

<b>Document Title</b>	Modeling Show Cases Report
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	789

<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	R19-11

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Description</b>
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Add Show case Structured Requirement</li> <li>• Changed Document Status from Final to published</li> </ul>
2018-10-31	4.4.0	AUTOSAR Release Management	Editorial Changes
2017-12-08	4.3.1	AUTOSAR Release Management	Editorial Changes
2016-11-30	4.3.0	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

1	Introduction	6
2	Overview	7
3	Measurement and Calibration	8
3.1	Introductory Show Case	9
3.1.1	Physical System	9
3.1.1.1	Components Overview	10
3.1.1.2	The Environment	11
3.1.1.3	The Plant	12
3.1.1.4	The Controller	13
3.1.2	AUTOSAR Modeling	15
3.1.3	RTE Generation, Measurement and Calibration	16
3.1.3.1	FlatMap	17
3.1.3.2	ECU Documentation, Measurement and Calibration	17
3.1.4	A2L File	17
3.1.5	Implementation in C	19
3.1.6	A walk with T_Plant through the Show Case	22
3.1.6.1	Physical System	22
3.1.6.2	AUTOSAR Modeling	23
3.1.6.3	System	35
3.1.6.4	ECU Configuration	36
3.1.6.5	RTE Generation	38
3.1.6.6	Implementation in C	41
3.1.6.7	A2L File	42
3.1.6.8	Measurement and Calibration Tool	43
3.1.7	Show cases in the Example	44
3.1.7.1	CompositionSwComponentTypes	44
3.1.7.2	ParameterSwComponentTypes	45
3.1.7.3	ApplicationSwComponentTypes	46
3.1.7.4	ParameterInterfaces	53
3.1.7.5	SenderReceiverInterfaces	56
3.1.7.6	ApplicationDataTypes, Category VALUE	57
3.1.7.7	Units	61
3.1.7.8	PhysicalDimensions	64
3.1.7.9	SwAddrMethods	67
3.2	Advanced Show Case	68
3.2.1	General Objectives of the Model Structure	68
3.2.1.1	The Ecu Description	68
3.2.1.2	The Ecu Extract	68
3.2.1.3	Data Types and Data Objects	69
3.2.1.4	Axis, Curves and Maps	70
3.2.1.5	Axis, Curves and Maps on ApplicationDataType level	70

3.2.1.6	Axis, Curves and Maps on DataPrototype and SwComponentPrototype level . . . . .	72
3.2.1.7	Arrays of Maps and Axes . . . . .	77
3.2.1.8	Measurement of Modes . . . . .	78
3.2.2	Show cases in the Example . . . . .	80
3.2.2.1	CompositionSwComponentTypes . . . . .	80
3.2.2.2	ParameterSwComponentTypes . . . . .	87
3.2.2.3	ApplicationSwComponentTypes . . . . .	89
3.2.2.4	ParameterInterfaces . . . . .	93
3.2.2.5	ModeSwitchInterfaces . . . . .	96
3.2.2.6	SenderReceiverInterfaces . . . . .	97
3.2.2.7	ApplicationDataTypes, Category BOOLEAN . . . . .	100
3.2.2.8	ApplicationDataTypes, Category VALUE . . . . .	101
3.2.2.9	ApplicationDataTypes, Category COM_AXIS . . . . .	105
3.2.2.10	ApplicationDataTypes, Category CURVE . . . . .	107
3.2.2.11	ApplicationDataTypes, Category MAP . . . . .	108
3.2.2.12	ApplicationArrayDataTypes . . . . .	109
3.2.2.13	ApplicationRecordDataTypes . . . . .	111
3.2.2.14	ModeDeclarationGroups . . . . .	112
3.2.2.15	Units . . . . .	113
3.2.2.16	PhysicalDimensions . . . . .	115
3.2.2.17	SwAddrMethods . . . . .	117
4	Structured Requirements . . . . .	119
4.1	Introductory Show Case . . . . .	119
4.1.1	Additional information (M1 parameter) . . . . .	119
4.1.2	Specification items as requirements . . . . .	122
4.1.3	Diagnostic requirements . . . . .	123
4.1.4	Decomposition of requirements . . . . .	124
A	Mentioned Class Tables . . . . .	127

## References

- [1] Methodology  
AUTOSAR\_TR\_Methodology
- [2] Modeling Show Cases Examples  
AUTOSAR\_EXP\_ModelingShowCases
- [3] Software Component Template  
AUTOSAR\_TPS\_SoftwareComponentTemplate
- [4] Standardization Template  
AUTOSAR\_TPS\_StandardizationTemplate
- [5] Specification of Platform Types  
AUTOSAR\_SWS\_PlatformTypes
- [6] System Template  
AUTOSAR\_TPS\_SystemTemplate

# 1 Introduction

The objective of this report is the illustration and execution of AUTOSAR modeling and the AUTOSAR methodology (see [1]) for selected show cases.

Each show case focuses on a few specific topics and gives an overview of their basic usage and their application in the field. Where appropriate, the show cases are based on real world applications of the AUTOSAR standard.

It contains

- explanatory background on the functional use case for which the specific part of the AUTOSAR modeling is applied.
- illustration of the AUTOSAR model content in form of interlinked tables
- explanation of the processing results of these AUTOSAR models (e.g. C code, A2L files, ...)
- snippets of the full-blown examples. The complete examples are provided in the archive `AUTOSAR_EXP_ModelingShowCases.zip` [2].

## 2 Overview

The report is organized in chapters according to the main focus of the contained show cases. Each chapter contains a topic specific overview and at least one show case. Each chapter is self contained and understandable without reading any other chapter.

The technical report on the AUTOSAR Methodology [1] deserves a special mentioning as accompanying document for going through the show cases.

In the first version of the technical report, the show cases are targeting the topic of measurement and calibration, involving the creation of A2L files based on AUTOSAR models. For these show cases, also the specification of the `SoftwareComponent-Template` [3] is a good accompanying document.

This updated version is extended by an additional show case targeting the topic of Structured Requirements. For these show cases, also the specification of the `StandardizationTemplate` [4] is a good accompanying document.

### 3 Measurement and Calibration

Measurements and Calibration (short: MC) is a major step in the development of electronic control units (ECUs). Measurement and Calibration systems (MC systems), involving software tools (MC tools) as well as the hardware to access an ECU (not in focus here), enable the developer to measure variables and to adapt calibration parameters (or "characteristics") during the run-time of the ECU.

For instance, the following tasks are regularly done by "Measurement and Calibration"

- Adaptation to real hardware (e.g. inserting the electrical characteristics of a sensor)
- Calibration of controllers (e.g. adjusting the parameters of a closed loop controller)
- Tuning of ECU internal environment models (e.g. for "virtual sensors")
- Validation of ECU functions
- Tracking of development errors
- Collecting data for automated optimization of parameters

The "Introductory Show Case" (see 3.1), illustrates all basic artifacts on the way from a physical system that is to be controlled by an ECU until measuring and calibrating with a MC system.

As didactic simplification only a few data types were used, e.g. neither `CURVES`<sup>1</sup> nor `MAPs`<sup>2</sup> were chosen, nor any `ApplicationCompositeDataType`.

However, those advanced topics, their modeling in AUTOSAR as well as their transfer to a MC tool, is of particular interest: they are regularly needed and used the field. Therefore, the "Advanced Show Case" in chapter 3.2 especially highlights these topics. This show case is directly derived from the real world modeling and structuring approach of a major Tier 1 in the powertrain domain. So it also illustrates "good practices" in the field for designing AUTOSAR systems which are to be measured and calibrated later on in their development.

---

<sup>1</sup>`CURVES` are two dimensional functions defined via axis points and the corresponding function values. Interpolation or extrapolation is used to calculate function values that are not directly defined.

<sup>2</sup>`MAPs` are similar to `CURVES` but three dimensional



### 3.1 Introductory Show Case

As introduction to measurement and calibration with AUTOSAR a simple, artificial closed-loop control system was chosen. This allows interesting feedback of the system when using a MC tool. At the same time, the model, the source code and generated files are still comprehensible.

A drawback is, that not all typical "real world" data types are featured, for instance. Such topics are covered in the "Advanced Show Case", chapter 3.2.

#### 3.1.1 Physical System

This section contains a description of the physical system setup. It can safely be skipped, if only the AUTOSAR modeling itself is of interest to the reader.

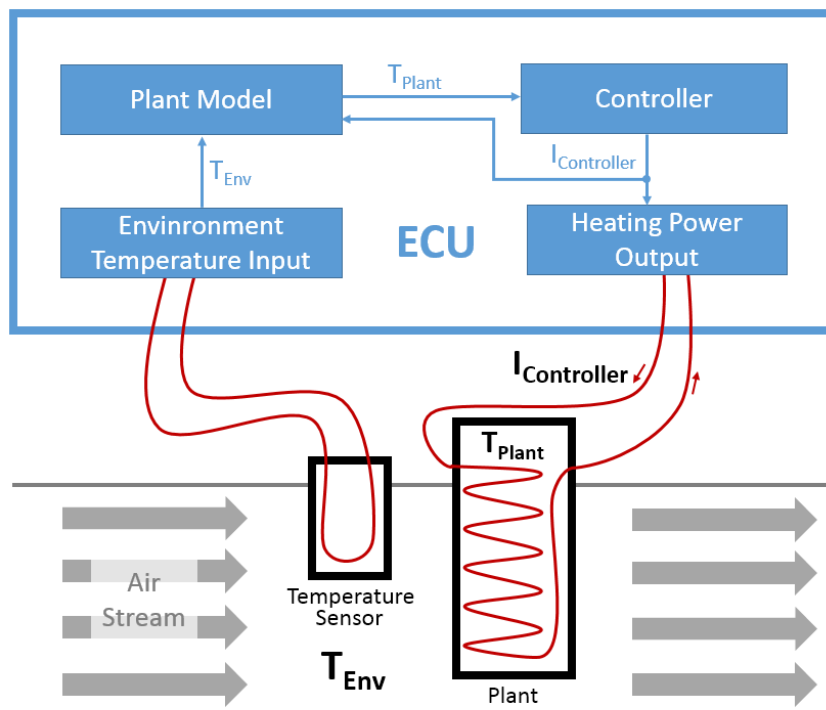


Figure 3.1: Physical Overview

Figure 3.1 shows the major physical values and entities of our system. The control task is the following: The plant is a sensor in an airstream that requires heating. The temperature  $T_{Plant}$  is to be controlled by the ECU. However, there is no direct way to measure  $T_{Plant}$ .

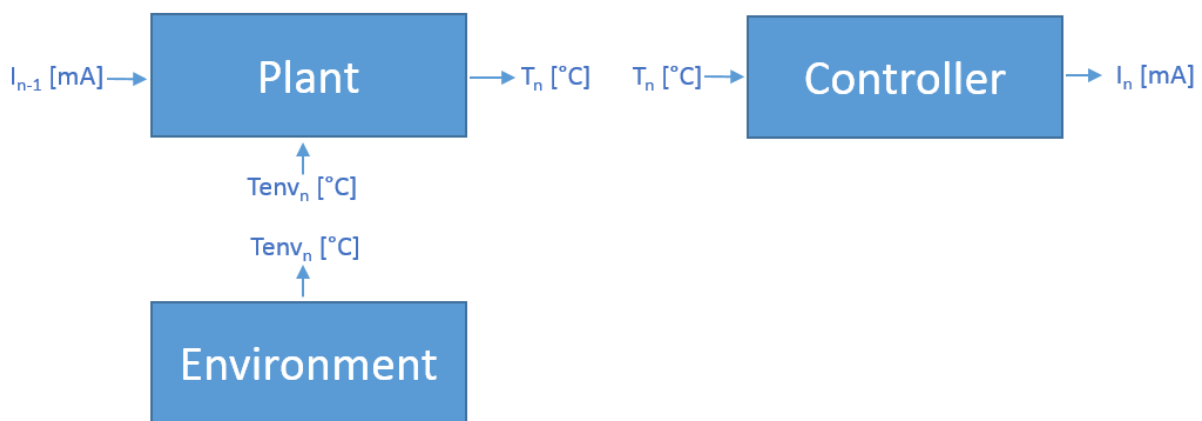
For the estimation of  $T_{Plant}$  the following properties of the system are used:

- $T_{Plant}$  depends on the temperature of the environment  $T_{Env}$ , i.e. the temperature of the air stream.  $T_{Env}$  can be measured directly.

- $T_{Plant}$  depends on the current  $I_{Controller}$  which is output by the ECU and controlled by the controller and therefore known.
- The plant itself acts as a thermal energy storage. So  $T_{Plant}$  also depends on the heat quantity that is currently stored within the plant.
- All other influences on  $T_{Plant}$  are considered to be insignificant. So they can safely be ignored for this control task.

An estimation of  $T_{Plant}$  can be calculated by a plant model, which uses  $T_{Env}$  and  $I_{Controller}$  as inputs and has the stored heat quantity as internal state.

### 3.1.1.1 Components Overview



**Figure 3.2: Component Overview**

For this show case, the interaction with a real physical environment is completely left out, i.e. there is no Heating Power Output component and the profile of  $T_{Env}$  is randomly generated inside the component Environment. This cuts off a lot of complexity from the example, and allows to run the software system on a PC without complex environment simulations.

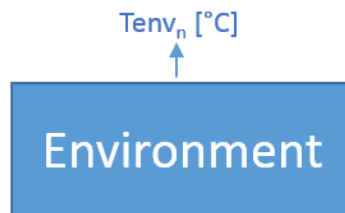
For completeness: The plant model is calculated inside the component Plant and the controller inside the component Controller.

As typical for ECUs the calculations happen in a time-discrete manner, i.e. the calculations in the components are executed periodically at discrete in time steps. In the following, the index  $n \in \{1, 2, \dots\}$  denotes the current time step. The previous time step is denoted by the index  $n - 1$ . The index 0 denotes the initialization value. This also means that time step 1 is the first, that is actually calculated by the ECU.

Furthermore  $\Delta t$  denotes the time in seconds, that elapsed between the calculation of the previous time step and the current time step. In case of time step 1,  $\Delta t$  denotes the time that elapsed between initialization of the system and time step 1. For setting

the actual value of  $\Delta t$  the frequency bandwidth of the physical properties in the system has to be taken into account. Decreasing the value of  $\Delta t$  usually increases the quality of the sampling of physical signals up to a certain point where the costs of further decreasing the value of  $\Delta t$  outweighs the benefit gained in terms of signal quality.

### 3.1.1.2 The Environment



**Figure 3.3: Environment**

The modeling and implementation of the environment is not in the focus of this show case. The temperature  $T_{env_n} [°C]$  is generated (pseudo) randomly. This is done in order to see the controller and the plant model "in action" during run-time of the system.

The generated profile is a random walk limited by an upper and a lower boundary, with saturation at these boundaries.

The random walk is configurable via  $T_{LowLimit} [°C]$  and  $T_{HighLimit} [°C]$ , for the boundaries, and  $T_{StepSize} [K]$ , for the change of the temperature during one time step.

Assuming  $rand_n [-] \in \{-1, 0, 1\}$  and  $n \in \{1, 2, 3, \dots\}$  then  $T_{env_n}$  is characterized by this equation (with  $T_{env_0} [°C] = -273.15 [°C]$ ):

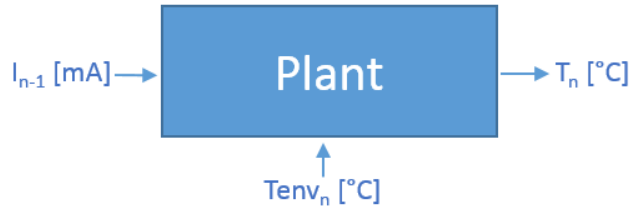
$$T_{env_n} [°C] = T_{env_{n-1}} [°C] + T_{StepSize} [K] \cdot rand_n [-]$$

if and only if  $T_{env_n}$  would be inside the boundaries, i.e.

$$T_{LowLimit} < T_{env_n} < T_{HighLimit}$$

If  $T_{env_n}$  would be outside one of the boundaries, it is set to the value of that boundary.

### 3.1.1.3 The Plant



**Figure 3.4: Plant**

The plant is an electrically heated mass that is exposed to the air flow in the environment. The heat quantity  $Q_{plant}$  that is stored inside the plant is considered to always be directly proportional to the temperature  $T$  with constant proportionality factor. Neither the mass of the plant, nor the specific heat capacity changes during the run-time of our system.

For simplicity, this proportionality factor is considered to be  $1 \left[ \frac{J}{K} \right]$ . For the calculations inside the `Plant` component, we are always using  $[K]$  as unit for temperatures, so the conversion from and to  $[°C]$  only happens at the interface of the component.

With this, we have the following:

$$Q_{plant_n} [J] = T_n [K] \cdot 1 \left[ \frac{J}{K} \right]$$

$$T_n [K] = \frac{Q_{plant_n} [J]}{1 \left[ \frac{J}{K} \right]}$$

This also means, that  $Q_{plant_n} [J] = 0 [J]$  corresponds  $T_n [K] = 0 [K]$ , i.e. absolute zero. So  $Q_{plant_n} [J] \geq 0 [J]$  shall always be true.

In each time step, there are two heat flows: One from the electrical heater to the plant and one from the plant to the environment. A negative heat flow means that heat energy is flowing away from the plant. Respectively, a positive heat flow means that heat energy is stored in the plant.

The heat flow  $Q_{heater_n} [J]$  from the electrical heater to the plant in one time step is considered to be proportional to the current  $I_n [mA]$  through the plant during this time step. The proportionality factor is  $h_{heater} \left[ \frac{J}{mA \cdot s} \right]$ . Of course, the plant can only be heated up by the electrical heater, i.e. a "negative" current  $I_n$  would not cool down the plant, but causes the same heat up as  $-I_n$ . So we have

$$Q_{heater_n} [J] = |I_n| [mA] \cdot h_{heater} \left[ \frac{J}{mA \cdot s} \right] \cdot \Delta t [s]$$

The cool down of the plant can only happen via the second heat flow, i.e. the heat flow  $Q_{env_n} [J]$  from the plant to the environment. The flow in one time step is considered

to be proportional to the difference between the temperature of the plant (calculated from the stored heat quantity during the last time step) and the temperature of the environment (received in this time step, but actually "measured" during the last time step). With the proportionality factor  $h_{Env} \left[ \frac{J}{K} \right]$ , we have:

$$Q_{env_n} [J] = (T_{env_n} [K] - T_{n-1} [K]) \cdot h_{Env} \left[ \frac{J}{K} \right] \cdot \Delta t [s]$$

The heat quantity that was stored in the plant in last time step  $Q_{plant_{n-1}}$  is now modified by these two heat flows. This results in the stored heat quantity in the current time step. With  $Q_{plant_0} [J] = 0 [J]$ , we have

$$Q_{plant_n} [J] = Q_{plant_{n-1}} [J] + Q_{heater_n} [J] + Q_{env_n} [J]$$

### 3.1.1.4 The Controller



**Figure 3.5: Controller**

For the closed loop control an I controller (by and large) was chosen for component `Controller`. This means that the amplification of the input signal is proportional to the integral of the errors, i.e. the deviation between measured variable and setpoint. Because the controller cannot actively cool down the temperature of the plant, the output  $I_n \geq 0$  for all  $n$ .

Again, all temperatures are converted to and from  $[^{\circ}C]$  at the interface of the component. All internal calculation are done in  $[K]$ .

The error during the current time step is the difference between  $T_{SetPoint} [K]$  and the measured variable  $T_n [K]$ :

$$e_n [K] = T_{SetPoint} [K] - T_n [K]$$

The integral part of the controller is calculated via summing up all errors from the previous steps. With  $eSum_n [Ks] = 0 [Ks]$  we have:

$$eSum_n [Ks] = eSum_{n-1} [Ks] + e_n [K] \cdot \Delta t$$

A further design decision for the controller was, to limit the integral and to saturate at the limits. This has the benefit that it limits the current  $I_n$  that is output by the controller. Furthermore, it enables the controller to react faster after long deviations.

The lower limit is  $0 \text{ [Ks]}$ . So if  $eSum_n$  would fall below zero in time step  $n$ , we set  $eSum_n \text{ [Ks]} = 0 \text{ [Ks]}$ . The upper limit is  $L_{MaxESum} \text{ [Ks]}$ . If  $eSum_n$  would exceed  $L_{MaxESum}$  in time step  $n$  we set  $eSum_n \text{ [Ks]} = L_{MaxESum} \text{ [Ks]}$ .

The integral state  $eSum_n$  of the controller is then amplified by  $k \left[ \frac{\text{mA}}{\text{Ks}} \right]$  to calculate the current  $I_n \text{ [mA]}$ , i.e. the output of the controller:

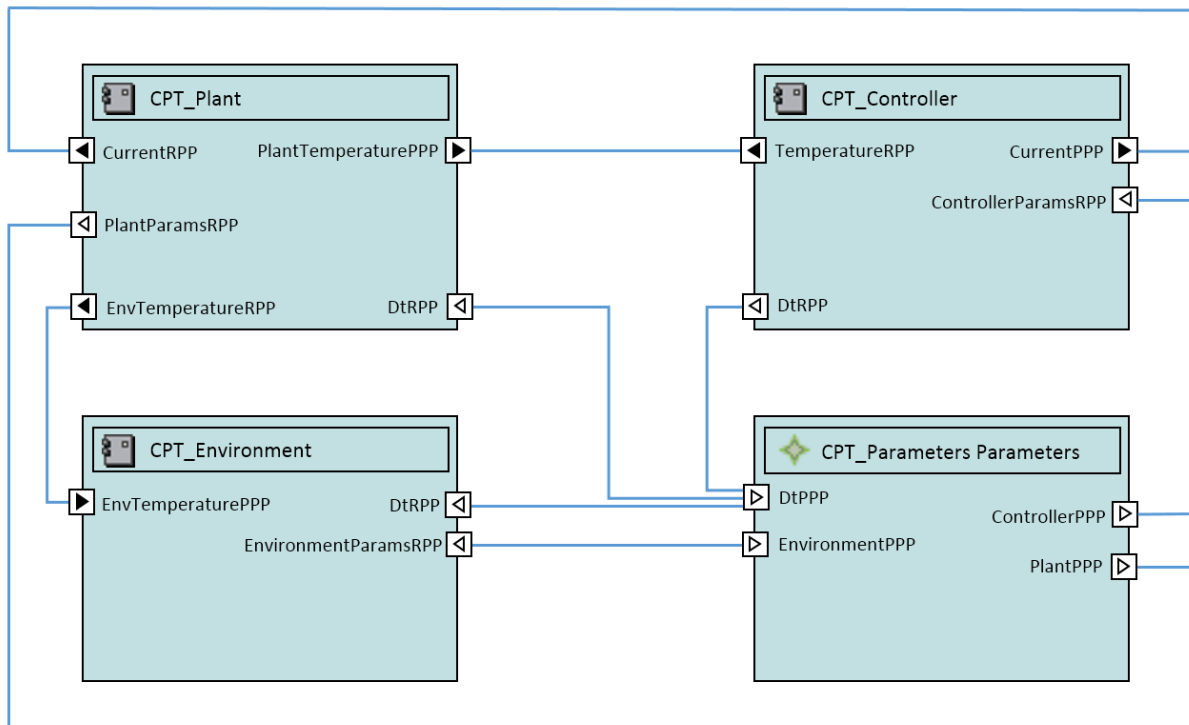
$$I_n \text{ [mA]} = eSum_n \text{ [Ks]} \cdot k \left[ \frac{\text{mA}}{\text{Ks}} \right]$$

So the limitations of the  $eSum_n$  guarantee, that

$$0 \text{ [mA]} \leq I_n \text{ [mA]} \leq L_{MaxESum} \text{ [Ks]} \cdot k \left[ \frac{\text{mA}}{\text{Ks}} \right]$$

### 3.1.2 AUTOSAR Modeling

This section gives a brief overview of the AUTOSAR modeling. More insight can be gained by browsing through the hyper-linked tables in section 3.1.7. These tables are generated from the AUTOSAR model of this show case. If this is still not sufficient, the complete model is available in .arxml format in AUTOSAR\_EXP\_ModelingShowCases.zip [2].



