]	Description of middleware technologies not standardized by AUTOSAR
[TPS_MANI_03035]	Content of the Machine configuration
[TPS_MANI_03036]	ServiceInterface deployment to a middleware transport layer
[TPS_MANI_03037]	Purpose of ServiceMethodDeployment
[TPS_MANI_03038]	Purpose of ServiceEventDeployment
[TPS_MANI_03039]	Purpose of ServiceFieldDeployment
[TPS_MANI_03040]	SOME/IP ServiceInterface binding
[TPS_MANI_03041]	Definition of SOME/IP EventGroups
[TPS_MANI_03042]	Definition of SOME/IP Service Version
[TPS_MANI_03043]	SOME/IP VariableDataPrototype binding
[TPS_MANI_03044]	SOME/IP ClientServerOperation binding
[TPS_MANI_03045]	UserDefined ServiceInterface binding
[TPS_MANI_03046]	User defined VariableDataPrototype binding
[TPS_MANI_03047]	User defined ClientServerOperation binding
[TPS_MANI_03048]	User defined Field binding
[TPS_MANI_03049]	Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03050]	Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance
[TPS_MANI_03051]	Usage of SomeipMethod.transportProtocol
[TPS_MANI_03052]	Static IPv4 configuration
[TPS_MANI_03053]	Static IPv6 configuration
[TPS_MANI_03056]	Usage of SomeipEvent.transportProtocol
[TPS_MANI_03057]	SOME/IP Field binding
[TPS_MANI_03059]	RequiredSomeipServiceInstance.requiredServiceInstanceId
[TPS_MANI_03061]	IPv6 Multicast event destination address
[TPS_MANI_03064]	SOME/IP Service Discovery message exchange configuration
[TPS_MANI_03065]	Hardware resources of the machine
[TPS_MANI_03066]	Description of machine states
[TPS_MANI_03067]	SOME/IP segmentation of udp SomeipEvents
[TPS_MANI_03068]	SOME/IP segmentation of SomeipMethod Calls
[TPS_MANI_03069]	SOME/IP segmentation of SomeipMethod Responses
[TPS_MANI_03070]	Size of a length field for a chosen array
[TPS_MANI_03071]	Size of a length field for a chosen structure

Number	Heading
[TPS_MANI_03072]	Size of a length field for a chosen union
[TPS_MANI_03073]	Alignment of a dynamic DataPrototype
[TPS_MANI_03074]	Size of a type selector field for a chosen union
[TPS_MANI_03075]	Byte Order of chosen DataPrototype in the serialized data stream
[TPS_MANI_03094]	Machine-specific platform configuration settings
[TPS_MANI_03095]	Implementation-specific platform configuration settings
[TPS_MANI_03096]	Machine-specific configuration settings for a generic module
[TPS_MANI_03097]	Implementation-specific configuration settings for a generic module
[TPS_MANI_03098]	Machine-specific configuration settings for the OS module
[TPS_MANI_03099]	Implementation-specific configuration settings for the OS module
[TPS_MANI_03100]	Transport layer independent TransportLayerIndependentInstanceIds
[TPS_MANI_03101]	SOME/IP serialization
[TPS_MANI_03102]	UserDefined serialization
[TPS_MANI_03103]	Default size for all array length fields
[TPS_MANI_03104]	Default size for all structure length fields
[TPS_MANI_03105]	Default size for all union length fields
[TPS_MANI_03106]	Default size for all union type selector fields
[TPS_MANI_03107]	Default alignment for all dynamic DataPrototypes
[TPS_MANI_03108]	Default Byte Order for all DataPrototypes
[TPS_MANI_03109]	TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface

Table D.2: Added Specification Items in original Version

E Splitable Elements in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpSplitable≫ in the scope of this document.

Each entry in the table consists of the identification of the specific model element itself and the applicable value of the tagged value atp.Splitkey.

For more information about the concept of splitable model elements and how these shall be treated please refer to [5].

Name of splitable element	Splitkey
ServiceInterface.namespace	shortName

Table E.1: Usage of splitable elements

F Variation Points in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpVariation≫ in the scope of this document.

Each entry in the table consists of the identification of the model element itself and the applicable value of the tagged value vh.latestBindingTime.

For more information about the concept of variation points and how model elements that contain variation points shall be treated please refer to [5].

Variation Point	Latest Binding Time
Machine.machineModeMachine	preCompileTime
ServiceInterface.event	blueprintDerivationTime
ServiceInterface.field	blueprintDerivationTime
ServiceInterface.method	blueprintDerivationTime

Table F.1: Usage of variation points