**Figure 3.6: Example Composition**

In this show case the components specified in section 3.1.1.1 are modeled as `ApplicationSwComponentTypes`.

- `Environment`
- `Plant`
- `Controller`

To keep the example simple, no `SwcImplementations` were modeled. For some tasks, like generation of a `MemMap` for an embedded controller, this would be needed.

The in- and outputs of the `ApplicationSwComponentTypes` are modeled as `SenderReceiverInterface`. The internal state is realized as `implicitInterRunnableVariables`. Besides the illustrative aspect, the rationale for this design decision was that the internal state is likely to be used by more than one runnable in `ApplicationSwComponentTypes` (at least "outside" of an introductory show case).

For variables that should just be available for measurement in a MC tool, `arTypedPerInstanceMemory`s are used. For this use case, no synchronization of access to the variable needs to be implemented, so the way with the least overhead was chosen.

All parameters in the specification of the components were put in a fourth `SwComponentType`, in the `ParameterSwComponentType` "Parameters".

A distinct `ParameterInterface` was defined for the parameters of each of the three `ApplicationSwComponentTypes`. The respective `PPortPrototypes` of the `ParameterSwComponentType` hold the `initValue` for each `ParameterDataPrototype` in the `PortPrototypes`. Each `value` is specified in a `ValueSpecification` aggregated by a `ParameterProvideComSpec`.

The component types are instantiated in the `CompositionSwComponentType` "Composition".

This `Composition` is the type of the `rootSoftwareComposition` of the `ECU_Extract`. This also implies that all `SwComponentPrototypes` of `Composition` are mapped to one `EcuInstance`.

Some information on the `FlatMap` can be found in section 3.1.3.1.

### 3.1.3 RTE Generation, Measurement and Calibration

The `McSupport` file is an interface between the RTE generator and the A2L generator. A RTE generator provides a `McDataInstance` for each calibrateable or measurable object. From logical view the generation of `McSupport` could be seen in two steps:

1. Provide unique names for all parameters, measurements, component prototypes which are instantiated one or multiple times. This is done by the used AUTOSAR Authoring Tool.
2. Generate the `McSupport` itself. This is usually done by the RTE generator. A2L supports only one global namespace, while AUTOSAR defines own namespaces within each `ARPackage`. This means, that on the one hand unique names are needed for all objects which are to be accessible during measurement and calibration (parameters, measurements, component prototypes). But on the other hand, unique names are needed for all other things that will appear in A2L, e.g. `CompuMethods`, `Units`. For them the RTE generator will create unique names.

AUTOSAR specifies additionally an `AliasNameSet` to override names which is not used here.

See `AUTOSAR_EXP_ModelingShowCases.zip` [2] for the generated `Rte_McSupportData.arxml` file.



### 3.1.3.1 FlatMap

In this show case, the `FlatMap` gives unique names to the

- `dataElements`
- `implicitInterRunnableVariables`
- `arTypedPerInstanceMemoryS`

The RTE-Generator uses this information for the generation of the `McSupport` file as well as for generation of the `.c` and `.h` files.

The `FlatMap` consists of a `FlatInstanceDescriptor` for each instance of these `VariableDataPrototypes`.

The flat map for this use case can be found in `AUTOSAR_EXP_ModelingShowCases.zip` [2].

### 3.1.3.2 ECU Documentation, Measurement and Calibration

When developing an ECU one usual requirement is, that objects described in A2L can be easily found in the documentation of the ECU. This is a challenge since documentation is on the level of `SwComponentTypes` while A2L is defined on the level of a `System` of `category` "ECU\_EXTRACT".

- The names of `SwComponentPrototypes` are potentially different to the names of `SwComponentTypes`
- The names of `McDataInstances` are potentially different to the names of `DataPrototypes`

The challenge gets bigger, if types are instantiated multiple times. This issue needs to be solved by proper architecture, modeling conventions and clever generation of the `FlatMap`.

In this show case, this topic is only slightly touched by instantiating `TemperatureSRIF` two times, for the interface transporting  $T_{Env}$  as well as for the interface transporting  $T_{Plant}$ .

It is demonstrated that the `FlatMap` can be used to solve the issue. However, we manually crafted our `FlatMap`, which is usually not possible in the field. `FlatMaps` are usually automatically generated by customizable, "clever", not standardized tools.

### 3.1.4 A2L File

With the information in the `McSupport` file an A2L file is generated. However, for this generation the memory addresses for the variables and characteristics are needed. They are usually extracted from the map file that is output by the linker of the ECU

executable. The exact process as well as the tool for the A2L file generation is not standardized.

An example A2L file is provided in `AUTOSAR_EXP_ModelingShowCases.zip`.

### 3.1.5 Implementation in C

The implementation in C is a straight forward realization of the physical specification within the AUTOSAR modeling (see section 3.1.1.1 and 3.1.2). Therefore, the listings are presented without further explanation besides the comments in the source code.

A remark on the (pseudo) random numbers generated in line 22 of `Environment.c` (Listing 3.1): The numbers don't have good "pseudo randomness" properties but are sufficient for this show case, nevertheless. This way of generation was only chosen, because it fits in one line of C code without introducing a dependency to a library.

**Listing 3.1: Environment.c**

```

1  #include "Rte_Environment.h"
2
3  #define envRE_START_SEC_CODE
4  #include "Environment_MemMap.h"
5
6  FUNC (void, Environment_CODE) envRE_func (void)
7  {
8      /* read parameters for simulation of the temperature profile */
9      float32 lLowLimit    = Rte_Prm_EnvParamsRPP_env_TLowLimit();
10     float32 lStepSize    = Rte_Prm_EnvParamsRPP_env_TStepSize();
11
12     /* retrieve internal state */
13     uint32  lSeed        = Rte_IrvIRead_envRE_Seed();
14     float32 lTEnv       = Rte_IrvIRead_envRE_TEnv();
15     float32 direction    = (float32)(lSeed % 3) - 1.0;
16
17     /* calc high limit with parameter, store for measurement */
18     *Rte_Pim_THighLimit()
19     = lLowLimit + Rte_Prm_EnvParamsRPP_env_THighLimitDistance();
20
21     /* update state for pseudo random number generation */
22     lSeed = (8253729 * lSeed + 2396403);
23
24     /* calculate environment temperature */
25     lTEnv += lStepSize * direction;
26
27     /* saturating environment temperature at the bounds */
28     if( lTEnv < lLowLimit) { lTEnv = lLowLimit; }
29     if( lTEnv > *Rte_Pim_THighLimit())
30         { lTEnv = *Rte_Pim_THighLimit(); }
31
32     /* Store internal state */
33     Rte_IrvIWrite_envRE_Seed(lSeed);
34     Rte_IrvIWrite_envRE_TEnv(lTEnv);
35
36     /* write output */
37     Rte_IWrite_envRE_EnvTemperaturePPP_T(lTEnv);
38 }
39 #define envRE_STOP_SEC_CODE
40 #include "Environment_MemMap.h"

```

**Listing 3.2: Plant.c**

```

1  #include "Rte_Plant.h"
2
3  #define plantRE_START_SEC_CODE
4  #include "Plant_MemMap.h"
5
6  FUNC (void, Plant_CODE) plantRE_func (void)
7  {
8      /* read input */ /*
9      float32 lTenv      = Rte_IRead_plantRE_EnvTemperatureRPP_T();
10     float32 lI         = Rte_IRead_plantRE_CurrentRPP_I();
11
12     /* retrieve internal state */ /*
13     float32 lQPlant    = Rte_IrvIRead_plantRE_QPlant();
14
15     /* read parameters */ /*
16     float32 lDt        = Rte_Prm_DtRPP_Dt();
17     float32 lEFactor   = Rte_Prm_PlantParamsRPP_plnt_EnvFactor();
18     float32 lHFactor   = Rte_Prm_PlantParamsRPP_plnt_HeaterFactor();
19
20     /* heat capacity of 1 assumed */ /*
21     float32 lTPlant    = lQPlant;
22
23     /* calculate heat flows, store in PIM to make them measurable */ /*
24     *Rte_Pim_QEnv()    = (lTenv - lTPlant) * lEFactor * lDt;
25     *Rte_Pim_QHeater() = lI * lHFactor * lDt;
26
27     /* update heat quantity in plant */ /*
28     lQPlant = lQPlant + *Rte_Pim_QHeater() + *Rte_Pim_QEnv();
29
30     /* limit heat quantity to absolute zero */ /*
31     lQPlant = lQPlant < 0 ? 0 : lQPlant;
32
33     /* heat capacity of 1 assumed */ /*
34     lTPlant = lQPlant;
35
36     /* store internal state of plant: stored heat quantity */ /*
37     Rte_IrvIWrite_plantRE_QPlant(lQPlant);
38
39     /* Write output of plant: temerature of plant */ /*
40     Rte_IWrite_plantRE_PlantTemperaturePPP_T(lTPlant);
41 }
42 #define plantRE_STOP_SEC_CODE
43 #include "Plant_MemMap.h"
    
```

**Listing 3.3: Controller.c**

```

1  #include "Rte_Controller.h"
2
3  #define ControllerRE_START_SEC_CODE
4  #include "Controller_MemMap.h"
5
6  FUNC (void, Controller_CODE) controllerRE_func (void)
7  {
8      /* read input, define output variable */
9      float32 lT      = Rte_IRead_ControllerRE_TemperatureRPP_T();
10     float32 lI;
11
12     /* retrieve internal state: Sum of errors until last time step */
13     float32 lESum    = Rte_IrviRead_ControllerRE_ESum();
14
15     /* read parameters */
16     float32 lDt      = Rte_Prm_DtRPP_Dt();
17     float32 lSetPoint = Rte_Prm_ControllerParamsRPP_ctrl_SetPoint();
18     float32 lK       = Rte_Prm_ControllerParamsRPP_ctrl_K();
19     float32 lMaxESum = Rte_Prm_ControllerParamsRPP_ctrl_MaxESum();
20
21     /* store current error in PIM to make it measurable */
22     *Rte_Pim_E() = lSetPoint - lT;
23
24     /* update eSum */
25     lESum += *Rte_Pim_E() * lDt;
26
27     /* limit eSum */
28     if(lESum > lMaxESum) { lESum = lMaxESum; }
29     if(lESum < 0)      { lESum = 0; }
30
31     /* Controller equation: Calculation of manipulated variable */
32     lI = lESum * lK;
33
34     /* Store internal state */
35     Rte_IrviWrite_ControllerRE_ESum(lESum);
36
37     /* Write output of controller */
38     Rte_IWrite_ControllerRE_CurrentPPP_I(lI);
39 }
40 #define ControllerRE_STOP_SEC_CODE
41 #include "Controller_MemMap.h"
    
```

### 3.1.6 A walk with $T_{Plant}$ through the Show Case

This section revisits the complete show case, but focuses on one physical value:  $T_{Plant}$ . It visits all artifacts and highlights all places that relate to  $T_{Plant}$  to illustrate the dependencies between all artifacts.

#### 3.1.6.1 Physical System

Our journey begins at the physical system, where the value of the physical system outside of the ECU is identified with a software value inside the ECU.

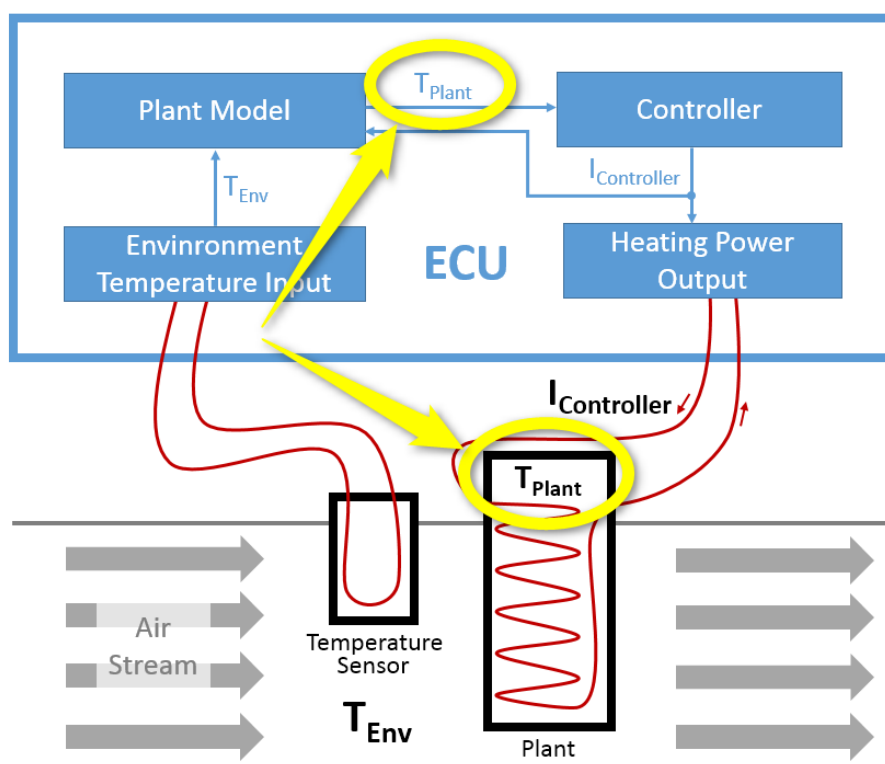
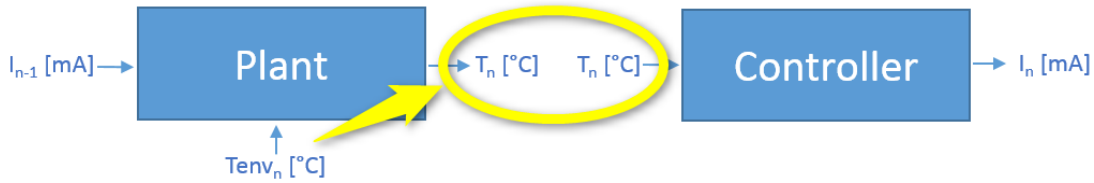


Figure 3.7: Physical Overview

#### 3.1.6.1.1 Components

It was located at the interface between two architectural components, sent by the `Plant` and received by the `Controller`. Furthermore a sequencing was introduced<sup>3</sup>, i.e. in one time step the `Plant` is calculated before the `Controller`.

<sup>3</sup>Please note that this sequencing is a design decision. As there is also a data flow from the `Plant` to the `Controller` one could also argue for another calculation sequence.



**Figure 3.8: Component Overview**

### 3.1.6.1.2 Equations

The functional behavior is defined by the equations for the `Plant`

$$Q_{plant_n} [J] = T_n [K] \cdot 1 \left[ \frac{J}{K} \right]$$

$$T_n [K] = \frac{Q_{plant_n} [J]}{1 \left[ \frac{J}{K} \right]}$$

**Figure 3.9: Dependency between  $Q_{Plant}$  and  $T_{Plant}$**

$$Q_{env_n} [J] = (T_{env_n} [K] - T_{n-1} [K]) \cdot h_{Env} \left[ \frac{J}{K} \right] \cdot \Delta t [s]$$

**Figure 3.10: Heat flow from the `Plant` to the `Environment`**

$T_{Plant}$  is also used by the physical equations in the component `Controller`:

$$e_n [K] = T_{SetPoint} [K] - T_n [K]$$

**Figure 3.11: Calculation of the control error in the `Controller`**

Furthermore, calculations inside the components are done in Kelvin [K]. The conversion from and to [°C] happens at the interface level.

### 3.1.6.2 AUTOSAR Modeling

This architecture, i.e. the layout of the physical system, is modeled in AUTOSAR. The functional behavior defined by the equations will be implemented in C Code later on.

### 3.1.6.2.1 Physical Dimension and Unit

A `PhysicalDimension` is defined:  $T_{Plant}$  is a temperature.

Common <code>PhysicalDimension</code> attributes	
<code>shortName</code>	Temperature
<code>currentExp</code>	0
<code>lengthExp</code>	0
<code>luminousIntensity-Exp</code>	0
<code>massExp</code>	0
<code>molarAmountExp</code>	0
<code>temperatureExp</code>	1
<code>timeExp</code>	0

**Table 3.1: PhysicalDimension Temperature**

The corresponding ARXML description is:

#### Listing 3.4: Physical Dimension of Temperature

```

<PHYSICAL-DIMENSION>
  <SHORT-NAME>Temperature</SHORT-NAME>
  <LENGTH-EXP>0</LENGTH-EXP>
  <MASS-EXP>0</MASS-EXP>
  <TIME-EXP>0</TIME-EXP>
  <CURRENT-EXP>0</CURRENT-EXP>
  <TEMPERATURE-EXP>1</TEMPERATURE-EXP>
  <MOLAR-AMOUNT-EXP>0</MOLAR-AMOUNT-EXP>
  <LUMINOUS-INTENSITY-EXP>0</LUMINOUS-INTENSITY-EXP>
</PHYSICAL-DIMENSION>
    
```

$T_{Plant}$  shall have the `Unit` `DegreeCelsius`:

Common <code>Unit</code> attributes	
<code>shortName</code>	DegreeCelsius
<code>displayName</code>	°C
<code>offsetSiToUnit</code>	-273.15
<code>factorSiToUnit</code>	1.0
<code>physicalDimension</code>	Temperature

**Table 3.2: Unit DegreeCelsius**

The corresponding ARXML description is:

#### Listing 3.5: Unit Degree Celsius

```

<UNIT>
  <SHORT-NAME>DegreeCelsius</SHORT-NAME>
  <DISPLAY-NAME>°C</DISPLAY-NAME>
    
```



```

<FACTOR-SI-TO-UNIT>1.0</FACTOR-SI-TO-UNIT>
<OFFSET-SI-TO-UNIT>-273.15</OFFSET-SI-TO-UNIT>
<PHYSICAL-DIMENSION-REF DEST="PHYSICAL-DIMENSION">
  /McInt/PhysicalDimensions/Temparature
</PHYSICAL-DIMENSION-REF>
</UNIT>
    
```

The following is presented for completeness, although not directly needed for  $T_{Plant}$ . It is possible to link more than one unit to a physical dimension. So in the model, there is also a definition for the unit Kelvin:

Common Unit attributes	
shortName	Kelvin
displayName	K
offsetSiToUnit	0.0
factorSiToUnit	1.0
physicalDimension	Temperature

**Table 3.3: Unit Kelvin**

The corresponding ARXML Code is:

**Listing 3.6: Unit Kelvin**

```

<UNIT>
  <SHORT-NAME>Kelvin</SHORT-NAME>
  <DISPLAY-NAME>K</DISPLAY-NAME>
  <FACTOR-SI-TO-UNIT>1.0</FACTOR-SI-TO-UNIT>
  <OFFSET-SI-TO-UNIT>0.0</OFFSET-SI-TO-UNIT>
  <PHYSICAL-DIMENSION-REF DEST="PHYSICAL-DIMENSION">
    /McInt/PhysicalDimensions/Temparature
  </PHYSICAL-DIMENSION-REF>
</UNIT>
    
```

### 3.1.6.2.2 Application Data Type

A new `ApplicationDataType` is defined for temperatures in degree Celsius:

Common <code>ApplicationDataType</code> attributes			
<code>shortName</code>	Temperature_C		
<code>category</code>	VALUE		
<code>desc</code>	Type for a temperature in [°C]		
<code>swCalibrationAccess</code>	readOnly		
<code>unit</code>	DegreeCelsius		
Range			
Conversion			
<code>category</code>	LINEAR		
<code>direction</code>	compuInternalToPhys		
<code>desc</code>	<code>lowerLimit</code>	<code>upperLimit</code>	<code>compuNumerator / compuDenominator</code>
-	-	-	$Phys = \frac{-273.15 + 1 * Internal}{1}$

**Table 3.4: ApplicationDataType Temperature\_C**

The corresponding ARXML Code is split between the definition of the `ApplicationDataType`:

**Listing 3.7: Datatype**

```

<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>Temperature_C</SHORT-NAME>
  <DESC>
    <L-2 L="EN">Type for a temperature in [°C]</L-2>
  </DESC>
  <CATEGORY>VALUE</CATEGORY>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
        <COMPU-METHOD-REF DEST="COMPU-METHOD">
          /McInt/CompuMethods/Temperature_C
        </COMPU-METHOD-REF>
        <UNIT-REF DEST="UNIT">/McInt/Units/DegreeCelsius</UNIT-REF>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
</APPLICATION-PRIMITIVE-DATA-TYPE>
    
```

and the `CompuMethod`, which is referenced by the `ApplicationDataType`:

**Listing 3.8: Conversion**

```

<COMPU-METHOD>
  <SHORT-NAME>Temperature_C</SHORT-NAME>
  <DESC>
    <L-2 L="EN">Conversion from [°C] to [K]</L-2>
  </DESC>
  </COMPU-METHOD>
    
```

```

</DESC>
<CATEGORY>LINEAR</CATEGORY>
<DISPLAY-FORMAT>%.1f</DISPLAY-FORMAT>
<UNIT-REF DEST="UNIT">/McInt/Units/DegreeCelsius</UNIT-REF>
<COMPU-INTERNAL-TO-PHYS>
<COMPU-SCALES>
  <COMPU-SCALE>
    <COMPU-RATIONAL-COEFFS>
      <COMPU-NUMERATOR>
        <V>-273.15</V>
        <V>1</V>
      </COMPU-NUMERATOR>
      <COMPU-DENOMINATOR>
        <V>1</V>
      </COMPU-DENOMINATOR>
    </COMPU-RATIONAL-COEFFS>
  </COMPU-SCALE>
</COMPU-SCALES>
</COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
    
```

This `ApplicationDataType` is mapped to the `ImplementationDataType` `float32`. The `DataTypeMappingSet` that contains this `DataTypeMap` is referenced inside the `SwcInternalBehaviors` of the `ApplicationSwComponentTypes` presented later on.

### Listing 3.9: Type Mapping

```

<DATA-TYPE-MAPPING-SET>
  <SHORT-NAME>DataTypeMappingSet</SHORT-NAME>
  <DATA-TYPE-MAPS>
    <DATA-TYPE-MAP>
      <APPLICATION-DATA-TYPE-REF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">
        /McInt/ApplicationDataTypes/Temperature_C
      </APPLICATION-DATA-TYPE-REF>
      <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
        /AUTOSAR_PlatformTypes/ImplementationDataTypes/float32
      </IMPLEMENTATION-DATA-TYPE-REF>
    </DATA-TYPE-MAP>
    ...
  </DATA-TYPE-MAPS>
</DATA-TYPE-MAPPING-SET>
    
```

For completeness, also the ARXML containing the definition of `float32` is inserted here:

### Listing 3.10: Implementation Type and Base Type

```

<AR-PACKAGE>
  <SHORT-NAME>AUTOSAR_PlatformTypes</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>ImplementationDataTypes</SHORT-NAME>
      <ELEMENTS>
        <IMPLEMENTATION-DATA-TYPE>
          <SHORT-NAME>float32</SHORT-NAME>
        </IMPLEMENTATION-DATA-TYPE>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>
    
```

```

<CATEGORY>VALUE</CATEGORY>
<SW-DATA-DEF-PROPS>
<SW-DATA-DEF-PROPS-VARIANTS>
  <SW-DATA-DEF-PROPS-CONDITIONAL>
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR_PlatformTypes/
      SwBaseTypes/float32</BASE-TYPE-REF>
  </SW-DATA-DEF-PROPS-CONDITIONAL>
</SW-DATA-DEF-PROPS-VARIANTS>
</SW-DATA-DEF-PROPS>
</IMPLEMENTATION-DATA-TYPE>
...
</ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
  <SHORT-NAME>SwBaseTypes</SHORT-NAME>
  <ELEMENTS>
    <SW-BASE-TYPE>
      <SHORT-NAME>float32</SHORT-NAME>
      <CATEGORY>FIXED_LENGTH</CATEGORY>
      <BASE-TYPE-SIZE>32</BASE-TYPE-SIZE>
      <BASE-TYPE-ENCODING>IEEE754</BASE-TYPE-ENCODING>
    </SW-BASE-TYPE>
    ...
  </ELEMENTS>
</AR-PACKAGE>
...
</AR-PACKAGE>
    
```

### 3.1.6.2.3 Port Interface

The `Temperature_C` is used to define the `SenderReceiverInterface` which is used to type the "transport" of a temperature in degree Celsius between `SwComponentTypes`. Please note that in the show case, this `PortInterface` is not only used to type the "transport" of  $T_{Plant}$ , but also to type the "transport" of  $T_{Env}$ .

Common <code>SenderReceiverInterface</code> attributes	
<code>shortName</code>	TemperatureSRIF
<code>desc</code>	Interface type for transferring temperatures in [°C]
properties of the <code>dataElementss</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	T
<code>type</code>	<code>Temperature_C</code>
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	VAR

**Table 3.5: SenderReceiverInterface TemperatureSRIF**

In ARXML:

### Listing 3.11: Port Interface

```

<SENDER-RECEIVER-INTERFACE>
  <SHORT-NAME>TemperatureSRIF</SHORT-NAME>
  <DESC>
    <L-2 L="EN">Interface type for transferring temperatures in [°C]</L-2
    >
  </DESC>
  <IS-SERVICE>>false</IS-SERVICE>
  <DATA-ELEMENTS>
    <VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>T</SHORT-NAME>
      <SW-DATA-DEF-PROPS>
        <SW-DATA-DEF-PROPS-VARIANTS>
          <SW-DATA-DEF-PROPS-CONDITIONAL>
            <SW-ADDR-METHOD-REF DEST="SW-ADDR-METHOD">/McInt/SwAddrMethods/
              VAR</SW-ADDR-METHOD-REF>
            <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
            <SW-IMPL-POLICY>STANDARD</SW-IMPL-POLICY>
          </SW-DATA-DEF-PROPS-CONDITIONAL>
        </SW-DATA-DEF-PROPS-VARIANTS>
      </SW-DATA-DEF-PROPS>
      <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/McInt/
        ApplicationDataTypes/Temperature_C</TYPE-TREF>
    </VARIABLE-DATA-PROTOTYPE>
  </DATA-ELEMENTS>
</SENDER-RECEIVER-INTERFACE>
    
```

For completeness, also the referenced [SwAddrMethod](#) is described here:

Common <a href="#">SwAddrMethod</a> attributes	
<a href="#">shortName</a>	VAR
<a href="#">desc</a>	Memory section for variables
<a href="#">sectionType</a>	<a href="#">var</a>
<a href="#">memoryAllocation-KeywordPolicy</a>	<a href="#">addrMethodShortName</a>
<a href="#">sectionInitializationPolicy</a>	-
<a href="#">option</a>	safetyQM

Table 3.6: SwAddrMethod VAR

In ARXML:

### Listing 3.12: Software Address Method

```

<SW-ADDR-METHOD>
  <SHORT-NAME>VAR</SHORT-NAME>
  <DESC>
    <L-2 L="EN">Memory section for variables</L-2>
  </DESC>
  <OPTIONS>
    
```

```

<OPTION>safetyQM</OPTION>
</OPTIONS>
</SW-ADDR-METHOD>
    
```

### 3.1.6.2.4 Software Components

The two `ApplicationSwComponentTypes` `Controller` and `Plant` are using  $T_{Plant}$ .

In `Plant` a `PPortPrototype`, typed by `TemperatureSRIF`, is defined for sending out  $T_{Plant}$ .

Furthermore `dataWriteAccess` is granted to the single `RunnableEntity` in this `ApplicationSwComponentType`. You also see the `symbol`, i.e. the name of the implementing C function, as well as the `TimingEvent` that triggers the execution of the `RunnableEntity`. These two are of further interest for tying together the system.

Common <code>ApplicationSwComponentType</code> attributes	
<code>shortName</code>	Plant
properties of the ports	
properties of <code>PPortPrototype</code>	
<code>shortName</code>	PlantTemperaturePPP
<code>desc</code>	Port for sending out the estimated temperature of the plant
<code>providedInterface</code>	<code>TemperatureSRIF</code>
[ ... ]	
<code>internalBehavior</code>	PlantInternalBehavior
[ ... ]	
properties of the runnables	
properties of <code>RunnableEntity</code>	
<code>shortName</code>	plantRE
<code>symbol</code>	plantRE_func
properties of the events	
properties of <code>TimingEvent</code>	
<code>shortName</code>	plant100ms
<code>startOnEvent</code>	plantRE
<code>period</code>	0.1

Table 3.7: `ApplicationSwComponentType` Plant

In ARXML:

#### Listing 3.13: Plant

```

<APPLICATION-SW-COMPONENT-TYPE>
  <SHORT-NAME>Plant</SHORT-NAME>
  <PORTS>
    <P-PORT-PROTOTYPE>
      <SHORT-NAME>PlantTemperaturePPP</SHORT-NAME>
      <DESC>
        <L-2 L="EN">Port for sending out the estimated temperature of the
          plant</L-2>
      </DESC>
      <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">
        /McInt/PortInterfaces/TemperatureSRIF
      </PROVIDED-INTERFACE-TREF>
    </P-PORT-PROTOTYPE>
    ...
  </PORTS>
  <INTERNAL-BEHAVIORS>
    <SWC-INTERNAL-BEHAVIOR>
      <SHORT-NAME>PlantInternalBehavior</SHORT-NAME>
      <DATA-TYPE-MAPPING-REFS>
        <DATA-TYPE-MAPPING-REF DEST="DATA-TYPE-MAPPING-SET">
          /McInt/DataTypeMappings/DataTypeMappingSet
        </DATA-TYPE-MAPPING-REF>
      </DATA-TYPE-MAPPING-REFS>
      <EVENTS>
        <TIMING-EVENT>
          <SHORT-NAME>plant100ms</SHORT-NAME>
          <START-ON-EVENT-REF DEST="RUNNABLE-ENTITY">
            /McInt/SwComponents/Plant/PlantInternalBehavior/plantRE
          </START-ON-EVENT-REF>
          <PERIOD>0.1</PERIOD>
        </TIMING-EVENT>
      </EVENTS>
      ...
    <RUNNABLES>
      <RUNNABLE-ENTITY>
        <SHORT-NAME>plantRE</SHORT-NAME>
        <DATA-WRITE-ACCESSS>
          <VARIABLE-ACCESS>
            <SHORT-NAME>DWA_PlantTemperature</SHORT-NAME>
            <ACCESSED-VARIABLE>
              <AUTOSAR-VARIABLE-IREF>
                <PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">
                  /McInt/SwComponents/Plant/PlantTemperaturePPP
                </PORT-PROTOTYPE-REF>
                <TARGET-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">
                  /McInt/PortInterfaces/TemperatureSRIF/T
                </TARGET-DATA-PROTOTYPE-REF>
              </AUTOSAR-VARIABLE-IREF>
            </ACCESSED-VARIABLE>
          </VARIABLE-ACCESS>
        </DATA-WRITE-ACCESSS>
        ...
      </RUNNABLE-ENTITY>
    </RUNNABLES>
  </SWC-INTERNAL-BEHAVIOR>
</INTERNAL-BEHAVIORS>
    
```

</APPLICATION-SW-COMPONENT-TYPE>

In `Controller` a `RPortPrototype`, typed by `TemperatureSRIF`, is defined for receiving  $T_{Plant}$ .

Furthermore `dataReadAccess` is granted to the single `RunnableEntity` in this `ApplicationSwComponentType`. You also see the `symbol`, i.e. the name of the implementing C function, as well as the `TimingEvent` that triggers the execution of the `RunnableEntity`. These two are of further interest for tying together the system.

Common <code>ApplicationSwComponentType</code> attributes	
<code>shortName</code>	Controller
<b>properties of the ports</b>	
<b>properties of <code>RPortPrototype</code></b>	
<code>shortName</code>	TemperatureRPP
<code>desc</code>	Port to receive the temperature of the plant
<code>requiredInterface</code>	TemperatureSRIF
[ ... ]	
<code>internalBehavior</code>	ControllerInternalBehavior
[ ... ]	
<b>properties of the runnables</b>	
<b>properties of <code>RunnableEntity</code></b>	
<code>shortName</code>	ControllerRE
<code>symbol</code>	controllerRE_func
<b>properties of the events</b>	
<b>properties of <code>TimingEvent</code></b>	
<code>shortName</code>	controller100ms
<code>startOnEvent</code>	ControllerRE
<code>period</code>	0.1

**Table 3.8: ApplicationSwComponentType Controller**

In ARXML:

### Listing 3.14: Controller

```

<APPLICATION-SW-COMPONENT-TYPE>
  <SHORT-NAME>Controller</SHORT-NAME>
  <PORTS>
    <R-PORT-PROTOTYPE>
      <SHORT-NAME>TemperatureRPP</SHORT-NAME>
      <DESC>
        <L-2 L="EN">Port to receive the temperature of the plant</L-2>
      </DESC>
      <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">
        /McInt/PortInterfaces/TemperatureSRIF
    </R-PORT-PROTOTYPE>
  </PORTS>
</APPLICATION-SW-COMPONENT-TYPE>
    
```



```

        </REQUIRED-INTERFACE-TREF>
    </R-PORT-PROTOTYPE>
    ...
</PORTS>
<INTERNAL-BEHAVIORS>
<SWC-INTERNAL-BEHAVIOR>
    <SHORT-NAME>ControllerInternalBehavior</SHORT-NAME>
    <DATA-TYPE-MAPPING-REFS>
    <DATA-TYPE-MAPPING-REF DEST="DATA-TYPE-MAPPING-SET">
        /McInt/DataTypeMappings/DataTypeMappingSet
    </DATA-TYPE-MAPPING-REF>
    </DATA-TYPE-MAPPING-REFS>
    <EVENTS>
    <TIMING-EVENT>
        <SHORT-NAME>controller100ms</SHORT-NAME>
        <START-ON-EVENT-REF DEST="RUNNABLE-ENTITY">
            /McInt/SwComponents/Controller/ControllerInternalBehavior/
            ControllerRE
        </START-ON-EVENT-REF>
        <PERIOD>0.1</PERIOD>
    </TIMING-EVENT>
    </EVENTS>
    ...
    <RUNNABLES>
    <RUNNABLE-ENTITY>
        <SHORT-NAME>ControllerRE</SHORT-NAME>
        <DATA-READ-ACCESSS>
        <VARIABLE-ACCESS>
            <SHORT-NAME>DRA_temperature</SHORT-NAME>
            <ACCESSED-VARIABLE>
            <AUTOSAR-VARIABLE-IREF>
                <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">
                    /McInt/SwComponents/Controller/TemperatureRPP
                </PORT-PROTOTYPE-REF>
                <TARGET-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">
                    /McInt/PortInterfaces/TemperatureSRIF/T
                </TARGET-DATA-PROTOTYPE-REF>
            </AUTOSAR-VARIABLE-IREF>
            </ACCESSED-VARIABLE>
        </VARIABLE-ACCESS>
        </DATA-READ-ACCESSS>
        ...
    </RUNNABLE-ENTITY>
    </RUNNABLES>
</SWC-INTERNAL-BEHAVIOR>
</INTERNAL-BEHAVIORS>
</APPLICATION-SW-COMPONENT-TYPE>
    
```

The two [ApplicationSwComponentTypes](#) are then used to type [SwComponent-Prototypes](#) in the [Composition](#). The [PortPrototypes](#) of the [SwComponent-Prototypes](#) are connected by an [AssemblySwConnector](#):

Common <code>CompositionSwComponentType</code> attributes	
<code>shortName</code>	Composition
properties of the <code>components</code>	
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CPT_Controller
<code>type</code>	<code>Controller</code>
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CPT_Plant
<code>type</code>	<code>Plant</code>
[ ... ]	

**Table 3.9: CompositionSwComponentType Composition**

In ARXML:

**Listing 3.15: Composition**

```

<COMPOSITION-SW-COMPONENT-TYPE>
  <SHORT-NAME>Composition</SHORT-NAME>
  <COMPONENTS>
    <SW-COMPONENT-PROTOTYPE>
      <SHORT-NAME>CPT_Controller</SHORT-NAME>
      <TYPE-TREF DEST="APPLICATION-SW-COMPONENT-TYPE">/McInt/SwComponents
        /Controller</TYPE-TREF>
    </SW-COMPONENT-PROTOTYPE>
      <SHORT-NAME>CPT_Plant</SHORT-NAME>
      <TYPE-TREF DEST="APPLICATION-SW-COMPONENT-TYPE">/McInt/SwComponents
        /Plant</TYPE-TREF>
    </SW-COMPONENT-PROTOTYPE>
    ...
  </COMPONENTS>
  <CONNECTORS>
    <ASSEMBLY-SW-CONNECTOR>
      <SHORT-NAME>
        ASC_CPT_Plant_TemperaturePPP_CPT_Controller_TemperatureRPP</
        SHORT-NAME>
      <PROVIDER-IREF>
      <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">/McInt/
        SwComponents/Composition/CPT_Plant</CONTEXT-COMPONENT-REF>
      <TARGET-P-PORT-REF DEST="P-PORT-PROTOTYPE">/McInt/SwComponents/
        Plant/PlantTemperaturePPP</TARGET-P-PORT-REF>
      </PROVIDER-IREF>
      <REQUESTER-IREF>
      <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">/McInt/
        SwComponents/Composition/CPT_Controller</CONTEXT-COMPONENT-REF>
      <TARGET-R-PORT-REF DEST="R-PORT-PROTOTYPE">/McInt/SwComponents/
        Controller/TemperatureRPP</TARGET-R-PORT-REF>
      </REQUESTER-IREF>
    </ASSEMBLY-SW-CONNECTOR>
    ...
  </CONNECTORS>
    
```



```

</ROOT-SW-COMPOSITION-PROTOTYPE>
</ROOT-SOFTWARE-COMPOSITIONS>
</SYSTEM>
    
```

The `FlatMap` that is referenced in the `ECU_Extract`, gives the name `TPlant` to a `dataElement` (see `ecuExtractReference` below). The name `TPlant` is later on displayed in the MC Tool.

### Listing 3.17: FlatMap

```

<FLAT-MAP>
  <SHORT-NAME>FlatMap</SHORT-NAME>
  <INSTANCES>
    <FLAT-INSTANCE-DESCRIPTOR>
      <SHORT-NAME>TPlant</SHORT-NAME>
      <ECU-EXTRACT-REFERENCE-IREF>
      <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE"/>McInt/
        System/EcuExtract/RootSwCompositionPrototype</CONTEXT-ELEMENT-
        REF>
      <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE"/>McInt/
        SwComponents/Composition/CPT_Plant</CONTEXT-ELEMENT-REF>
      <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE"/>McInt/SwComponents/
        Plant/PlantTemperaturePPP</CONTEXT-ELEMENT-REF>
      <TARGET-REF DEST="VARIABLE-DATA-PROTOTYPE"/>McInt/PortInterfaces/
        TemperatureSRIF/T</TARGET-REF>
      </ECU-EXTRACT-REFERENCE-IREF>
    </FLAT-INSTANCE-DESCRIPTOR>
    ...
  </INSTANCES>
</FLAT-MAP>
    
```

### 3.1.6.4 ECU Configuration

There are further things that need to be defined before the RTE and the OS can be generated. For instance, the order in which the `RTEEvents` for the `RunnableEntities` are invoked and the assignment to an `OsTask`. This is done via `EcucModuleConfigurationValues`. The interesting parts of the RTE configuration are:

### Listing 3.18: RTE Config

```

...
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>controller100ms</SHORT-NAME>
  <DEFINITION-REF ...>.../RteEventToTaskMapping</DEFINITION-REF>
  <PARAMETER-VALUES>
    <ECUC-NUMERICAL-PARAM-VALUE>
      <DEFINITION-REF ...>.../RtePositionInTask</DEFINITION-REF>
      <VALUE>3</VALUE>
    </ECUC-NUMERICAL-PARAM-VALUE>
    ...
  </PARAMETER-VALUES>
  <REFERENCE-VALUES>
  <ECUC-REFERENCE-VALUE>
    
```

```

        <DEFINITION-REF ...>.../RteMappedToTaskRef</DEFINITION-REF>
        <VALUE-REF ...>.../OS/OS_CFG/task_100ms</VALUE-REF>
    </ECUC-REFERENCE-VALUE>
<ECUC-REFERENCE-VALUE>
    <DEFINITION-REF ...>.../RteEventRef</DEFINITION-REF>
    <VALUE-REF DEST="TIMING-EVENT">.../controller100ms</VALUE-REF>
</ECUC-REFERENCE-VALUE>
</REFERENCE-VALUES>
</ECUC-CONTAINER-VALUE>
...
<ECUC-CONTAINER-VALUE>
    <SHORT-NAME>plant100ms</SHORT-NAME>
    <DEFINITION-REF ...>.../RteEventToTaskMapping</DEFINITION-REF>
    <PARAMETER-VALUES>
    <ECUC-NUMERICAL-PARAM-VALUE>
        <DEFINITION-REF ...>.../RtePositionInTask</DEFINITION-REF>
        <VALUE>2</VALUE>
    </ECUC-NUMERICAL-PARAM-VALUE>
    ...
</PARAMETER-VALUES>
<REFERENCE-VALUES>
<ECUC-REFERENCE-VALUE>
    <DEFINITION-REF ...>.../RteMappedToTaskRef</DEFINITION-REF>
    <VALUE-REF ...>.../OS/OS_CFG/task_100ms</VALUE-REF>
</ECUC-REFERENCE-VALUE>
<ECUC-REFERENCE-VALUE>
    <DEFINITION-REF ...>.../RteEventRef</DEFINITION-REF>
    <VALUE-REF DEST="TIMING-EVENT">.../plant100ms</VALUE-REF>
</ECUC-REFERENCE-VALUE>
</REFERENCE-VALUES>
</ECUC-CONTAINER-VALUE>
...

```

This part of the OS configuration defines the name of the OSTask, that we see later on in the generated C code:

### Listing 3.19: OsConfig

```

...
<AR-PACKAGE>
    <SHORT-NAME>OS</SHORT-NAME>
    <ELEMENTS>
    <ECUC-MODULE-CONFIGURATION-VALUES>
        <SHORT-NAME>OS_CFG</SHORT-NAME>
        <DEFINITION-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/Os</
            DEFINITION-REF>
        <CONTAINERS>
            ...
        <ECUC-CONTAINER-VALUE>
            <SHORT-NAME>task_100ms</SHORT-NAME>
            <DEFINITION-REF ...>.../OsTask</DEFINITION-REF>
            <PARAMETER-VALUES>
                ...
            </ECUC-CONTAINER-VALUE>
        ...
    </ECUC-MODULE-CONFIGURATION-VALUES>
</AR-PACKAGE>

```

These configurations are tied to the ECU\_Extract by an [EcucValueCollection](#):

### Listing 3.20: EcuC Value Collection

```

<ECUC-VALUE-COLLECTION>
  <SHORT-NAME>EcucValueCollection</SHORT-NAME>
  <ECU-EXTRACT-REF DEST="SYSTEM">/McInt/System/EcuExtract</ECU-EXTRACT-
    REF>
  <ECUC-VALUES>
  <ECUC-MODULE-CONFIGURATION-VALUES-REF-CONDITIONAL>
    <ECUC-MODULE-CONFIGURATION-VALUES-REF DEST="ECUC-MODULE-
      CONFIGURATION-VALUES">/McInt/RTE/RTE_CFG</ECUC-MODULE-
        CONFIGURATION-VALUES-REF>
  </ECUC-MODULE-CONFIGURATION-VALUES-REF-CONDITIONAL>
  <ECUC-MODULE-CONFIGURATION-VALUES-REF-CONDITIONAL>
    <ECUC-MODULE-CONFIGURATION-VALUES-REF DEST="ECUC-MODULE-
      CONFIGURATION-VALUES">/McInt/OS/OS_CFG</ECUC-MODULE-
        CONFIGURATION-VALUES-REF>
  </ECUC-MODULE-CONFIGURATION-VALUES-REF-CONDITIONAL>
  </ECUC-VALUES>
</ECUC-VALUE-COLLECTION>
    
```

This completes the presentation of the AUTOSAR modeling in our walk through.

### 3.1.6.5 RTE Generation

In the following, some snippets of the generated RTE are presented. However, they are examples only and may differ if different RTE generators are used.

Among other things, the `OsTask` is generated as defined in the ECU configuration above:

### Listing 3.21: Rte.c

```

1  ...
2  #define RTE_START_SEC_VAR
3  #include "MemMap.h" /*lint !e537 permit multiple inclusion */
4  ...
5  VAR(float32, RTE_DATA) TPlant;
6  ...
7  #define RTE_STOP_SEC_VAR
8  #include "MemMap.h" /*lint !e537 permit multiple inclusion */
9  ...
10 TASK(task_100ms)
11 {
12     ...
13     Rte_ImplicitBufs.isa_1._task_100ms.sbuf1.value = TPlant;
14     ...
15     plantRE_func();
16     ...
17     controllerRE_func();
18     ...
19     TPlant = Rte_ImplicitBufs.isa_1._task_100ms.sbuf1.value;
20     ...
21 } /* task_100ms */
    
```

22 ...

Also a MACRO to write  $T_{Plant}$  in the Plant

### Listing 3.22: Rte\_Plant.h

```

1 ...
2 #define Rte_IRead_plantRE_EnvTemperatureRPP_T() ((CONST(float32,
    RTE_DATA)) Rte_ImplicitBufs.isa_1._task_100ms.sbuf0.value )
3 ...
    
```

and to read  $T_{Plant}$  in the Controller

### Listing 3.23: Rte\_Controller.h

```

1 ...
2 #define Rte_IRead_ControllerRE_TemperatureRPP_T() ((CONST(float32,
    RTE_DATA)) Rte_ImplicitBufs.isa_1._task_100ms.sbuf1.value )
3 ...
    
```

was generated. Furthermore, the McSupport file is generated as an interface between the "AUTOSAR world" and the "A2L world". As the reader can see, this is a compilation of necessary data from the AUTOSAR model presented before:

### Listing 3.24: McSupportData

```

...
<AR-PACKAGE>
  <SHORT-NAME>BswImplementations</SHORT-NAME>
  <ELEMENTS>
    <BSW-IMPLEMENTATION>
      <SHORT-NAME>Rte</SHORT-NAME>
      <MC-SUPPORT>
        ...
        <MC-VARIABLE-INSTANCES>
          <MC-DATA-INSTANCE>
            <SHORT-NAME>TPlant</SHORT-NAME>
            <DESC>
              <L-2 L="EN">Type for a temperature in [°C]</L-2>
            </DESC>
            <CATEGORY>VALUE</CATEGORY>
            <FLAT-MAP-ENTRY-REF DEST="FLAT-INSTANCE-DESCRIPTOR"/>McInt/
              System/FlatMap/TPlant</FLAT-MAP-ENTRY-REF>
            <RESULTING-PROPERTIES>
              <SW-DATA-DEF-PROPS-VARIANTS>
                <SW-DATA-DEF-PROPS-CONDITIONAL>
                  <BASE-TYPE-REF BASE="Rte_MCSD_SwBaseTypes" DEST="SW-BASE-
                    TYPE">float32</BASE-TYPE-REF>
                  <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
                  <COMPU-METHOD-REF BASE="Rte_MCSD_CompuMethods" DEST="COMPU-
                    METHOD">McInt_CompuMethods_Temperature_C</COMPU-
                    METHOD-REF>
                  <DISPLAY-FORMAT>%.1f</DISPLAY-FORMAT>
                  <UNIT-REF BASE="Rte_MCSD_Units" DEST="UNIT">
                    McInt_Units_DegreeCelsius</UNIT-REF>
                </SW-DATA-DEF-PROPS-CONDITIONAL>
              </SW-DATA-DEF-PROPS-VARIANTS>
            </RESULTING-PROPERTIES>
          </MC-DATA-INSTANCE>
        </MC-VARIABLE-INSTANCES>
      </MC-SUPPORT>
    </BSW-IMPLEMENTATION>
  </ELEMENTS>
</AR-PACKAGE>
    
```

```

        </RESULTING-PROPERTIES>
        <SYMBOL>TPlant</SYMBOL>
        </MC-DATA-INSTANCE>
        ...
    </MC-VARIABLE-INSTANCES>
</ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
    <SHORT-NAME>Units</SHORT-NAME>
    <ELEMENTS>
        <UNIT>
            <SHORT-NAME>McInt_Units_DegreeCelsius</SHORT-NAME>
            <DISPLAY-NAME>°C</DISPLAY-NAME>
            <FACTOR-SI-TO-UNIT>1.0</FACTOR-SI-TO-UNIT>
            <OFFSET-SI-TO-UNIT>-273.15</OFFSET-SI-TO-UNIT>
            <PHYSICAL-DIMENSION-REF BASE="Rte_MCSD_PhysicalDimensions" DEST="
                PHYSICAL-DIMENSION">McInt_PhysicalDimensions_Temperature</
                PHYSICAL-DIMENSION-REF>
            </UNIT>
            ...
        </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
        <SHORT-NAME>CompuMethods</SHORT-NAME>
        <ELEMENTS>
            <COMPU-METHOD>
                <SHORT-NAME>McInt_CompuMethods_Temperature_C</SHORT-NAME>
                <DESC>
                    <L-2 L="EN">Conversion from [°C] at an interface to [K] for
                        internal computations</L-2>
                </DESC>
                <CATEGORY>LINEAR</CATEGORY>
                <DISPLAY-FORMAT>%f</DISPLAY-FORMAT>
                <UNIT-REF BASE="Rte_MCSD_Units" DEST="UNIT">
                    McInt_Units_DegreeCelsius</UNIT-REF>
                <COMPU-INTERNAL-TO-PHYS>
                    <COMPU-SCALES>
                        <COMPU-SCALE>
                            <COMPU-RATIONAL-COEFFS>
                                <COMPU-NUMERATOR>
                                    <V>-273.15</V>
                                    <V>1</V>
                                </COMPU-NUMERATOR>
                                <COMPU-DENOMINATOR>
                                    <V>1</V>
                                </COMPU-DENOMINATOR>
                            </COMPU-RATIONAL-COEFFS>
                        </COMPU-SCALE>
                    </COMPU-SCALES>
                </COMPU-INTERNAL-TO-PHYS>
            </COMPU-METHOD>
            ...
        </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
        <SHORT-NAME>PhysicalDimensions</SHORT-NAME>
    
```



```

<ELEMENTS>
  <PHYSICAL-DIMENSION>
    <SHORT-NAME>McInt_PhysicalDimensions_Temperature</SHORT-NAME>
    <LENGTH-EXP>0</LENGTH-EXP>
    <MASS-EXP>0</MASS-EXP>
    <TIME-EXP>0</TIME-EXP>
    <CURRENT-EXP>0</CURRENT-EXP>
    <TEMPERATURE-EXP>1</TEMPERATURE-EXP>
    <MOLAR-AMOUNT-EXP>0</MOLAR-AMOUNT-EXP>
    <LUMINOUS-INTENSITY-EXP>0</LUMINOUS-INTENSITY-EXP>
  </PHYSICAL-DIMENSION>
  ...
</AR-PACKAGE>
<AR-PACKAGE>
  <SHORT-NAME>SwBaseTypes</SHORT-NAME>
  <ELEMENTS>
    <SW-BASE-TYPE>
      <SHORT-NAME>float32</SHORT-NAME>
      <CATEGORY>FIXED_LENGTH</CATEGORY>
      <BASE-TYPE-SIZE>32</BASE-TYPE-SIZE>
      <BASE-TYPE-ENCODING>IEEE754</BASE-TYPE-ENCODING>
    </SW-BASE-TYPE>
    ...
  </ELEMENTS>
</AR-PACKAGE>
...

```

### 3.1.6.6 Implementation in C

The implementation in C-Code is a direct implementation of the physical equations. The Plant uses the MACRO, generated by the RTE generator, to write  $T_{Plant}$ :

**Listing 3.25: Plant**

```

1  #include "Rte_Plant.h"
2  ...
3  FUNC (void, Plant_CODE) plantRE_func (void)
4  {
5    ...
6    /* heat capacity of 1 assumed */
7    float32 lTPlant = lQPlant;
8    /* calculate heat flows, store in PIM to make them measurable */
9    *Rte_Pim_QEnv() = (lTenv - lTPlant) * lEfactor * lDt;
10   ...
11   /* heat capacity of 1 assumed */
12   lTPlant = lQPlant;
13   ...
14   /* Write output of plant: temerature of plant */
15   Rte_IWrite_plantRE_PlantTemperaturePPP_T(lTPlant);
16 }

```

The Controller uses the MACRO, generated by the RTE generator, to read  $T_{Plant}$ :

**Listing 3.26: Controller**

```

1  #include "Rte_Controller.h"
2  ...
3  FUNC (void, Controller_CODE) controllerRE_func (void)
4  {
5      /* read input, define output variable */
6      float32 lT          = Rte_IRead_ControllerRE_TemperatureRPP_T();
7      ...
8
9      /* store current error in PIM to make it measurable */
10     *Rte_Pim_E() = lSetPoint - lT;
11     ...
12 }
    
```

### 3.1.6.7 A2L File

Using the `McSupport` file and the map file from the linker, an example A2L file was generated for this show case. The snippet below is an example only and could differ if a different A2L file generator is used:

**Listing 3.27: A2L File**

```

1  ...
2  /begin MEASUREMENT TPlant
3      "TPlant"
4      FLOAT32_IEEE
5      McInt_CompuMethods_Temperature_C
6      0
7      0
8      -1E+32
9      1E+32
10     DISPLAY_IDENTIFIER "TPlant"
11     ECU_ADDRESS 0xe000001c
12     FORMAT "%.1f"
13     PHYS_UNIT "°C"
14 /end MEASUREMENT
15 ...
16 /begin UNIT McInt_PhysicalDimensions_Temperature
17     "McInt_PhysicalDimensions_Temperature"
18     "McInt_PhysicalDimensions_Temperature"
19     EXTENDED_SI
20     SI_EXPONENTS 0 0 0 0 1 0 0
21 /end UNIT
22 /begin UNIT McInt_Units_DegreeCelsius
23     "McInt_Units_DegreeCelsius"
24     "°C"
25     DERIVED
26     REF_UNIT McInt_PhysicalDimensions_Temperature
27     UNIT_CONVERSION 1 -273.15
28 /end UNIT
29 /begin COMPU_METHOD McInt_CompuMethods_Temperature_C
30     "McInt_CompuMethods_Temperature_C"
31     LINEAR
32     "%f"
33     "°C"
    
```

```

34          COEFFS_LINEAR 1 -273.15
35          REF_UNIT McInt_Units_DegreeCelsius
36 /end COMPU_METHOD
37 ...

```

### 3.1.6.8 Measurement and Calibration Tool

The A2L file is then used by a MC tool to measure  $T_{Plant}$ . Of course, in addition to the A2L file a suitable ECU access<sup>4</sup> must be available, to actually do measurement and calibration with the AUTOSAR system of this show case. However, the ECU access is not presented because this is not in the focus of this show case.

Below is a typical screen shot from a MC tool during an actual measurement and calibration task. You can see  $T_{Plant}$  measured and displayed in degree Celsius.

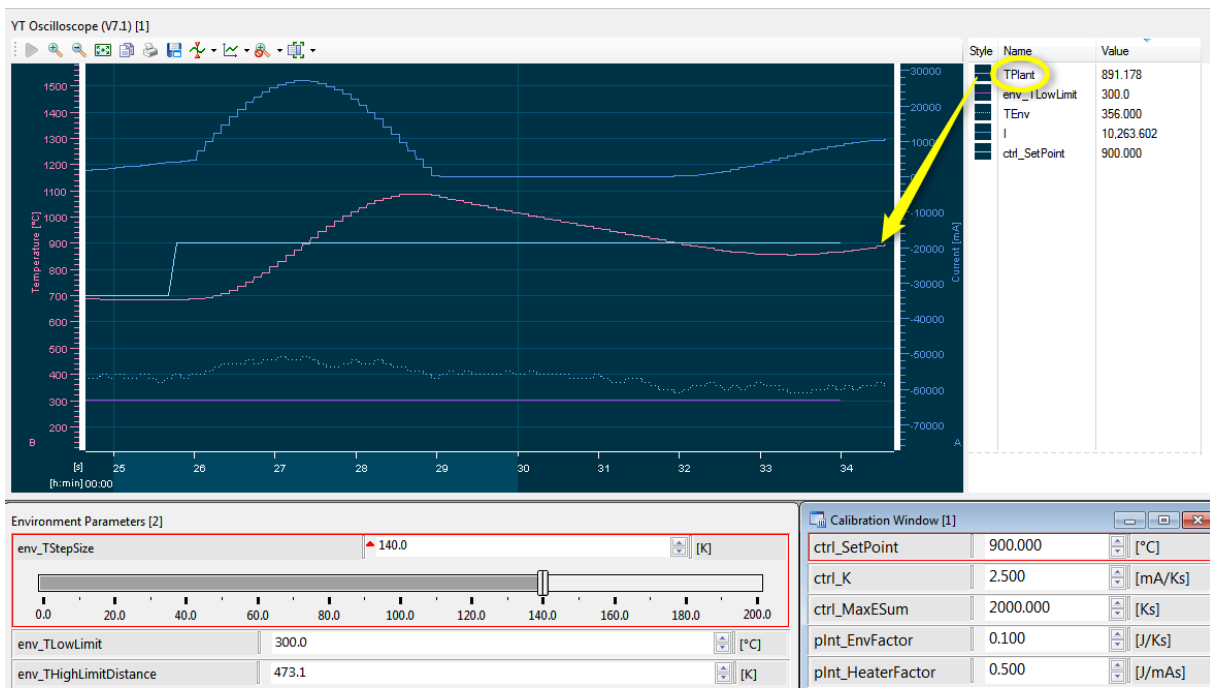


Figure 3.12: Screenshot of a MC Tool

<sup>4</sup>for instance, a measurement and calibration service like XCP or a hardware access to the memory of the micro controller

### 3.1.7 Show cases in the Example

#### 3.1.7.1 CompositionSwComponentTypes

Common <code>CompositionSwComponentType</code> attributes	
<code>shortName</code>	Composition
properties of the <code>components</code>	
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CPT_Controller
<code>type</code>	<code>Controller</code>
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CPT_Parameters
<code>type</code>	<code>Parameters</code>
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CPT_Plant
<code>type</code>	<code>Plant</code>
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CPT_Environment
<code>type</code>	<code>Environment</code>

**Table 3.10: CompositionSwComponentType Composition**

### 3.1.7.2 ParameterSwComponentTypes

Common <code>ParameterSwComponentType</code> attributes	
<code>shortName</code>	Parameters
<code>desc</code>	Type for providing the parameters to the ApplicationSwCompoments
<b>properties of the ports</b>	
<b>properties of <code>PPortPrototype</code></b>	
<code>shortName</code>	ControllerPPP
<code>desc</code>	Port for providing the parameters for the controller
<code>providedInterface</code>	<code>ControllerPIF</code>
<b>properties of <code>PPlantPrototype</code></b>	
<code>shortName</code>	PlantPPP
<code>desc</code>	Port for providing the parameters for the plant
<code>providedInterface</code>	<code>PlantPIF</code>
<b>properties of <code>PEnvironmentPrototype</code></b>	
<code>shortName</code>	EnvironmentPPP
<code>desc</code>	Port for providing the parameters for the environment
<code>providedInterface</code>	<code>EnvironmentPIF</code>
<b>properties of <code>PTimeStepPrototype</code></b>	
<code>shortName</code>	DtPPP
<code>desc</code>	Time of one time step
<code>providedInterface</code>	<code>DtPIF</code>

**Table 3.11: ParameterSwComponentType Parameters**

### 3.1.7.3 ApplicationSwComponentTypes

Common <code>ApplicationSwComponentType</code> attributes	
<code>shortName</code>	Controller
properties of the ports	
properties of <code>RPortPrototype</code>	
<code>shortName</code>	TemperatureRPP
<code>desc</code>	Port to receive the temperature of the plant
<code>requiredInterface</code>	TemperatureSRIF
properties of <code>PPortPrototype</code>	
<code>shortName</code>	CurrentPPP
<code>desc</code>	Port for sending out the current output by this controller
<code>providedInterface</code>	CurrentSRIF
properties of <code>RPortPrototype</code>	
<code>shortName</code>	ControllerParamsRPP
<code>desc</code>	Port to get the parameters for the controller
<code>requiredInterface</code>	ControllerPIF
properties of <code>RPortPrototype</code>	
<code>shortName</code>	DtRPP
<code>desc</code>	Port to get delta t, i.e. time of one time step
<code>requiredInterface</code>	DtPIF
<code>internalBehavior</code>	ControllerInternalBehavior

**Table 3.12: ApplicationSwComponentType Controller**

Common <code>SwcInternalBehavior</code> attributes	
<code>shortName</code>	ControllerInternalBehavior
properties of <code>implicitInterRunnableVariables</code> / <code>explicitInterRunnableVariables</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	ESum
<code>desc</code>	Internal state of the controller: the sum of control errors
<code>type</code>	ESum
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	CODE
properties of the <code>arTypedPerInstanceMemoryS</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	E
<code>desc</code>	Measurement point for the control error, the deviation between set point and actual temperature of the plant, in the current time step
<code>type</code>	Temperature_K
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	CODE
properties of the <code>runnables</code>	
properties of <code>RunnableEntity</code>	
<code>shortName</code>	ControllerRE
<code>symbol</code>	controllerRE_func
properties of the <code>events</code>	
properties of <code>TimingEvent</code>	
<code>shortName</code>	controller100ms
<code>startOnEvent</code>	ControllerRE
<code>period</code>	0.1

**Table 3.13: SwcInternalBehavior ControllerInternalBehavior**

Common <code>ApplicationSwComponentType</code> attributes	
<code>shortName</code>	Plant
properties of the ports	
properties of <code>RPortPrototype</code>	
<code>shortName</code>	CurrentRPP
<code>desc</code>	Port to receive the current from the controller
<code>requiredInterface</code>	<code>CurrentSRIF</code>
properties of <code>PPortPrototype</code>	
<code>shortName</code>	PlantTemperaturePPP
<code>desc</code>	Port for sending out the estimated temperature of the plant
<code>providedInterface</code>	<code>TemperatureSRIF</code>
properties of <code>RPortPrototype</code>	
<code>shortName</code>	PlantParamsRPP
<code>desc</code>	Port to get the parameters for the plant
<code>requiredInterface</code>	<code>PlantPIF</code>
properties of <code>RPortPrototype</code>	
<code>shortName</code>	EnvTemperatureRPP
<code>desc</code>	Port to receive the temperature of the environment
<code>requiredInterface</code>	<code>TemperatureSRIF</code>
properties of <code>RPortPrototype</code>	
<code>shortName</code>	DtRPP
<code>desc</code>	Port to get delta t, i.e. time of one time step
<code>requiredInterface</code>	<code>DtPIF</code>
<code>internalBehavior</code>	<code>PlantInternalBehavior</code>

**Table 3.14: ApplicationSwComponentType Plant**



Common <code>SwcInternalBehavior</code> attributes	
<code>shortName</code>	PlantInternalBehavior
properties of <code>implicitInterRunnableVariables</code> / <code>explicitInterRunnableVariables</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	QPlant
<code>desc</code>	Internal state of the plant: the stored eneregy quantity in the current time step
<code>type</code>	Energy
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	CODE
properties of the <code>arTypedPerInstaceMemoryS</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	QHeater
<code>desc</code>	Measurement point for heat flow between the electrical heater and the plant in the current time step.
<code>type</code>	Energy
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	VAR
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	QEnv
<code>desc</code>	Measurement point for heat flow between the plant and the environment in the current time step.
<code>type</code>	Energy
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	CODE
properties of the <code>runnables</code>	
properties of <code>RunnableEntity</code>	
<code>shortName</code>	plantRE
<code>symbol</code>	plantRE_func
properties of the <code>events</code>	
properties of <code>TimingEvent</code>	
<code>shortName</code>	plant100ms
<code>startOnEvent</code>	plantRE
<code>period</code>	0.1





**Table 3.15: SwcInternalBehavior PlantInternalBehavior**

Common <code>ApplicationSwComponentType</code> attributes	
<code>shortName</code>	Environment
properties of the ports	
properties of <code>PPortPrototype</code>	
<code>shortName</code>	EnvTemperaturePPP
<code>desc</code>	Port to send out the temperature of the environment
<code>providedInterface</code>	TemperatureSRIF
properties of <code>RPortPrototype</code>	
<code>shortName</code>	EnvParamsRPP
<code>desc</code>	Port to get the parameters for the environment
<code>requiredInterface</code>	EnvironmentPIF
properties of <code>RPortPrototype</code>	
<code>shortName</code>	DtRPP
<code>desc</code>	Port to get delta t, i.e. time of one time step
<code>requiredInterface</code>	DtPIF
<code>internalBehavior</code>	EnvironmentInternalBehavior

**Table 3.16: ApplicationSwComponentType Environment**

Common <code>SwcInternalBehavior</code> attributes	
<code>shortName</code>	EnvironmentInternalBehavior
properties of <code>implicitInterRunnableVariables</code> / <code>explicitInterRunnableVariables</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	Seed
<code>desc</code>	Internal state of the environment: the current seed for the (pseudo) random number generation
<code>type</code>	uint32
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	notAccessible
<code>swAddrMethod</code>	CODE
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	TEnv
<code>desc</code>	Internal state of the environment: the temperature of the environment
<code>type</code>	Temperature_C
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	CODE
properties of the <code>arTypedPerInstanceMemoryS</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	THighLimit
<code>desc</code>	Measurement point for the upper limit of the generated temperature profile
<code>type</code>	Temperature_C
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	CODE
properties of the <code>runnables</code>	
properties of <code>RunnableEntity</code>	
<code>shortName</code>	envRE
<code>symbol</code>	envRE_func
properties of the <code>events</code>	
properties of <code>TimingEvent</code>	
<code>shortName</code>	env100ms
<code>startOnEvent</code>	envRE
<code>period</code>	0.1





---

**Table 3.17: SwcInternalBehavior EnvironmentInternalBehavior**

### 3.1.7.4 ParameterInterfaces

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	ControllerPIF
<code>desc</code>	Interface with all parameters for the controller
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	ctrl_SetPoint
<code>desc</code>	Set point for the temperature of the plant
<code>type</code>	Temperature_C
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CALIB
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	ctrl_K
<code>desc</code>	Amplification factor for the I-controller
<code>type</code>	Amplification
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CALIB
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	ctrl_MaxESum
<code>desc</code>	Upper limit of the integral part of the I-controller
<code>type</code>	ESum
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CALIB

**Table 3.18: ParameterInterface ControllerPIF**

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	PlantPIF
<code>desc</code>	Interface with all parameters for the plant
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	plnt_EnvFactor
<code>desc</code>	Proportionality factor for the heat flow between plant and environment
<code>type</code>	<code>EnvFactor</code>
<code>swImplPolicy</code>	<code>standard</code>
<code>swCalibrationAccess</code>	<code>readWrite</code>
<code>swAddrMethod</code>	<code>CALIB</code>
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	plnt_HeaterFactor
<code>desc</code>	Proportionality factor for the heat flow between plant and the electrical heater
<code>type</code>	<code>HeaterFactor</code>
<code>swImplPolicy</code>	<code>standard</code>
<code>swCalibrationAccess</code>	<code>readWrite</code>
<code>swAddrMethod</code>	<code>CALIB</code>

**Table 3.19: ParameterInterface PlantPIF**

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	EnvironmentPIF
<code>desc</code>	Interface with all parameters for the environment
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	env_TLowLimit
<code>desc</code>	Lower limit of the generated temepature profile
<code>type</code>	Temperature_C
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CALIB
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	env_TStepSize
<code>desc</code>	The maximal temperature diffenrence of the environment in one time step
<code>type</code>	Temperature_K
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CALIB
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	env_THighLimitDistance
<code>desc</code>	Distance of the upper limit from the lower limit for the generated temepature profile.
<code>type</code>	Temperature_K
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CALIB

**Table 3.20: ParameterInterface EnvironmentPIF**

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	DtPIF
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	Dt
<code>desc</code>	Scheduling time of the components
<code>type</code>	Time
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CALIB

**Table 3.21: ParameterInterface DtPIF**

### 3.1.7.5 SenderReceiverInterfaces

Common <code>SenderReceiverInterface</code> attributes	
<code>shortName</code>	TemperatureSRIF
<code>desc</code>	Interface type for transferring temperatures in [°C]
properties of the <code>dataElementss</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	T
<code>type</code>	Temperature_C
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	VAR

**Table 3.22: SenderReceiverInterface TemperatureSRIF**

Common <code>SenderReceiverInterface</code> attributes	
<code>shortName</code>	CurrentSRIF
<code>desc</code>	Interface type for transferring a current in [mA]
properties of the <code>dataElementss</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	I
<code>type</code>	Current
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	VAR

**Table 3.23: SenderReceiverInterface CurrentSRIF**



### 3.1.7.6 ApplicationDataTypes, Category VALUE

Common <code>ApplicationDataType</code> attributes			
<code>shortName</code>	Temperature_C		
<code>category</code>	VALUE		
<code>desc</code>	Type for a temperature in [°C]		
<code>swCalibrationAccess</code>	<code>readOnly</code>		
<code>unit</code>	<code>DegreeCelsius</code>		
Range			
Conversion			
<code>category</code>	LINEAR		
<code>direction</code>	<code>compuInternalToPhys</code>		
<code>desc</code>	<code>lowerLimit</code>	<code>upperLimit</code>	<code>compuNumerator / compuDenominator</code>
-	-	-	$Phys = \frac{-273.15 + 1 * Internal}{1}$

Table 3.24: ApplicationDataType Temperature\_C

Common <code>ApplicationDataType</code> attributes			
<code>shortName</code>	Current		
<code>category</code>	VALUE		
<code>desc</code>	Type for the current in [mA]		
<code>swCalibrationAccess</code>	<code>readOnly</code>		
<code>unit</code>	<code>MilliAmpere</code>		
Range			
Conversion			
<code>category</code>	LINEAR		
<code>direction</code>	<code>compuInternalToPhys</code>		
<code>desc</code>	<code>lowerLimit</code>	<code>upperLimit</code>	<code>compuNumerator / compuDenominator</code>
-	-	-	$Phys = \frac{0 + 1000 * Internal}{1}$

Table 3.25: ApplicationDataType Current

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	EnvFactor
<code>category</code>	VALUE
<code>desc</code>	Type for the environt factor in [J/Ks]
<code>swCalibrationAccess</code>	<code>readOnly</code>
<code>unit</code>	<code>JoulePerKelvinSecond</code>
Range	
Conversion	
<code>category</code>	IDENTICAL
<code>direction</code>	-

**Table 3.26: ApplicationDataType EnvFactor**

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	Temperature_K
<code>category</code>	VALUE
<code>desc</code>	Type for a temperature in [K]
<code>swCalibrationAccess</code>	<code>readOnly</code>
<code>unit</code>	<code>Kelvin</code>
Range	
Conversion	
<code>category</code>	IDENTICAL
<code>direction</code>	-

**Table 3.27: ApplicationDataType Temperature\_K**

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	Amplification
<code>category</code>	VALUE
<code>desc</code>	Type for an amplification factor in a controller in [mA/Ks]
<code>swCalibrationAccess</code>	<code>readOnly</code>
<code>unit</code>	<code>MilliAmperePerKelvinSecond</code>
Range	
Conversion	
<code>category</code>	IDENTICAL
<code>direction</code>	-

**Table 3.28: ApplicationDataType Amplification**

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	Energy
<code>category</code>	VALUE
<code>desc</code>	Type for energy [J]
<code>swCalibrationAccess</code>	<code>readOnly</code>
<code>unit</code>	<code>Joule</code>
Range	
Conversion	
<code>category</code>	IDENTICAL
<code>direction</code>	-

**Table 3.29: ApplicationDataType Energy**

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	ESum
<code>category</code>	VALUE
<code>desc</code>	Type for the sum of control errors of an I controller in [Ks]
<code>swCalibrationAccess</code>	<code>readOnly</code>
<code>unit</code>	<code>KelvinSecond</code>
Range	
Conversion	
<code>category</code>	IDENTICAL
<code>direction</code>	-

**Table 3.30: ApplicationDataType ESum**

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	HeaterFactor
<code>category</code>	VALUE
<code>desc</code>	Type of a proportionality factor for the heat flow from an electrical heater to a thermal energy storage in [J/mAs]
<code>swCalibrationAccess</code>	<code>readOnly</code>
<code>unit</code>	<code>JoulePerMilliAmpereSecond</code>
Range	
Conversion	
<code>category</code>	IDENTICAL
<code>direction</code>	-

**Table 3.31: ApplicationDataType HeaterFactor**

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	Time
<code>category</code>	VALUE
<code>desc</code>	Type for time in [s]
<code>swCalibrationAccess</code>	<code>readOnly</code>
<code>unit</code>	<code>Second</code>
<b>Range</b>	
<b>Conversion</b>	
<code>category</code>	IDENTICAL
<code>direction</code>	-

**Table 3.32: ApplicationDataType Time**

### 3.1.7.7 Units

Common Unit attributes	
shortName	DegreeCelsius
displayName	°C
offsetSiToUnit	-273.15
factorSiToUnit	1.0
physicalDimension	Temperature

**Table 3.33: Unit DegreeCelsius**

Common Unit attributes	
shortName	Kelvin
displayName	K
offsetSiToUnit	0.0
factorSiToUnit	1.0
physicalDimension	Temperature

**Table 3.34: Unit Kelvin**

Common Unit attributes	
shortName	Joule
displayName	J
offsetSiToUnit	0.0
factorSiToUnit	1.0
physicalDimension	Energy

**Table 3.35: Unit Joule**

Common Unit attributes	
shortName	MilliAmpere
displayName	mA
offsetSiToUnit	0.0
factorSiToUnit	1000.0
physicalDimension	Current

**Table 3.36: Unit MilliAmpere**

Common Unit attributes	
shortName	KelvinSecond
displayName	Ks
offsetSiToUnit	0.0
factorSiToUnit	1.0
physicalDimension	TemperatureTime

**Table 3.37: Unit KelvinSecond**

Common Unit attributes	
shortName	JoulePerKelvinSecond
displayName	J/Ks
offsetSiToUnit	0.0
factorSiToUnit	1.0
physicalDimension	EnergyPerTemperatureTime

**Table 3.38: Unit JoulePerKelvinSecond**

Common Unit attributes	
shortName	JoulePerMilliAmpereSecond
displayName	J/mAs
offsetSiToUnit	0.0
factorSiToUnit	0.001
physicalDimension	EnergyPerCurrentTime

**Table 3.39: Unit JoulePerMilliAmpereSecond**

Common Unit attributes	
shortName	MilliAmperePerKelvinSecond
displayName	mA/Ks
offsetSiToUnit	0.0
factorSiToUnit	1000.0
physicalDimension	CurrentPerTemperatureTime

**Table 3.40: Unit MilliAmperePerKelvinSecond**

Common Unit attributes	
<b>shortName</b>	Second
<b>displayName</b>	s
<b>offsetSiToUnit</b>	0.0
<b>factorSiToUnit</b>	1.0
<b>physicalDimension</b>	Time

**Table 3.41: Unit Second**

### 3.1.7.8 PhysicalDimensions

Common <code>PhysicalDimension</code> attributes	
<code>shortName</code>	Energy
<code>currentExp</code>	0
<code>lengthExp</code>	2
<code>luminousIntensity-Exp</code>	0
<code>massExp</code>	1
<code>molarAmountExp</code>	0
<code>temperatureExp</code>	0
<code>timeExp</code>	-2

**Table 3.42: PhysicalDimension Energy**

Common <code>PhysicalDimension</code> attributes	
<code>shortName</code>	Current
<code>currentExp</code>	1
<code>lengthExp</code>	0
<code>luminousIntensity-Exp</code>	0
<code>massExp</code>	0
<code>molarAmountExp</code>	0
<code>temperatureExp</code>	0
<code>timeExp</code>	0

**Table 3.43: PhysicalDimension Current**

Common <code>PhysicalDimension</code> attributes	
<code>shortName</code>	CurrentPerTemperatureTime
<code>currentExp</code>	1
<code>lengthExp</code>	0
<code>luminousIntensity-Exp</code>	0
<code>massExp</code>	0
<code>molarAmountExp</code>	0
<code>temperatureExp</code>	-1
<code>timeExp</code>	-1

**Table 3.44: PhysicalDimension CurrentPerTemperatureTime**



Common <code>PhysicalDimension</code> attributes	
shortName	EnergyPerCurrentTime
currentExp	-1
lengthExp	2
luminousIntensity-Exp	0
massExp	1
molarAmountExp	0
temperatureExp	0
timeExp	-3

**Table 3.45: PhysicalDimension EnergyPerCurrentTime**

Common <code>PhysicalDimension</code> attributes	
shortName	EnergyPerTemperatureTime
currentExp	0
lengthExp	2
luminousIntensity-Exp	0
massExp	1
molarAmountExp	0
temperatureExp	-1
timeExp	-3

**Table 3.46: PhysicalDimension EnergyPerTemperatureTime**

Common <code>PhysicalDimension</code> attributes	
shortName	Time
currentExp	0
lengthExp	0
luminousIntensity-Exp	0
massExp	0
molarAmountExp	0
temperatureExp	0
timeExp	1

**Table 3.47: PhysicalDimension Time**

Common <code>PhysicalDimension</code> attributes	
<code>shortName</code>	Temperature
<code>currentExp</code>	0
<code>lengthExp</code>	0
<code>luminousIntensity-Exp</code>	0
<code>massExp</code>	0
<code>molarAmountExp</code>	0
<code>temperatureExp</code>	1
<code>timeExp</code>	0

**Table 3.48: PhysicalDimension Temperature**

Common <code>PhysicalDimension</code> attributes	
<code>shortName</code>	TemperatureTime
<code>currentExp</code>	0
<code>lengthExp</code>	0
<code>luminousIntensity-Exp</code>	0
<code>massExp</code>	0
<code>molarAmountExp</code>	0
<code>temperatureExp</code>	1
<code>timeExp</code>	1

**Table 3.49: PhysicalDimension TemperatureTime**

### 3.1.7.9 SwAddrMethods

Common SwAddrMethod attributes	
shortName	VAR
desc	Memory section for variables
sectionType	var
memoryAllocation-KeywordPolicy	addrMethodShortName
sectionInitializa-tionPolicy	-
option	safetyQM

**Table 3.50: SwAddrMethod VAR**

Common SwAddrMethod attributes	
shortName	CALIB
desc	Memory section for calibration parameters
sectionType	var
memoryAllocation-KeywordPolicy	addrMethodShortName
sectionInitializa-tionPolicy	-
option	safetyQM

**Table 3.51: SwAddrMethod CALIB**

Common SwAddrMethod attributes	
shortName	CODE
desc	Memory section for code
sectionType	var
memoryAllocation-KeywordPolicy	addrMethodShortName
sectionInitializa-tionPolicy	-
option	safetyQM

**Table 3.52: SwAddrMethod CODE**

## 3.2 Advanced Show Case

### 3.2.1 General Objectives of the Model Structure

#### 3.2.1.1 The Ecu Description

Since the show case is focusing on measurement and calibration only a minimal system model is provided. Hereby the file `Pprj_EcuDescr_U_SystemNodeStub.arxml` defines the `System` `SystemU_EcuDescr` of `category` `ECU_SYSTEM_DESCRIPTION` which contains only the `RootSwCompositionPrototype`. The file `Pprj_EcuDescr_U.arxml` contains the according `CompositionSwComponentType` describing the hierarchical top-level-composition of software components shown in table `SystemURootComposition_EcuDescr`.

#### 3.2.1.2 The Ecu Extract

The file `Pprj_EcuExtract_U_SystemNodeStub.arxml` defines the `System` `SystemU_System` of `category` `ECU_EXTRACT` which contains only the `RootSwCompositionPrototype` `SystemU` referencing the ECU Flat Map and the flat top-level-composition `SystemU_Root`. The file `Pprj_EcuExtract_U.arxml` contains the according `CompositionSwComponentType` describing the flat top-level-composition of software components shown in table `SystemU_Root`.

Please note that the flat top-level-composition uses the identical software component types as the hierarchical top-level-composition. Therefore an identification of component and data instances in the hierarchical software component structure or in the flat structure requires the correct iteration from the according `System` nodes.

##### 3.2.1.2.1 The ECU Flat Map

The file `Pprj_EcuExtract_U_FlatMap.arxml` contains the ECU Flat Map.

The ECU Flat Map is utilized to assign unique and comprehensible names to all `DataPrototype`s representing measurements and characteristics. This is important for the calibration engineers<sup>5</sup>

The applied strategy for the creation of a `FlatInstanceDescriptor.shortName` is to shorten it to the `shortName` of the `DataPrototype` when only a single instance of the `DataPrototype` is used.

---

<sup>5</sup>Calibration engineers in this context means the engineers working with measurement and calibration tooling e.g. to determine the correct calibration parameter values in order to adopt functionality in the software components to the mechanical components in the vehicle.

### 3.2.1.3 Data Types and Data Objects

The components are designed top down coming from the physical function down to the implementation in the target programming language C. Hereby the interfaces of Software Components are typically typed with `ApplicationDataTypes` in order to describe the physical meaning of the `DataPrototypes`. The only exceptions are the interfaces to AUTOSAR Services which are typed by `ImplementationDataTypes` directly as those are standardized. `ApplicationPrimitiveDataTypes` are mainly of `category`

- BOOLEAN
- VALUE
- CURVE
- MAP
- COM\_AXIS

and the most important CompuMethod `categorys` are

- LINEAR
- TEXTTABLE

In case of `LINEAR` conversions it is supported to differentiate the `Unit` used for the implemented calculations and an additional `Unit` used in the MCD system. This relationship of such `Units` are expressed with `UnitGroups`. The `ARElements` are structured in a way to support the common usage of elements relevant for the interface description up to the level of `PortInterfaces` by several Component Descriptions. Those elements are located under `Tier1/ARPlatform1/DataDictionary/<KindPackage>` in the file `Pprj_DataDictionary.arxml`.

The `CompuMethods` and `DataConstrs` are exclusively used by one `ApplicationPrimitiveDataType`. The possible reuse between `ApplicationPrimitiveDataTypes` supported by AUTOSAR is not used in this model structure. When such a `ApplicationDataType` is defined the intended mapping to the reasonable `ImplementationDataType` is already considered in order to get an optimal usage of the possible range of the `ImplementationDataType`. Nevertheless, the several physical meanings are **not** reflected by the definition of individual `ImplementationDataType` but only the standardized Platform Types [5] are used to describe primitives on implementation level. This has the effect that the RTE APIs are typed by the standardized Platform Types in cases of primitives and arrays of primitives. Only structure types are getting observable in the types of RTE APIs. This approach allows the direct usage of data read from RTE in mathematical or interpolation libraries without any type cast.

The memory allocation of the data objects is controlled by the usage of `SwAddrMethods`. Those are defined for `ParameterDataPrototypes` and `VariableDataPrototypes` on level of the `PortInterfaces`. A few examples are shown in

the chapter 3.2.2.17 for the basic uses cases like calibration parameter, normal data and code.

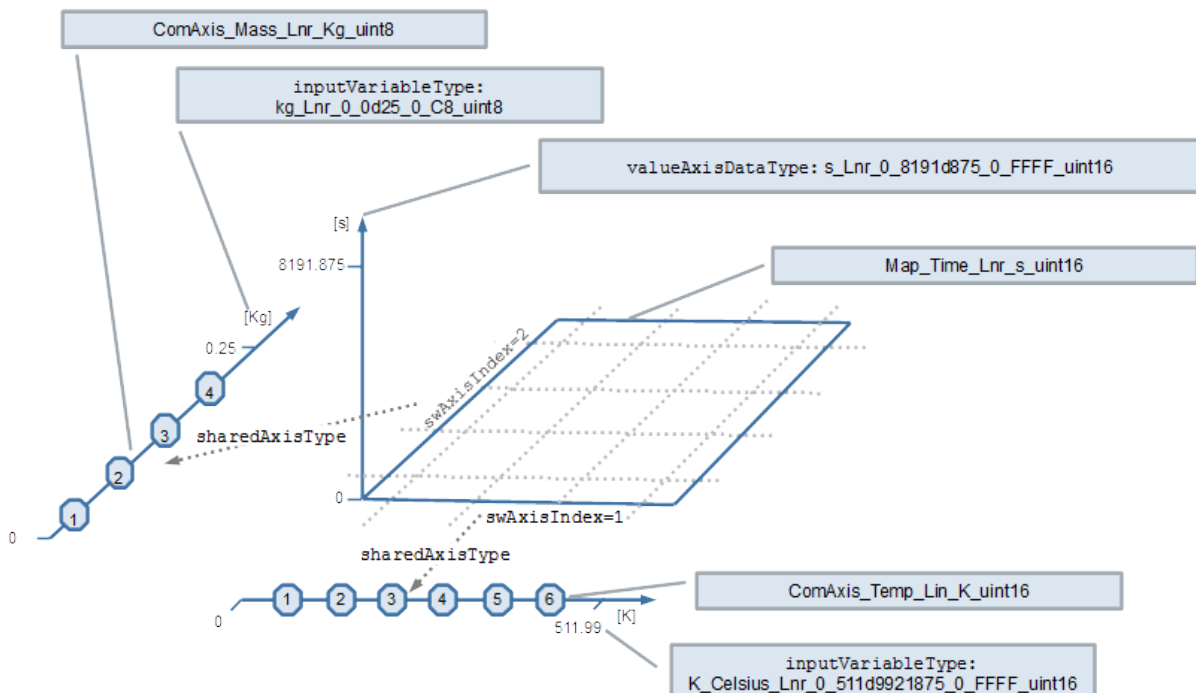
### 3.2.1.4 Axis, Curves and Maps

The show case contains description for axis, curves and maps which are in AUTOSAR so called compound primitives. In order to understand the structure and the defined attributes in the example it is helpful to understand how such objects are described in AUTOSAR. For this it is necessary to look at the hierarchy of [ApplicationDataTypes](#), [DataPrototypes](#), [PortPrototypes](#), [SwComponentTypes](#) and [FlatMap](#).

### 3.2.1.5 Axis, Curves and Maps on ApplicationDataType level

Figure 3.13 is based on the example of the [ApplicationPrimitiveDataType](#) `Map_Time_Lnr_s_uint16`. It shows the relationships between the [ApplicationPrimitiveDataTypes](#) describing the

- MAP itself
- its axis being a group axis
- in turn the properties of a matching working point



**Figure 3.13: ApplicationPrimitiveDataType of category MAP and its group axes**

The [ApplicationPrimitiveDataType](#) `Map_Time_Lnr_s_uint16` defines a data type for a MAP with group axes. The physical meaning and range of the contained values is described with the [ApplicationPrimitiveDataType](#) `s_Lnr_0_8191d875_0_FFFF_uint16`. It is referenced with the [valueAxisDataType](#) attribute. This means it's a value in the range 0 .. 8191.875 [second] with the resolution of 0.125 [second].

The referenced [ApplicationPrimitiveDataType](#) in the role [valueAxisDataType](#) represents the primitive data type of the value axis within a compound primitive (e.g. CURVE, MAP). It supersedes `CompuMethod`, `Unit`, and `BaseType`. In the particular example, the [valueAxisDataType](#) provides the properties of the primitive elements of the CURVE or MAP via a [valueAxisDataType](#) reference to an [ApplicationPrimitiveDataType](#). This in turn defines the attributes:

- [dataConstr](#)
- [compuMethod](#)
- [displayFormat](#)
- [unit](#)
- [swCalibrationAccess](#)

Thereby, despite being set, the value of [swCalibrationAccess](#) of the referenced [ApplicationPrimitiveDataType](#) is meaningless for the using CURVE and MAP. Note: The referenced data type needs to be a real primitive (typically of category VALUE. Category BOOLEAN is also supported).

The [ApplicationPrimitiveDataType](#) of the CURVE and MAP can additionally define [SwDataDefProps](#) which are relevant for the whole compound primitive. Currently the following attributes are used in the example:

- [swCalprmAxisSet](#)
- [swRecordLayout](#)
- [swCalibrationAccess](#) (but will be refined on [DataPrototype](#) level)

Further on, via the [dataTypeMapping](#) of the using software component, the properties of [ImplementationDataType](#) and [SwBaseType](#) are described.

As axes of the MAP two group axes are used. The properties of the group axes are described by two [ApplicationPrimitiveDataTypes](#) of category `COM_AXIS`. The attribute [swAxisIndex](#) indicates for which dimension the group axis applies (1 = X, 2 = Y). With the attribute [sharedAxisType](#) the reference to the [ApplicationPrimitiveDataType](#) describing the axis is defined.

In the example, the group axis `ComAxis_Temp_Lin_K_uint16` defines the applicable minimum and maximum number of axis points. Additionally the [inputVariableType](#) reference to the [ApplicationPrimitiveDataType](#) `K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16` defines the properties of the input

value for the axis. This in turn corresponds to the values stored as axis point. The same principle applies for the group axis `ComAxis_Mass_Lnr_Kg_uint8`.

Please note, the above mentioned properties are defined on the level of `ApplicationDataTypes` and so far not any data instance implementing such properties exists. This requires an instantiation of such `ApplicationDataTypes`.

### 3.2.1.6 Axis, Curves and Maps on DataPrototype and SwComponentPrototype level

#### 3.2.1.6.1 Instantiation of Axis, Curves and Maps

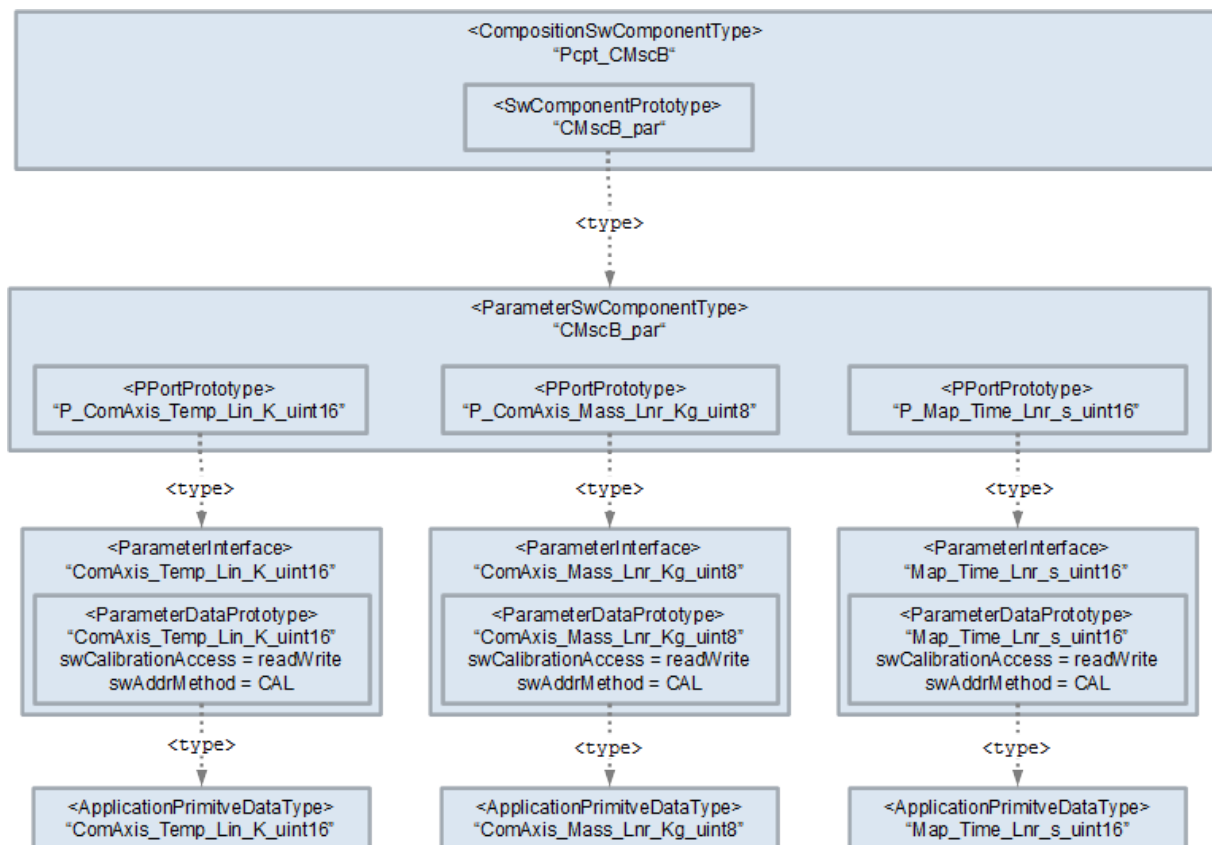


Figure 3.14: Instantiation of a MAP and its group axes

Figure 3.14 shows the instantiation of the `ApplicationPrimitiveDataType` `ComAxis_Temp_Lin_K_uint16`, `ComAxis_Mass_Lnr_Kg_uint8`, and `Map_Time_Lnr_s_uint16` up to the level of the `CompositionSwComponentType` `Pcpt_CMscB`.

Thereby `ParameterDataPrototypes` are typed by the mentioned `ApplicationPrimitiveDataTypes`. Each `ParameterDataPrototype` is owned by an own `ParameterInterface`. This offers the most flexibility to instantiate the map and axes independently from each other. On the level of the `ParameterDataPrototype` additionally the `swCalibrationAccess` and the `swAddrMethod` is defined.



Further on, the `ParameterSwComponentType CMscB_par` defines three `Port-Prototypes` typed by the `ParameterInterfaces`.

Please note that a group axes of a curve or map are not necessarily provided by the same `ParameterSwComponentType` as the one providing the curve or map. This case is illustrated with the map `ArrldMap_Time_Lnr_s_uint16` using the group axes `ArrldComAxis_Temp_Lin_K_uint16` provided by `CMscD_par` and `ComAxis_Mass_Lnr_Kg_uint8` provided by `CMscB_par`.

### 3.2.1.6.2 Usage of Axis, Curves and Maps by Software Components

#### 3.2.1.6.3 Linking map and curve instances to its axes instances

Consider a software component that uses curves and maps with group axes. It is then required to denote which instance of curve and map uses which instance of a group axis as axis of abscissae and, in case of a map, as axis of ordinate.

The AUTOSAR meta model provides hereby two possibilities:

- `RunnableEntity.parameterAccess.swDataDefProps`
- or
- `SwcInternalBehavior.instantiationDataDefProps.swDataDefProps`.

Inside one software component it's very unlikely, that the same curve or map is used with different axes by different `RunnableEntities` (note that this cannot be expressed by ASAM MCD-2MC, also) . Therefore, in this show case the second ability is used. This avoids the risk of inconsistencies when several `RunnableEntities` are defining `parameterAccesses` to the same curve or map instance.

The according `instantiationDataDefProps.parameterInstance` references the map instance in the scope of the `SwComponentType` and the `swDataDefProps.swCalprmAxisSet.swCalprmAxis.swCalprmAxisTypeProps.swCalprmRef` references the applied group axes with the according `SwCalprmAxis.swAxisIndex`

#### Listing 3.28: Example of an `InstantiationDataDefProps` for an map

```

<INSTANTIATION-DATA-DEF-PROPS>
  <PARAMETER-INSTANCE>
    <AUTOSAR-PARAMETER-IREF>
      <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/Tier1/ARplatform1/
        Pcpt_CMscB/CMscB/R_Map_Time_Lnr_s_uint16</PORT-PROTOTYPE-REF>
      <TARGET-DATA-PROTOTYPE-REF DEST="PARAMETER-DATA-PROTOTYPE">/Tier1
        /ARplatform1/DataDictionary/PortInterfaces/V1_0_0/
        Map_Time_Lnr_s_uint16/Map_Time_Lnr_s_uint16</TARGET-DATA-
        PROTOTYPE-REF>
    </AUTOSAR-PARAMETER-IREF>
  </PARAMETER-INSTANCE>
</SW-DATA-DEF-PROPS>
  <SW-DATA-DEF-PROPS-VARIANTS>

```

```

<SW-DATA-DEF-PROPS-CONDITIONAL>
  <SW-CALPRM-AXIS-SET>
    <SW-CALPRM-AXIS>
      <SW-AXIS-INDEX>1</SW-AXIS-INDEX>
      <SW-AXIS-GROUPED>
        <AR-PARAMETER>
          <AUTOSAR-PARAMETER-IREF>
            <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE"/>Tier1/
              ARPlatform1/Pcpt_CMscB/CMscB/
              R_ComAxis_Temp_Lin_K_uint16</PORT-PROTOTYPE-REF>
            <TARGET-DATA-PROTOTYPE-REF DEST="PARAMETER-DATA-
              PROTOTYPE"/>Tier1/ARPlatform1/DataDictionary/
              PortInterfaces/V1_0_0/ComAxis_Temp_Lin_K_uint16/
              ComAxis_Temp_Lin_K_uint16</TARGET-DATA-PROTOTYPE-
                REF>
          </AUTOSAR-PARAMETER-IREF>
        </AR-PARAMETER>
      </SW-AXIS-GROUPED>
    </SW-CALPRM-AXIS>
    <SW-CALPRM-AXIS>
      <SW-AXIS-INDEX>2</SW-AXIS-INDEX>
      <SW-AXIS-GROUPED>
        <AR-PARAMETER>
          <AUTOSAR-PARAMETER-IREF>
            <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE"/>Tier1/
              ARPlatform1/Pcpt_CMscB/CMscB/
              R_ComAxis_Mass_Lnr_Kg_uint8</PORT-PROTOTYPE-REF>
            <TARGET-DATA-PROTOTYPE-REF DEST="PARAMETER-DATA-
              PROTOTYPE"/>Tier1/ARPlatform1/DataDictionary/
              PortInterfaces/V1_0_0/ComAxis_Mass_Lnr_Kg_uint8/
              ComAxis_Mass_Lnr_Kg_uint8</TARGET-DATA-PROTOTYPE-
                REF>
          </AUTOSAR-PARAMETER-IREF>ARPlatform1
        </AR-PARAMETER>
      </SW-AXIS-GROUPED>
    </SW-CALPRM-AXIS>
  </SW-CALPRM-AXIS-SET>
</SW-DATA-DEF-PROPS-CONDITIONAL>
</SW-DATA-DEF-PROPS-VARIANTS>
</SW-DATA-DEF-PROPS>
</INSTANTIATION-DATA-DEF-PROPS>
    
```

#### 3.2.1.6.4 Linking axes instances to its working point instances

When a software component uses compound primitives containing axes (e.g. curves, maps, or group axes) it's beneficial to indicate which data is used as input for the according axis. This enables the measurement and calibration tool to display the current working point. Like explained in section 3.2.1.6.3, this information can be provided

- at the `ParameterAccess.swDataDefProps` of the compound primitives containing the axis **or**
- by means of `instantiationDataDefProps.swDataDefProps`.

In this show case the second ability is used for the same reasons as discussed in section 3.2.1.6.3.

The according `instantiationDataDefProps.parameterInstance` references the axes instance in the scope of the `SwComponentType CMscB`. The `swDataDefProps.swCalprmAxisSet.swCalprmAxis.swCalprmAxisTypeProps.swVariableRef` references the applied working point variable (in this case, a `dataElement` in a `RPortPrototype`) with the according `SwCalprmAxis.swAxisIndex`.

### Listing 3.29: Example of an `InstantiationDataDefProps` for an axis

```

<INSTANTIATION-DATA-DEF-PROPS>
  <PARAMETER-INSTANCE>
    <AUTOSAR-PARAMETER-IREF>
      <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE"/>Tier1/ARplatform1/
        Pcpt_CMscB/CMscB/R_ComAxis_Temp_Lin_K_uint16</PORT-PROTOTYPE-
        REF>
      <TARGET-DATA-PROTOTYPE-REF DEST="PARAMETER-DATA-PROTOTYPE"/>Tier1
        /ARplatform1/DataDictionary/PortInterfaces/V1_0_0/
        ComAxis_Temp_Lin_K_uint16/ComAxis_Temp_Lin_K_uint16</TARGET-
        DATA-PROTOTYPE-REF>
    </AUTOSAR-PARAMETER-IREF>
  </PARAMETER-INSTANCE>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <SW-CALPRM-AXIS-SET>
          <SW-CALPRM-AXIS>
            <SW-AXIS-INDEX>1</SW-AXIS-INDEX>
            <SW-AXIS-INDIVIDUAL>
              <SW-VARIABLE-REFS>
                <AUTOSAR-VARIABLE>
                  <AUTOSAR-VARIABLE-IREF>
                    <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE"/>Tier1/
                      ARplatform1/Pcpt_CMscB/CMscB/
                      R_PrimData_Temperature_Lin_K_C_uint16</PORT-
                      PROTOTYPE-REF>
                    <TARGET-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                      PROTOTYPE"/>Tier1/ARplatform1/DataDictionary/
                      PortInterfaces/V1_0_0/
                      PrimData_Temperature_Lin_K_C_uint16/
                      PrimData_Temperature_Lin_K_C_uint16</TARGET-
                      DATA-PROTOTYPE-REF>
                  </AUTOSAR-VARIABLE-IREF>
                </AUTOSAR-VARIABLE>
              </SW-VARIABLE-REFS>
            </SW-AXIS-INDIVIDUAL>
          </SW-CALPRM-AXIS>
        </SW-CALPRM-AXIS-SET>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
</INSTANTIATION-DATA-DEF-PROPS>
    
```

### 3.2.1.6.5 Axis, Curves and Maps in the ECU Flat Map

The ECU Flat Map contains entries for all curves, maps, axes and working point variables. The used naming patterns are described in [3.2.1.2.1](#).

**Listing 3.30: Example of a `FlatInstanceDescriptor` for map axis and working point variable**

```

<FLAT-INSTANCE-DESCRIPTOR>
  <SHORT-NAME>Map_Time_Lnr_s_uint16</SHORT-NAME>
  <ECU-EXTRACT-REFERENCE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/Tier1/
      ARPlatform1/System/SystemU_System/SystemU</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/Tier1/
      ARPlatform1/System/CompositionSwComponentTypes/SystemU_Root/
      CMscB_par</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE">/Tier1/ARPlatform1/
      Pcpt_CMscB/CMscB_par/P_Map_Time_Lnr_s_uint16</CONTEXT-ELEMENT-
      REF>
    <TARGET-REF DEST="PARAMETER-DATA-PROTOTYPE">/Tier1/ARPlatform1/
      DataDictionary/PortInterfaces/V1_0_0/Map_Time_Lnr_s_uint16/
      Map_Time_Lnr_s_uint16</TARGET-REF>
  </ECU-EXTRACT-REFERENCE-IREF>
</FLAT-INSTANCE-DESCRIPTOR>
<FLAT-INSTANCE-DESCRIPTOR>
  <SHORT-NAME>ComAxis_Temp_Lin_K_uint16</SHORT-NAME>
  <ECU-EXTRACT-REFERENCE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/Tier1/
      ARPlatform1/System/SystemU_System/SystemU</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/Tier1/
      ARPlatform1/System/CompositionSwComponentTypes/SystemU_Root/
      CMscB_par</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE">/Tier1/ARPlatform1/
      Pcpt_CMscB/CMscB_par/P_ComAxis_Temp_Lin_K_uint16</CONTEXT-
      ELEMENT-REF>
    <TARGET-REF DEST="PARAMETER-DATA-PROTOTYPE">/Tier1/ARPlatform1/
      DataDictionary/PortInterfaces/V1_0_0/ComAxis_Temp_Lin_K_uint16/
      ComAxis_Temp_Lin_K_uint16</TARGET-REF>
  </ECU-EXTRACT-REFERENCE-IREF>
</FLAT-INSTANCE-DESCRIPTOR>
<FLAT-INSTANCE-DESCRIPTOR>
  <SHORT-NAME>PrimData_Temperature_Lin_K_C_uint16</SHORT-NAME>
  <ECU-EXTRACT-REFERENCE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/Tier1/
      ARPlatform1/System/SystemU_System/SystemU</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/Tier1/
      ARPlatform1/System/CompositionSwComponentTypes/SystemU_Root/
      CMscA</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE">/Tier1/ARPlatform1/
      Pcpt_CMscA/CMscA/P_PrimData_Temperature_Lin_K_C_uint16</CONTEXT
      -ELEMENT-REF>
    <TARGET-REF DEST="VARIABLE-DATA-PROTOTYPE">/Tier1/ARPlatform1/
      DataDictionary/PortInterfaces/V1_0_0/
      PrimData_Temperature_Lin_K_C_uint16/
      PrimData_Temperature_Lin_K_C_uint16</TARGET-REF>
  </ECU-EXTRACT-REFERENCE-IREF>

```

</FLAT-INSTANCE-DESCRIPTOR>

### 3.2.1.7 Arrays of Maps and Axes

The ability of curves, maps and cuboids is usually used to describe the physical dependency of a characteristic on other physical input values. Hereby each input value is described by an orthogonal axis. In contrast to this, arrays are used to group a set of values of the same nature which can be handled by the same algorithm. Typically, in this case the algorithm iterates over the array with an index. Nevertheless, each array element may represent a particular part of the vehicle, e.g. a specific cylinder or a specific sensor. It's possible to combine these design principles. This ends up in the need to describe arrays of curves, maps, cuboids and the according axes.

The show case illustrates the model of those objects by the following elements:

- [ArrldMap\\_Time\\_Lnr\\_s\\_uint16](#)
- [ArrldComAxis\\_Temp\\_Lin\\_K\\_uint16](#)
- [ArrldPrimData\\_Temperature\\_Lin\\_K\\_C\\_uint16](#)

Hereby, the array of the map [ArrldMap\\_Time\\_Lnr\\_s\\_uint16](#) uses for the x-axis an array of group axes [ArrldComAxis\\_Temp\\_Lin\\_K\\_uint16](#) which in turn uses an array of primitive values as working points [ArrldPrimData\\_Temperature\\_Lin\\_K\\_C\\_uint16](#). In this case, the n'th map uses the n'th x-axes which uses the n'th value as working point. In contrast, the map uses one group axis [ComAxis\\_Mass\\_Lnr\\_Kg\\_uint8](#) for the y-axis. In this case all maps in the array are using the same y-axis.

#### 3.2.1.7.1 Arrays of Maps and Axes in the ECU Flat Map

In the ECU Flat Map the ability to reference [ApplicationCompositeElementDataPrototypes](#) is used to express the specific meaning of each array element in the array of map and group axis.

For instance, each element in the array [ArrldMap\\_Time\\_Lnr\\_s\\_uint16](#) is named in a way to indicate the specific meaning:

- [ArrldMap\\_Time\\_Lnr\\_s\\_uint16\\_FrontLeft](#)
- [ArrldMap\\_Time\\_Lnr\\_s\\_uint16\\_FrontRight](#)
- [ArrldMap\\_Time\\_Lnr\\_s\\_uint16\\_RearLeft](#)
- [ArrldMap\\_Time\\_Lnr\\_s\\_uint16\\_RearRight](#)

The following listing shows the structure of such an FlatInstanceDescriptor on one example:

**Listing 3.31: Example of a `FlatInstanceDescriptor` for an `ApplicationCompositeElementDataPrototype`**

```

<FLAT-INSTANCE-DESCRIPTOR>
  <SHORT-NAME>ArrlDMap_Time_Lnr_s_uint16_FrontLeft</SHORT-NAME>
  <ECU-EXTRACT-REFERENCE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE"/>Tier1/
      ARPlatform1/System/SystemU_System/SystemU</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE"/>Tier1/
      ARPlatform1/System/CompositionSwComponentTypes/SystemU_Root/
      CMscD_par</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE"/>Tier1/ARPlatform1/
      Pcpt_CMscD/CMscD_par/P_ArrlDMap_Time_Lnr_s_uint16</CONTEXT-
      ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="PARAMETER-DATA-PROTOTYPE"/>Tier1/
      ARPlatform1/DataDictionary/PortInterfaces/V1_0_0/
      ArrlDMap_Time_Lnr_s_uint16/ArrlDMap_Time_Lnr_s_uint16</CONTEXT-
      ELEMENT-REF>
    <TARGET-REF DEST="APPLICATION-ARRAY-ELEMENT" INDEX="0"/>Tier1/
      ARPlatform1/DataDictionary/ApplicationDataTypes/
      Map_Time_Lnr_s_uint16_ScNoOfWheels/
      Map_Time_Lnr_s_uint16_ScNoOfWheels</TARGET-REF>
  </ECU-EXTRACT-REFERENCE-IREF>
</FLAT-INSTANCE-DESCRIPTOR>

```

Please note the usage of the `index` attribute in the `target` reference.

**3.2.1.8 Measurement of Modes****3.2.1.8.1 Enabling Measurement of Modes**

The measurement of a mode is enabled in the software-component description by setting the `ModeDeclarationGroupPrototype.swCalibrationAccess` to `readOnly`. See `ModeDirection`.

**3.2.1.8.2 Modes in the ECU Flat Map**

AUTOSAR supports the measurement of the current mode, the previous mode and the next mode. Hereby the last two are useful when the mode is measured during an ongoing transition to identify the kind of transition. In this show case only the measurement of the current mode is illustrated. For this, the `FlatMap` contains a `FlatInstanceDescriptor` pointing to the `ModeDeclarationGroupPrototype` which is to be measured. The `role` attribute of the `FlatInstanceDescriptor` is set to `CURRENT_MODE`

**Listing 3.32: Example of a `FlatInstanceDescriptor` for a `ModeDeclarationGroupPrototype`**

```

<FLAT-INSTANCE-DESCRIPTOR>
  <SHORT-NAME>ModeDirection</SHORT-NAME>
  <ROLE>CURRENT_MODE</ROLE>

```

```

<ECU-EXTRACT-REFERENCE-IREF>
  <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE"/>Tier1/
    ARPlatform1/System/SystemU_System/SystemU</CONTEXT-ELEMENT-REF>
  <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE"/>Tier1/
    ARPlatform1/System/CompositionSwComponentTypes/SystemU_Root/
    CMscA</CONTEXT-ELEMENT-REF>
  <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE"/>Tier1/ARPlatform1/
    Pcpt_CMscA/CMscA/P_ModeDirection</CONTEXT-ELEMENT-REF>
  <TARGET-REF DEST="MODE-DECLARATION-GROUP-PROTOTYPE"/>Tier1/
    ARPlatform1/DataDictionary/PortInterfaces/V1_0_0/ModeDirection/
    ModeDirection</TARGET-REF>
</ECU-EXTRACT-REFERENCE-IREF>
</FLAT-INSTANCE-DESCRIPTOR>

```

## **3.2.2 Show cases in the Example**

### **3.2.2.1 CompositionSwComponentTypes**



Common <code>CompositionSwComponentType</code> attributes	
<code>shortName</code>	Pcpt_CMscA
<code>desc</code>	Modeling show case for primitive measurement and calculation.
properties of the <code>ports</code>	
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_ModeDirection
<code>desc</code>	Mode to indicate a direction
<code>providedInterface</code>	ModeDirection
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimCal_Mass_Lnr_Kg
<code>desc</code>	Primitive calibration parameter for minimum egg mass.
<code>providedInterface</code>	PrimCal_Mass_Lnr_Kg
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimData_Mass_Lnr_Kg_uint8
<code>desc</code>	Mass in kilogram
<code>providedInterface</code>	PrimData_Mass_Lnr_Kg_uint8
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimData_StepsSpeed_Txt_sint8
<code>desc</code>	Stepwise speed indication
<code>providedInterface</code>	PrimData_StepsSpeed_Txt_sint8
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimData_Temperature_Lin_K_C_uint16
<code>desc</code>	Temperature 1 in Kelvin but displayed as degree Celsius
<code>providedInterface</code>	PrimData_Temperature_Lin_K_C_uint16
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_PrimData_StepsSpeed_Txt_sint8
<code>desc</code>	Stepwise speed indication
<code>requiredInterface</code>	PrimData_StepsSpeed_Txt_sint8
properties of the <code>components</code>	
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscA
<code>type</code>	CMscA
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscA_par
<code>type</code>	CMscA_par




**Table 3.53: CompositionSwComponentType Pcpt\_CMscA**

Common <code>CompositionSwComponentType</code> attributes	
<code>shortName</code>	SystemU_Root
properties of the <code>components</code>	
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscA
<code>type</code>	CMscA
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscA_par
<code>type</code>	CMscA_par
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscB
<code>type</code>	CMscB
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscB_par
<code>type</code>	CMscB_par
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscC_nvm
<code>type</code>	CMscC_nvm
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscD
<code>type</code>	CMscD
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscD_par
<code>type</code>	CMscD_par

**Table 3.54: CompositionSwComponentType SystemU\_Root**

Common <code>CompositionSwComponentType</code> attributes	
<code>shortName</code>	SystemURootComposition_EcuDescr
properties of the <code>components</code>	
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscC
<code>type</code>	Pcpt_CMscC

**Table 3.55: CompositionSwComponentType SystemURootComposition\_EcuDescr**

Common <code>CompositionSwComponentType</code> attributes	
<code>shortName</code>	Pcpt_CMscD
<code>desc</code>	Modeling show case for arrays of axes and mapes.
properties of the <code>ports</code>	
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_ComAxis_Mass_Lnr_Kg_uint8
<code>desc</code>	Shared axis for mass
<code>requiredInterface</code>	ComAxis_Mass_Lnr_Kg_uint8
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_PrimData_MassCorrected_Lnr_Kg_uint8
<code>desc</code>	Primitve data for the corrected mass in kg.
<code>requiredInterface</code>	PrimData_MassCorrected_Lnr_Kg_uint8
properties of the <code>components</code>	
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscD
<code>type</code>	CMscD
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscD_par
<code>type</code>	CMscD_par

**Table 3.56: CompositionSwComponentType Pcpt\_CMscD**

Common <code>CompositionSwComponentType</code> attributes	
<code>shortName</code>	Pcpt_CMscC
<code>desc</code>	Composit of modeling show case C
properties of the <code>ports</code>	
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimData_Time_Lnr_s_uint16
<code>desc</code>	Primitve data holding a time value.
<code>providedInterface</code>	PrimData_Time_Lnr_s_uint16
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimData_ValidState_Txt_noUnit_boolean
<code>desc</code>	Boolean representing the data validity
<code>providedInterface</code>	PrimData_ValidState_Txt_noUnit_boolean
properties of the <code>components</code>	
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscA
<code>type</code>	Pcpt_CMscA
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscB
<code>type</code>	Pcpt_CMscB
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscC_nvm
<code>type</code>	CMscC_nvm
properties of <code>SwComponentPrototype</code>	
<code>shortName</code>	CMscD
<code>type</code>	Pcpt_CMscD

**Table 3.57: CompositionSwComponentType Pcpt\_CMscC**

Common <code>CompositionSwComponentType</code> attributes	
<code>shortName</code>	Pcpt_CMscB
<code>desc</code>	Modeling show case for axes, curves and mapes.
properties of the ports	
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_ComAxis_Mass_Lnr_Kg_uint8
<code>desc</code>	Shared axis for mass
<code>providedInterface</code>	ComAxis_Mass_Lnr_Kg_uint8
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimData_MassCorrected_Lnr_Kg_uint8
<code>desc</code>	Primitve data for the corrected mass in kg.
<code>providedInterface</code>	PrimData_MassCorrected_Lnr_Kg_uint8
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimData_Time_Lnr_s_uint16
<code>desc</code>	Primitve data holding a time value.
<code>providedInterface</code>	PrimData_Time_Lnr_s_uint16
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimData_ValidState_Txt_noUnit_boolean
<code>desc</code>	Boolean representing the data validity
<code>providedInterface</code>	PrimData_ValidState_Txt_noUnit_boolean
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_ModeDirection
<code>desc</code>	Mode to indicate a direction
<code>requiredInterface</code>	ModeDirection
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_PrimData_Mass_Lnr_Kg_uint8
<code>desc</code>	Mass in kilogram
<code>requiredInterface</code>	PrimData_Mass_Lnr_Kg_uint8
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_PrimData_StepsSpeed_Txt_sint8
<code>desc</code>	Stepwise speed indication
<code>requiredInterface</code>	PrimData_StepsSpeed_Txt_sint8
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_PrimData_Temperature_Lin_K_C_uint16
<code>desc</code>	Temperature 1 in Kelvin but displayed as degree Celsius
<code>requiredInterface</code>	PrimData_Temperature_Lin_K_C_uint16





properties of the <b>components</b>	
<b>properties of SwComponentPrototype</b>	
<b>shortName</b>	CMscB
<b>type</b>	CMscB
<b>properties of SwComponentPrototype</b>	
<b>shortName</b>	CMscB_par
<b>type</b>	CMscB_par

**Table 3.58: CompositionSwComponentType Pcpt\_CMscB**

### 3.2.2.2 ParameterSwComponentTypes

Common <code>ParameterSwComponentType</code> attributes	
<code>shortName</code>	CMscA_par
<code>desc</code>	Modeling show case for primitive measurement and calculation.
properties of the ports	
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimCal_Mass_Lnr_Kg
<code>desc</code>	Primitive calibration parameter for minimum egg mass.
<code>providedInterface</code>	PrimCal_Mass_Lnr_Kg

**Table 3.59: ParameterSwComponentType CMscA\_par**

Common <code>ParameterSwComponentType</code> attributes	
<code>shortName</code>	CMscD_par
<code>desc</code>	Modeling show case for arrays of axes and maps.
properties of the ports	
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_Arr1dComAxis_Temp_Lin_K_uint16
<code>desc</code>	Array of shared axis for temperature
<code>providedInterface</code>	Arr1dComAxis_Temp_Lin_K_uint16
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_Arr1dMap_Time_Lnr_s_uint16
<code>desc</code>	Map to get time dependent on temperature and mass.
<code>providedInterface</code>	Arr1dMap_Time_Lnr_s_uint16

**Table 3.60: ParameterSwComponentType CMscD\_par**

Common <code>ParameterSwComponentType</code> attributes	
<code>shortName</code>	CMscB_par
<code>desc</code>	Modeling show case for axes, curves and mapes.
properties of the <code>ports</code>	
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_ComAxis_Mass_Lnr_Kg_uint8
<code>desc</code>	Shared axis for mass
<code>providedInterface</code>	ComAxis_Mass_Lnr_Kg_uint8
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_ComAxis_Steps_Txt_sint8
<code>desc</code>	Shared axis for speed steps
<code>providedInterface</code>	ComAxis_Steps_Txt_sint8
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_ComAxis_Temp_Lin_K_uint16
<code>desc</code>	Shared axis for temperature
<code>providedInterface</code>	ComAxis_Temp_Lin_K_uint16
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_Curve_Mass_Lnr_Kg_uint8
<code>desc</code>	Curve to get mass according differnt speed steps.
<code>providedInterface</code>	Curve_Mass_Lnr_Kg_uint8
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_Map_Time_Lnr_s_uint16
<code>desc</code>	Map to get time dependent on temperature and mass.
<code>providedInterface</code>	Map_Time_Lnr_s_uint16

**Table 3.61: ParameterSwComponentType CMscB\_par**



### 3.2.2.3 ApplicationSwComponentTypes

Common <code>ApplicationSwComponentType</code> attributes	
<code>shortName</code>	CMscD
<code>desc</code>	Modeling show case for arrays of axes and mapes.
<b>properties of the ports</b>	
<b>properties of <code>PPortPrototype</code></b>	
<code>shortName</code>	P_Arr1dPrimData_Temperature_Lin_K_C_uint16
<code>desc</code>	Temperature 1 in Kelvin but displayed as degree Celsius
<code>providedInterface</code>	Arr1dPrimData_Temperature_Lin_K_C_uint16
<b>properties of <code>RPortPrototype</code></b>	
<code>shortName</code>	R_Arr1dComAxis_Temp_Lin_K_uint16
<code>desc</code>	Array of shared axis for temperature
<code>requiredInterface</code>	Arr1dComAxis_Temp_Lin_K_uint16
<b>properties of <code>RPortPrototype</code></b>	
<code>shortName</code>	R_Arr1dMap_Time_Lnr_s_uint16
<code>desc</code>	Map to get time dependent on temperature and mass.
<code>requiredInterface</code>	Arr1dMap_Time_Lnr_s_uint16
<b>properties of <code>RPortPrototype</code></b>	
<code>shortName</code>	R_Arr1dPrimData_Temperature_Lin_K_C_uint16
<code>desc</code>	Temperature 1 in Kelvin but displayed as degree Celsius
<code>requiredInterface</code>	Arr1dPrimData_Temperature_Lin_K_C_uint16
<b>properties of <code>RPortPrototype</code></b>	
<code>shortName</code>	R_ComAxis_Mass_Lnr_Kg_uint8
<code>desc</code>	Shared axis for mass
<code>requiredInterface</code>	ComAxis_Mass_Lnr_Kg_uint8
<b>properties of <code>RPortPrototype</code></b>	
<code>shortName</code>	R_PrimData_MassCorrected_Lnr_Kg_uint8
<code>desc</code>	Primitve data for the corrected mass in kg.
<code>requiredInterface</code>	PrimData_MassCorrected_Lnr_Kg_uint8
<code>internalBehavior</code>	CMscD

**Table 3.62: ApplicationSwComponentType CMscD**

Common <code>SwcInternalBehavior</code> attributes	
<code>shortName</code>	CMscD
properties of the <code>runnables</code>	
properties of <code>RunnableEntity</code>	
<code>shortName</code>	CMscD_Process
<code>symbol</code>	CMscD_Process

**Table 3.63: SwcInternalBehavior CMscD**

Common <code>ApplicationSwComponentType</code> attributes	
<code>shortName</code>	CMscB
<code>desc</code>	Modeling show case for axes, curves and mapes.
properties of the ports	
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimData_MassCorrected_Lnr_Kg_uint8
<code>desc</code>	Primitve data for the corrected mass in kg.
<code>providedInterface</code>	<code>PrimData_MassCorrected_Lnr_Kg_uint8</code>
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimData_Time_Lnr_s_uint16
<code>desc</code>	Primitve data holding a time value.
<code>providedInterface</code>	<code>PrimData_Time_Lnr_s_uint16</code>
properties of <code>PPortPrototype</code>	
<code>shortName</code>	P_PrimData_ValidState_Txt_noUnit_boolean
<code>desc</code>	Boolean representing the data validity
<code>providedInterface</code>	<code>PrimData_ValidState_Txt_noUnit_boolean</code>
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_ComAxis_Mass_Lnr_Kg_uint8
<code>desc</code>	Shared axis for mass
<code>requiredInterface</code>	<code>ComAxis_Mass_Lnr_Kg_uint8</code>
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_ComAxis_Steps_Txt_sint8
<code>desc</code>	Shared axis for speed steps
<code>requiredInterface</code>	<code>ComAxis_Steps_Txt_sint8</code>
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_ComAxis_Temp_Lin_K_uint16
<code>desc</code>	Shared axis for temperature
<code>requiredInterface</code>	<code>ComAxis_Temp_Lin_K_uint16</code>
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_Curve_Mass_Lnr_Kg_uint8
<code>desc</code>	Curve to get mass according differnt speed steps.
<code>requiredInterface</code>	<code>Curve_Mass_Lnr_Kg_uint8</code>
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_Map_Time_Lnr_s_uint16
<code>desc</code>	Map to get time dependent on temperature and mass.
<code>requiredInterface</code>	<code>Map_Time_Lnr_s_uint16</code>
properties of <code>RPortPrototype</code>	
<code>shortName</code>	R_ModeDirection





<b>desc</b>	Mode to indicate a direction
<b>requiredInterface</b>	ModeDirection
<b>properties of RPortPrototype</b>	
<b>shortName</b>	R_PrimData_Mass_Lnr_Kg_uint8
<b>desc</b>	Mass in kilogram
<b>requiredInterface</b>	PrimData_Mass_Lnr_Kg_uint8
<b>properties of RPortPrototype</b>	
<b>shortName</b>	R_PrimData_MassCorrected_Lnr_Kg_uint8
<b>desc</b>	Primitve data for the corrected mass in kg.
<b>requiredInterface</b>	PrimData_MassCorrected_Lnr_Kg_uint8
<b>properties of RPortPrototype</b>	
<b>shortName</b>	R_PrimData_StepsSpeed_Txt_sint8
<b>desc</b>	Stepwise speed indication
<b>requiredInterface</b>	PrimData_StepsSpeed_Txt_sint8
<b>properties of RPortPrototype</b>	
<b>shortName</b>	R_PrimData_Temperature_Lin_K_C_uint16
<b>desc</b>	Temperature 1 in Kelvin but displayed as degree Celsius
<b>requiredInterface</b>	PrimData_Temperature_Lin_K_C_uint16
<b>properties of RPortPrototype</b>	
<b>shortName</b>	R_PrimData_Time_Lnr_s_uint16
<b>desc</b>	Primitve data holding a time value.
<b>requiredInterface</b>	PrimData_Time_Lnr_s_uint16
<b>properties of RPortPrototype</b>	
<b>shortName</b>	R_PrimData_ValidState_Txt_noUnit_boolean
<b>desc</b>	Boolean representing the data validity
<b>requiredInterface</b>	PrimData_ValidState_Txt_noUnit_boolean
<b>internalBehavior</b>	CMscB

**Table 3.64: ApplicationSwComponentType CMscB**

<b>Common SwcInternalBehavior attributes</b>	
<b>shortName</b>	CMscB
<b>properties of the runnables</b>	
<b>properties of RunnableEntity</b>	
<b>shortName</b>	CMscB_Process
<b>desc</b>	cyclic process for calculation
<b>symbol</b>	CMscB_Process

**Table 3.65: SwcInternalBehavior CMscB**

### 3.2.2.4 ParameterInterfaces

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	Arr1dComAxis_Temp_Lin_K_uint16
<code>desc</code>	Array of shared axis for temperature
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	Arr1dComAxis_Temp_Lin_K_uint16
<code>desc</code>	Array of shared axis for temperature
<code>type</code>	<code>ComAxis_Temp_Lin_K_uint16_ScNoOfWheels</code>
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CAL

**Table 3.66: ParameterInterface Arr1dComAxis\_Temp\_Lin\_K\_uint16**

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	Arr1dMap_Time_Lnr_s_uint16
<code>desc</code>	Map to get time dependent on temperature and mass.
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	Arr1dMap_Time_Lnr_s_uint16
<code>desc</code>	Map to get time dependent on temperature and mass.
<code>type</code>	<code>Map_Time_Lnr_s_uint16_ScNoOfWheels</code>
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CAL

**Table 3.67: ParameterInterface Arr1dMap\_Time\_Lnr\_s\_uint16**

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	ComAxis_Mass_Lnr_Kg_uint8
<code>desc</code>	Shared axis for mass
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	ComAxis_Mass_Lnr_Kg_uint8
<code>desc</code>	Shared axis for mass
<code>type</code>	ComAxis_Mass_Lnr_Kg_uint8
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CAL

**Table 3.68: ParameterInterface ComAxis\_Mass\_Lnr\_Kg\_uint8**

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	ComAxis_Steps_Txt_sint8
<code>desc</code>	Shared axis for speed steps
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	ComAxis_Steps_Txt_sint8
<code>desc</code>	Shared axis for speed steps
<code>type</code>	ComAxis_Steps_Txt_sint8
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CAL

**Table 3.69: ParameterInterface ComAxis\_Steps\_Txt\_sint8**

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	ComAxis_Temp_Lin_K_uint16
<code>desc</code>	Shared axis for temperature
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	ComAxis_Temp_Lin_K_uint16
<code>desc</code>	Shared axis for temperature
<code>type</code>	ComAxis_Temp_Lin_K_uint16
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CAL

**Table 3.70: ParameterInterface ComAxis\_Temp\_Lin\_K\_uint16**

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	Curve_Mass_Lnr_Kg_uint8
<code>desc</code>	Curve to get mass according differnt speed steps.
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	Curve_Mass_Lnr_Kg_uint8
<code>desc</code>	Curve to get mass according differnt speed steps.
<code>type</code>	Curve_Mass_Lnr_Kg_uint8
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CAL

**Table 3.71: ParameterInterface Curve\_Mass\_Lnr\_Kg\_uint8**

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	Map_Time_Lnr_s_uint16
<code>desc</code>	Map to get time dependent on temperature and mass.
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	Map_Time_Lnr_s_uint16
<code>desc</code>	Map to get time dependent on temperature and mass.
<code>type</code>	Map_Time_Lnr_s_uint16
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CAL

**Table 3.72: ParameterInterface Map\_Time\_Lnr\_s\_uint16**

Common <code>ParameterInterface</code> attributes	
<code>shortName</code>	PrimCal_Mass_Lnr_Kg
<code>desc</code>	Primitive calibration parameter for minimum egg mass.
properties of the <code>parameters</code>	
properties of <code>ParameterDataPrototype</code>	
<code>shortName</code>	PrimCal_Mass_Lnr_Kg
<code>desc</code>	Primitive calibration parameter for minimum egg mass.
<code>type</code>	kg_Lnr_0_0d25_0_C8_uint8
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readWrite
<code>swAddrMethod</code>	CAL

**Table 3.73: ParameterInterface PrimCal\_Mass\_Lnr\_Kg**

### 3.2.2.5 ModeSwitchInterfaces

Common <code>ModeSwitchInterface</code> attributes	
<code>shortName</code>	ModeDirection
<code>desc</code>	Mode to indicate a direction
properties of the <code>modeGroups</code>	
<code>shortName</code>	ModeDirection
<code>swCalibrationAccess</code>	<code>readOnly</code>
<code>type</code>	<code>Direction</code>

**Table 3.74: ModeSwitchInterface ModeDirection**



### 3.2.2.6 SenderReceiverInterfaces

Common <code>SenderReceiverInterface</code> attributes	
<code>shortName</code>	Arr1dPrimData_Temperature_Lin_K_C_uint16
<code>desc</code>	Temperature 1 in Kelvin but displayed as degree Celsius
properties of the dataElementss	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	Arr1dPrimData_Temperature_Lin_K_C_uint16
<code>desc</code>	Temperature 1 in Kelvin but displayed as degree Celsius
<code>type</code>	K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16_- ScNoOfWheels
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	DATA

**Table 3.75: SenderReceiverInterface Arr1dPrimData\_Temperature\_Lin\_K\_C\_uint16**

Common <code>SenderReceiverInterface</code> attributes	
<code>shortName</code>	PrimData_Mass_Lnr_Kg_uint8
<code>desc</code>	Mass in kilogram
properties of the dataElementss	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	PrimData_Mass_Lnr_Kg_uint8
<code>desc</code>	Mass in kilogram
<code>type</code>	kg_Lnr_0_0d25_0_C8_uint8
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	DATA

**Table 3.76: SenderReceiverInterface PrimData\_Mass\_Lnr\_Kg\_uint8**

Common <code>SenderReceiverInterface</code> attributes	
<code>shortName</code>	PrimData_MassCorrected_Lnr_Kg_uint8
<code>desc</code>	Primitve data for the corrected mass in kg.
properties of the <code>dataElementss</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	PrimData_MassCorrected_Lnr_Kg_uint8
<code>desc</code>	Primitve data for the corrected mass in kg.
<code>type</code>	<code>kg_Lnr_0_0d25_0_C8_uint8</code>
<code>swImplPolicy</code>	<code>standard</code>
<code>swCalibrationAccess</code>	<code>readOnly</code>
<code>swAddrMethod</code>	<code>DATA</code>

**Table 3.77: SenderReceiverInterface PrimData\_MassCorrected\_Lnr\_Kg\_uint8**

Common <code>SenderReceiverInterface</code> attributes	
<code>shortName</code>	PrimData_Temperature_Lin_K_C_uint16
<code>desc</code>	Temperature 1 in Kelvin but displayed as degree Celsius
properties of the <code>dataElementss</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	PrimData_Temperature_Lin_K_C_uint16
<code>desc</code>	Temperature 1 in Kelvin but displayed as degree Celsius
<code>type</code>	<code>K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16</code>
<code>swImplPolicy</code>	<code>standard</code>
<code>swCalibrationAccess</code>	<code>readOnly</code>
<code>swAddrMethod</code>	<code>DATA</code>

**Table 3.78: SenderReceiverInterface PrimData\_Temperature\_Lin\_K\_C\_uint16**

Common <code>SenderReceiverInterface</code> attributes	
<code>shortName</code>	PrimData_Time_Lnr_s_uint16
<code>desc</code>	Primitve data holding a time value.
properties of the <code>dataElementss</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	PrimData_Time_Lnr_s_uint16
<code>desc</code>	Primitve data holding a time value.
<code>type</code>	<code>s_Lnr_0_8191d875_0_FFFF_uint16</code>
<code>swImplPolicy</code>	<code>standard</code>
<code>swCalibrationAccess</code>	<code>readOnly</code>
<code>swAddrMethod</code>	<code>DATA</code>

**Table 3.79: SenderReceiverInterface PrimData\_Time\_Lnr\_s\_uint16**

Common <code>SenderReceiverInterface</code> attributes	
<code>shortName</code>	PrimData_ValidState_Txt_noUnit_boolean
<code>desc</code>	Boolean representing the data validity
properties of the <code>dataElementss</code>	
properties of <code>VariableDataPrototype</code>	
<code>shortName</code>	PrimData_ValidState_Txt_noUnit_boolean
<code>desc</code>	Boolean representing the data validity
<code>type</code>	<code>DataValidityType</code>
<code>swImplPolicy</code>	standard
<code>swCalibrationAccess</code>	readOnly
<code>swAddrMethod</code>	DATA

**Table 3.80: SenderReceiverInterface PrimData\_ValidState\_Txt\_noUnit\_boolean**

### 3.2.2.7 ApplicationDataTypes, Category BOOLEAN

Common <code>ApplicationDataType</code> attributes				
<code>shortName</code>	DataValidityType			
<code>category</code>	BOOLEAN			
<code>desc</code>	Boolean to represent the data validity			
<code>swCalibrationAccess</code>	<code>notAccessible</code>			
<code>unit</code>	<code>NoUnit</code>			
Range				
	<code>lowerLimit</code>	<code>upperLimit</code>		
<code>physConstrs</code>	0	1		
Conversion				
<code>category</code>	TEXTTABLE			
<code>direction</code>	<code>compuInternalToPhys</code>			
<code>desc</code>	<code>lowerLimit</code>	<code>upperLimit</code>	<code>vt</code>	<code>symbol</code>
-	0	0	Invalid	
-	1	1	Valid	

**Table 3.81: ApplicationDataType DataValidityType**

### 3.2.2.8 ApplicationDataTypes, Category VALUE

Common <code>ApplicationDataType</code> attributes			
<code>shortName</code>	K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16		
<code>category</code>	VALUE		
<code>desc</code>	Temperature		
<code>swCalibrationAccess</code>	notAccessible		
<code>unit</code>	K		
Range			
	<code>lowerLimit</code>	<code>upperLimit</code>	
<code>physConstrs</code>	0	511.9921875	
Conversion			
<code>category</code>	LINEAR		
<code>direction</code>	compuInternalToPhys		
<code>desc</code>	<code>lowerLimit</code>	<code>upperLimit</code>	<code>compuNumerator / compuDenominator</code>
-	-	-	$Phys = \frac{0 + 0.0078125 * Internal}{1}$

Table 3.82: ApplicationDataType K\_Celsius\_Lnr\_0\_511d9921875\_0\_FFFF\_uint16

Common <code>ApplicationDataType</code> attributes			
<code>shortName</code>	kg_Lnr_0_0d25_0_C8_uint8		
<code>category</code>	VALUE		
<code>desc</code>	Mass		
<code>swCalibrationAccess</code>	notAccessible		
<code>unit</code>	kg		
Range			
	<code>lowerLimit</code>	<code>upperLimit</code>	
<code>physConstrs</code>	0	0.25	
Conversion			
<code>category</code>	LINEAR		
<code>direction</code>	compuInternalToPhys		
<code>desc</code>	<code>lowerLimit</code>	<code>upperLimit</code>	<code>compuNumerator / compuDenominator</code>
-	-	-	$Phys = \frac{0 + 0.00125 * Internal}{1}$

Table 3.83: ApplicationDataType kg\_Lnr\_0\_0d25\_0\_C8\_uint8

Common <code>ApplicationDataType</code> attributes			
<code>shortName</code>	NoUnit_Lnr_1_4_1_4_uint8		
<code>category</code>	VALUE		
<code>swCalibrationAccess</code>	notAccessible		
<code>unit</code>	NoUnit		
Range			
	<code>lowerLimit</code>	<code>upperLimit</code>	
<code>physConstrs</code>	1	4	
Conversion			
<code>category</code>	LINEAR		
<code>direction</code>	compuInternalToPhys		
<code>desc</code>	<code>lowerLimit</code>	<code>upperLimit</code>	<code>compuNumerator / compuDenominator</code>
-	-	-	$Phys = \frac{0 + 1 * Internal}{1}$

**Table 3.84: ApplicationDataType NoUnit\_Lnr\_1\_4\_1\_4\_uint8**

Common <code>ApplicationDataType</code> attributes			
<code>shortName</code>	NoUnit_Lnr_1_65535_1_FFFF_uint16		
<code>category</code>	VALUE		
<code>swCalibrationAccess</code>	notAccessible		
<code>unit</code>	NoUnit		
Range			
	<code>lowerLimit</code>	<code>upperLimit</code>	
<code>physConstrs</code>	1	65535	
Conversion			
<code>category</code>	LINEAR		
<code>direction</code>	compuInternalToPhys		
<code>desc</code>	<code>lowerLimit</code>	<code>upperLimit</code>	<code>compuNumerator / compuDenominator</code>
-	-	-	$Phys = \frac{0 + 1 * Internal}{1}$

**Table 3.85: ApplicationDataType NoUnit\_Lnr\_1\_65535\_1\_FFFF\_uint16**

Common <code>ApplicationDataType</code> attributes			
<code>shortName</code>	s_Lnr_0_8191d875_0_FFFF_uint16		
<code>category</code>	VALUE		
<code>desc</code>	cooking time in seconds		
<code>swCalibrationAccess</code>	notAccessible		
<code>unit</code>	s		
Range			
	<code>lowerLimit</code>	<code>upperLimit</code>	
<code>physConstrs</code>	0	8191.875	
Conversion			
<code>category</code>	LINEAR		
<code>direction</code>	compuInternalToPhys		
<code>desc</code>	<code>lowerLimit</code>	<code>upperLimit</code>	<code>compuNumerator / compuDenominator</code>
-	-	-	$Phys = \frac{0 + 0.125 * Internal}{1}$

Table 3.86: ApplicationDataType s\_Lnr\_0\_8191d875\_0\_FFFF\_uint16

Common <code>ApplicationDataType</code> attributes				
<code>shortName</code>	speedSteps			
<code>category</code>	VALUE			
<code>desc</code>	Possible speed steps			
<code>swCalibrationAccess</code>	notAccessible			
<code>unit</code>	NoUnit			
Range				
	<code>lowerLimit</code>	<code>upperLimit</code>		
<code>physConstrs</code>	-1	2		
Conversion				
<code>category</code>	TEXTTABLE			
<code>direction</code>	compuInternalToPhys			
<code>desc</code>	<code>lowerLimit</code>	<code>upperLimit</code>	<code>vt</code>	<code>symbol</code>
-	-1	-1	Stop	
-	0	0	LightSpeed	
-	1	1	RidiculousSpeed	
-	2	2	LudicrousSpeed	

Table 3.87: ApplicationDataType speedSteps

Common <code>ApplicationDataType</code> attributes				
<code>shortName</code>	TxWheelNames			
<code>category</code>	VALUE			
<code>desc</code>	Wheel names			
<code>swCalibrationAccess</code>	<code>notAccessible</code>			
<code>unit</code>	<code>NoUnit</code>			
Range				
	<code>lowerLimit</code>	<code>upperLimit</code>		
<code>physConstrs</code>	0	3		
Conversion				
<code>category</code>	TEXTTABLE			
<code>direction</code>	<code>compuInternalToPhys</code>			
<code>desc</code>	<code>lowerLimit</code>	<code>upperLimit</code>	<code>vt</code>	<code>symbol</code>
-	0	0	FrontLeft	
-	1	1	FrontRight	
-	2	2	RearLeft	
-	3	3	RearRight	

**Table 3.88: ApplicationDataType TxWheelNames**



### 3.2.2.9 ApplicationDataTypes, Category COM\_AXIS

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	ComAxis_Temp_Lin_K_uint16
<code>category</code>	COM_AXIS
<code>swCalibrationAccess</code>	<code>notAccessible</code>
<code>swRecordLayout</code>	RL20_ME_Axis
properties of the axes ( <code>swCalprmAxisSet</code> )	
properties of <code>SwAxisIndividual</code> ( <code>swCalprmAxis</code> and <code>swCalprmAxisTypeProps</code> )	
<code>swAxisIndex</code>	1
<code>category</code>	COM_AXIS
<code>inputVariableType</code>	<code>K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16</code>
<code>swMaxAxisPoints</code>	6
<code>swMinAxisPoints</code>	6

**Table 3.89: ApplicationDataType ComAxis\_Temp\_Lin\_K\_uint16**

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	ComAxis_Steps_Txt_sint8
<code>category</code>	COM_AXIS
<code>swCalibrationAccess</code>	<code>notAccessible</code>
<code>swRecordLayout</code>	RL20_ME_Axis
properties of the axes ( <code>swCalprmAxisSet</code> )	
properties of <code>SwAxisIndividual</code> ( <code>swCalprmAxis</code> and <code>swCalprmAxisTypeProps</code> )	
<code>swAxisIndex</code>	1
<code>category</code>	COM_AXIS
<code>inputVariableType</code>	<code>speedSteps</code>
<code>swMaxAxisPoints</code>	4
<code>swMinAxisPoints</code>	4

**Table 3.90: ApplicationDataType ComAxis\_Steps\_Txt\_sint8**

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	ComAxis_Mass_Lnr_Kg_uint8
<code>category</code>	COM_AXIS
<code>swCalibrationAccess</code>	notAccessible
<code>swRecordLayout</code>	RL20_ME_Axis
properties of the axes ( <code>swCalprmAxisSet</code> )	
properties of <code>SwAxisIndividual</code> ( <code>swCalprmAxis</code> and <code>swCalprmAxisTypeProps</code> )	
<code>swAxisIndex</code>	1
<code>category</code>	COM_AXIS
<code>inputVariableType</code>	kg_Lnr_0_0d25_0_C8_uint8
<code>swMaxAxisPoints</code>	4
<code>swMinAxisPoints</code>	4

**Table 3.91: ApplicationDataType ComAxis\_Mass\_Lnr\_Kg\_uint8**

### 3.2.2.10 ApplicationDataTypes, Category CURVE

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	Curve_Mass_Lnr_Kg_uint8
<code>category</code>	CURVE
<code>swCalibrationAccess</code>	notAccessible
<code>swRecordLayout</code>	RL20_ME_1DimMap
<code>valueAxisDataType</code>	kg_Lnr_0_0d25_0_C8_uint8
properties of the axes ( <code>swCalprmAxisSet</code> )	
properties of <code>SwAxisGrouped</code> ( <code>swCalprmAxis</code> and <code>swCalprmAxisTypeProps</code> )	
<code>swAxisIndex</code>	1
<code>category</code>	COM_AXIS
<code>sharedAxisType</code>	ComAxis_Steps_Txt_sint8

**Table 3.92: ApplicationDataType Curve\_Mass\_Lnr\_Kg\_uint8**

### 3.2.2.11 ApplicationDataTypes, Category MAP

Common <code>ApplicationDataType</code> attributes	
<code>shortName</code>	Map_Time_Lnr_s_uint16
<code>category</code>	MAP
<code>swCalibrationAccess</code>	notAccessible
<code>swRecordLayout</code>	RL20_ME_2DimMap
<code>valueAxisDataType</code>	s_Lnr_0_8191d875_0_FFFF_uint16
properties of the axes ( <code>swCalprmAxisSet</code> )	
properties of <code>SwAxisGrouped</code> ( <code>swCalprmAxis</code> and <code>swCalprmAxisTypeProps</code> )	
<code>swAxisIndex</code>	1
<code>category</code>	COM_AXIS
<code>sharedAxisType</code>	ComAxis_Temp_Lin_K_uint16
properties of <code>SwAxisGrouped</code> ( <code>swCalprmAxis</code> and <code>swCalprmAxisTypeProps</code> )	
<code>swAxisIndex</code>	2
<code>category</code>	COM_AXIS
<code>sharedAxisType</code>	ComAxis_Mass_Lnr_Kg_uint8

Table 3.93: ApplicationDataType Map\_Time\_Lnr\_s\_uint16

### 3.2.2.12 ApplicationArrayDataTypes

Common <code>ApplicationArrayDataType</code> attributes	
<code>shortName</code>	ComAxis_Temp_Lin_K_uint16_ScNoOfWheels
<code>category</code>	ARRAY
<code>swCalibrationAccess</code>	notAccessible
properties of the <code>elements</code>	
properties of <code>ApplicationArrayElement</code>	
<code>shortName</code>	ComAxis_Temp_Lin_K_uint16_ScNoOfWheels
<code>category</code>	COM_AXIS
<code>type</code>	ComAxis_Temp_Lin_K_uint16
<code>arraySizeSemantics</code>	fixedSize
<code>maxNumberOfElements</code>	

Table 3.94: ApplicationArrayDataType ComAxis\_Temp\_Lin\_K\_uint16\_ScNoOfWheels

Common <code>ApplicationArrayDataType</code> attributes	
<code>shortName</code>	K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16_ScNoOfWheels
<code>category</code>	ARRAY
<code>swCalibrationAccess</code>	notAccessible
properties of the <code>elements</code>	
properties of <code>ApplicationArrayElement</code>	
<code>shortName</code>	K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16_ScNoOfWheels
<code>category</code>	VALUE
<code>type</code>	K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16
<code>arraySizeSemantics</code>	fixedSize
<code>maxNumberOfElements</code>	

Table 3.95: ApplicationArrayDataType K\_Celsius\_Lnr\_0\_511d9921875\_0\_FFFF\_uint16\_ScNoOfWheels

Common <code>ApplicationArrayDataType</code> attributes	
<code>shortName</code>	Map_Time_Lnr_s_uint16_ScNoOfWheels
<code>category</code>	ARRAY
<code>swCalibrationAccess</code>	notAccessible
properties of the <code>elements</code>	
properties of <code>ApplicationArrayElement</code>	
<code>shortName</code>	Map_Time_Lnr_s_uint16_ScNoOfWheels
<code>category</code>	MAP
<code>type</code>	Map_Time_Lnr_s_uint16
<code>arraySizeSemantics</code>	fixedSize
<code>maxNumberOfElements</code>	

**Table 3.96: ApplicationArrayDataType Map\_Time\_Lnr\_s\_uint16\_ScNoOfWheels**

### 3.2.2.13 ApplicationRecordDataTypes

Common <code>ApplicationRecordDataType</code> attributes	
<code>shortName</code>	CMscC_nvm_NvBlockATyp
<code>category</code>	STRUCTURE
<code>swCalibrationAccess</code>	notAccessible
properties of the <code>elements</code>	
properties of <code>ApplicationRecordElement</code>	
<code>shortName</code>	PrimData_StepsSpeed_Txt_sint8
<code>category</code>	VALUE
<code>type</code>	speedSteps

**Table 3.97: ApplicationRecordDataType CMscC\_nvm\_NvBlockATyp**

### 3.2.2.14 ModeDeclarationGroups

Common <code>ModeDeclarationGroup</code> attributes	
<code>shortName</code>	Direction
<code>category</code>	EXPLICIT_ORDER
<code>initialMode</code>	Halt
properties of the <code>modeDeclarations</code>	
properties of <code>ModeDeclaration</code>	
<code>shortName</code>	Backward
<code>desc</code>	Backward direction
<code>value</code>	2
properties of <code>ModeDeclaration</code>	
<code>shortName</code>	Forward
<code>desc</code>	Forward direction
<code>value</code>	1
properties of <code>ModeDeclaration</code>	
<code>shortName</code>	Halt
<code>desc</code>	Standstill
<code>value</code>	0

**Table 3.98: ModeDeclarationGroup Direction**



### 3.2.2.15 Units

Common Unit attributes	
shortName	Celsius
desc	Degrees Celsius
displayName	°C
offsetSiToUnit	-273.15
factorSiToUnit	1.0
physicalDimension	PD_K

**Table 3.99: Unit Celsius**

Common Unit attributes	
shortName	K
desc	Temperature
displayName	K
offsetSiToUnit	0.0
factorSiToUnit	1.0
physicalDimension	PD_K

**Table 3.100: Unit K**

Common Unit attributes	
shortName	kg
desc	Mass
displayName	kg
offsetSiToUnit	0.0
factorSiToUnit	1.0
physicalDimension	PD_kg

**Table 3.101: Unit kg**

Common Unit attributes	
shortName	NoUnit
desc	No Unit
displayName	-
offsetSiToUnit	0.0
factorSiToUnit	1.0
physicalDimension	PD_NoUnit

**Table 3.102: Unit NoUnit**

Common Unit attributes	
shortName	s
desc	Time
displayName	s
offsetSiToUnit	0.0
factorSiToUnit	1.0
physicalDimension	PD_s

**Table 3.103: Unit s**

### 3.2.2.16 PhysicalDimensions

Common <code>PhysicalDimension</code> attributes	
<code>shortName</code>	PD_K
<code>currentExp</code>	0
<code>lengthExp</code>	0
<code>luminousIntensity-Exp</code>	0
<code>massExp</code>	0
<code>molarAmountExp</code>	0
<code>temperatureExp</code>	1
<code>timeExp</code>	0

**Table 3.104: PhysicalDimension PD\_K**

Common <code>PhysicalDimension</code> attributes	
<code>shortName</code>	PD_kg
<code>currentExp</code>	0
<code>lengthExp</code>	0
<code>luminousIntensity-Exp</code>	0
<code>massExp</code>	1
<code>molarAmountExp</code>	0
<code>temperatureExp</code>	0
<code>timeExp</code>	0

**Table 3.105: PhysicalDimension PD\_kg**

Common <code>PhysicalDimension</code> attributes	
<code>shortName</code>	PD_NoUnit
<code>currentExp</code>	0
<code>lengthExp</code>	0
<code>luminousIntensity-Exp</code>	0
<code>massExp</code>	0
<code>molarAmountExp</code>	0
<code>temperatureExp</code>	0
<code>timeExp</code>	0

**Table 3.106: PhysicalDimension PD\_NoUnit**

Common <code>PhysicalDimension</code> attributes	
shortName	PD_s
<code>currentExp</code>	0
<code>lengthExp</code>	0
<code>luminousIntensity-Exp</code>	0
<code>massExp</code>	0
<code>molarAmountExp</code>	0
<code>temperatureExp</code>	0
<code>timeExp</code>	1

**Table 3.107: PhysicalDimension PD\_s**

### 3.2.2.17 SwAddrMethods

Common SwAddrMethod attributes	
shortName	CAL
desc	Calibratable constants; safety level QM. Constants will be located in different memory sections depending on the alignment of the constant.
sectionType	calprm
memoryAllocation-KeywordPolicy	addrMethodShortNameAndAlignment
sectionInitializationPolicy	-
option	safetyQM

**Table 3.108: SwAddrMethod CAL**

Common SwAddrMethod attributes	
shortName	CODE_10MS
desc	Code of ECU-functions called every 10 ms; safety level QM.
sectionType	code
memoryAllocation-KeywordPolicy	addrMethodShortName
sectionInitializationPolicy	-
option	safetyQM

**Table 3.109: SwAddrMethod CODE\_10MS**

Common SwAddrMethod attributes	
shortName	CONST_SLOW
desc	Non calibratable constants of ECU-functions called seldom; safety level QM.
sectionType	const
memoryAllocation-KeywordPolicy	addrMethodShortName
sectionInitializationPolicy	-
option	safetyQM

**Table 3.110: SwAddrMethod CONST\_SLOW**

Common <code>SwAddrMethod</code> attributes	
<code>shortName</code>	DATA
<code>desc</code>	Variables of ECU-functions; safety level QM. Variables will be located in different memory sections depending on the alignment of the variable.
<code>sectionType</code>	var
<code>memoryAllocation-KeywordPolicy</code>	<code>addrMethodShortNameAndAlignment</code>
<code>sectionInitializationPolicy</code>	INIT
<code>option</code>	safetyQM

**Table 3.111: SwAddrMethod DATA**

Common <code>SwAddrMethod</code> attributes	
<code>shortName</code>	DATA_NVDAT
<code>desc</code>	Variables stored in non-volatile memory; safety level QM.
<code>sectionType</code>	var
<code>memoryAllocation-KeywordPolicy</code>	<code>addrMethodShortName</code>
<code>sectionInitializationPolicy</code>	NO-INIT
<code>option</code>	nvData, safetyQM

**Table 3.112: SwAddrMethod DATA\_NVDAT**

## 4 Structured Requirements

Structured Requirements are available at different sections in the meta model. There exists a clear definition of the elements of `StructuredReq`. Based on these elements a variety of possible use cases can be applied. The following show cases shall illustrate this:

- provide additional information for configuration
- provide specification items as requirements
- provide diagnostical requirements
- provide decomposition of requirements

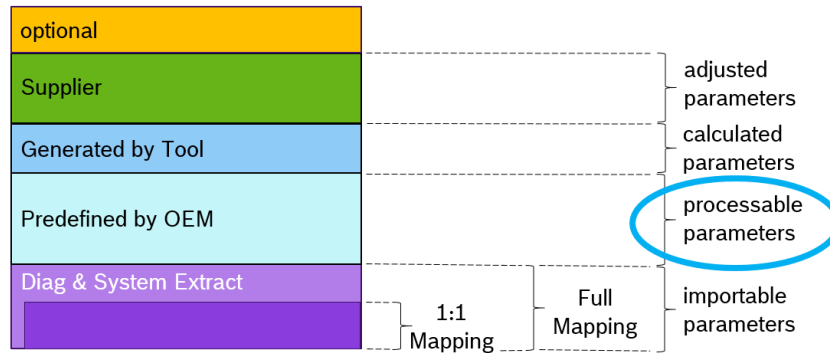
### 4.1 Introductory Show Case

An OEMs e.g. can specify additional unmapped Ecu Configuration information using Structured Requirements. They are serialized using standardized ARXML. This would ensure a consistent processing of these data by a suitable tooling solution on supplier side. In addition, further validation can be applied to data and lead to a high quality of the exchanged information. No new and additional elements have to be introduced in the meta model because `StructuredReq` still provides all needed entities.

#### 4.1.1 Additional information (M1 parameter)

The `System Template` [6] defines upstream mappings for a variety of M1 parameters. They are transported by the System Extract or Diagnostic Extract and presented in figure 4.1 as importable parameters.

The show case that an OEM needs to specify a value for not importable M1 parameters (no full mapping specified in AUTOSAR templates) to determine a dedicated behavior, e.g. `CanDevErrorDetect` which can be enabled by `true` or disabled by `false` is illustrated in figure 4.1 as processable parameters. A tool based processing of these information can increase efficiency and minimize error-proneness. This show case is targeting to solve this by using the `StructuredReq`.



**Figure 4.1: Types of Parameters**

The figure 4.1 shows further type of parameters; calculated and adjusted parameters; which exist but are not in the scope of show case.

In listing 4.1 the [StructuredReq](#) for CanDevErrorDetect is illustrated.

**Listing 4.1: Example 1 for [StructuredReq](#)**

```

<STRUCTURED-REQ SI="UNMAPPED_ECUC">
  <SHORT-NAME>CAN_00001</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">CanDevErrorDetect</L-4>
  </LONG-NAME>
  <CATEGORY>REQUIREMENT_ITEM</CATEGORY>
  <DATE>2018-03-19</DATE>
  <ISSUED-BY>OEM</ISSUED-BY>
  <TYPE>valid</TYPE>
  <IMPORTANCE>High</IMPORTANCE>
  <DESCRIPTION>
    <LIST>
      <ITEM SI="Guideline">
        <P>
          <L-1 L="EN">Switches the development error detection and
            notification on or off. true: detection and notification is
            enabled. false: detection and notification is disabled.</L-1>
        </P>
      </ITEM>
      <ITEM SI="SupplierSetting">
        <P>
          <L-1 L="FOR-ALL">FALSE</L-1>
        </P>
      </ITEM>
      <ITEM SI="ARVersion">
        <P>
          <L-1 L="FOR-ALL">4.2.2</L-1>
        </P>
        <P>
          <L-1 L="FOR-ALL">4.3.0</L-1>
        </P>
        <P>
          <L-1 L="FOR-ALL">4.4.0</L-1>
        </P>
      </ITEM>
    </LIST>
  </DESCRIPTION>
</STRUCTURED-REQ>
    
```



```

        <ITEM SI="ValueRange">
            <P>
                <L-1 L="FOR-ALL">TRUE</L-1>
            </P>
            <P>
                <L-1 L="FOR-ALL">FALSE</L-1>
            </P>
        </ITEM>
    </LIST>
</DESCRIPTION>
<RATIONALE>
    <P>
        <L-1 L="EN">The feature development error detection shall be
            disabled for production code else it has to be ensured that there
            will be no side effects from this code.</L-1>
    </P>
</RATIONALE>
<USE-CASE>
    <P>
        <L-1 L="EN">preconfigured/default</L-1>
    </P>
</USE-CASE>
</STRUCTURED-REQ>
    
```

For illustrative reasons the following listing 4.2 only contains a subset of the example where the ValueRange is speccified using a DataConstr.

#### Listing 4.2: Example 2 for ValueRange in StructuredReq

```

<ITEM SI="ValueRange">
    <P>
        <L-1 L="FOR-ALL">
            <XREF><REFERRABLE-REF DEST="DATA-CONSTR">/OEM/
                ImplementationDataTypes/DataConstrs/Range_Boolean</REFERRABLE-
                REF></XREF></L-1>
        </P>
    </ITEM>
    
```

The corresponding DataConstr is shown in the listing 4.3.

#### Listing 4.3: Example 2 for DataConstr

```

<DATA-CONSTR>
    <SHORT-NAME>Range_Boolean</SHORT-NAME>
    <DATA-CONSTR-RULES>
        <DATA-CONSTR-RULE>
            <PHYS-CONSTRS>
                <LOWER-LIMIT>TRUE</LOWER-LIMIT>
                <UPPER-LIMIT>FALSE</UPPER-LIMIT>
            </PHYS-CONSTRS>
        </DATA-CONSTR-RULE>
    </DATA-CONSTR-RULES>
</DATA-CONSTR>
    
```

The same data can also be rendered in the table 4.1.

<b>ReqID</b>	CAN_00001	CanDevErrorDetect	<b>Setting</b>	FALSE
<b>Use Case</b>	preconfigured/default	4.2.2, 4.3.0, 4.4.0	<b>Range</b>	FALSE, TRUE
<b>Description</b>	Switches the development error detection and notification on or off. true: detection and notification is enabled. false: detection and notification is disabled.	<b>Rational</b>	The feature development error detection shall be disabled for production code else it has to be ensured that there will be no side effects from this code.	

**Table 4.1: StructuredReq of CanDevErrorDetect**

## 4.1.2 Specification items as requirements

AUTOSAR uses [StructuredReq](#) within AUTOSAR RS documents [TPS\_STDT\_00060]. They are automatically generated based on the specification generation process. These specification items use the attributes of [StructuredReq](#) including [Traceable](#), to realize the up tracing. A dedicated example is illustrated in [4.4](#) this.

### Listing 4.4: Example for [StructuredReq](#) in RS AUTOSAR Specification

```

<STRUCTURED-REQ>
  <SHORT-NAME>RS_SYST_00020</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">Exclusion of signals from a specific physical line</L-4>
  </LONG-NAME>
  <CATEGORY>REQUIREMENT_ITEM</CATEGORY>
  <TRACE-REFS>
    <TRACE-REF BASE="ArTrace" DEST="TRACEABLE">RS_Main_00150</TRACE-REF>
    <TRACE-REF BASE="ArTrace" DEST="TRACEABLE">RS_Main_00030</TRACE-REF>
  </TRACE-REFS>
  <DATE>2004-10-29</DATE>
  <ISSUED-BY>WP Methodology& Templates</ISSUED-BY>
  <TYPE>valid</TYPE>
  <IMPORTANCE>medium</IMPORTANCE>
  <DESCRIPTION>
    <P>
      <L-1 L="EN">The System Constraint Description shall be able to
        describe that signals have not to be mapped to a specific
        physical line.</L-1>
    </P>
  </DESCRIPTION>
  <RATIONALE>
    <P>
      <L-1 L="EN">Some physical lines can result unsuitable (too slow,
        unsafe communication protocol, etc.) for the transmission of
        some specific signals.</L-1>
    </P>
  </RATIONALE>
  <USE-CASE>
    <P>
      <L-1 L="EN">Most of power train signals cannot be mapped to a low
        speed CAN bus, due to their timing requirements.</L-1>
    </P>
  </USE-CASE>
    
```

```

<CONFLICTS>
  <P>
    <L-1 L="EN">None identified.</L-1>
  </P>
</CONFLICTS>
<SUPPORTING-MATERIAL>
  <P>
    <L-1 L="EN">---</L-1>
  </P>
</SUPPORTING-MATERIAL>
<REMARK>
  <P>
    <L-1 L="EN">---</L-1>
  </P>
</REMARK>
</STRUCTURED-REQ>
    
```

The [Traceable](#) is realized by the tags `<TRACE-REF>`s in inside `<TRACE-REFS>`. In the specification the up traces appear as links below the table.

**[RS\_SYST\_00020] Exclusion of signals from a specific physical line [**

<b>Type:</b>	valid
<b>Description:</b>	The System Constraint Description shall be able to describe that signals have not to be mapped to a specific physical line.
<b>Rationale:</b>	Some physical lines can result unsuitable (too slow, unsafe communication protocol, etc.) for the transmission of some specific signals.
<b>Dependencies:</b>	None identified.
<b>Use Case:</b>	Most of power train signals cannot be mapped to a low speed CAN bus, due to their timing requirements.
<b>Supporting Material:</b>	-

**]([RS\\_Main\\_00150](#), [RS\\_Main\\_00030](#))**

**Figure 4.2: Spec item represented by [StructuredReq](#)**

Using [StructuredReq](#) a complete requirements description and tracing is available.

### 4.1.3 Diagnostic requirements

In this context the [StructuredReq](#) is embedded in [DiagnosticAccessPermission](#). Mainly the description of the two conditions are shown here.

**Listing 4.5: Example for the definition of pre- and run-conditions for [DiagnosticAccessPermission](#)**

```

<DIAGNOSTIC-ACCESS-PERMISSION>
  <SHORT-NAME>exampleAccessPermission</SHORT-NAME>
  <INTRODUCTION>
    <STRUCTURED-REQ>
      <SHORT-NAME>precondition</SHORT-NAME>
    
```

```

<CATEGORY>DIAG_ACCESS_PERM_PRE_COND</CATEGORY>
<DESCRIPTION>
  <P>
    <L-1 L="EN">This is a textual description of a pre-
      condition</L-1>
  </P>
</DESCRIPTION>
</STRUCTURED-REQ>
<STRUCTURED-REQ>
  <SHORT-NAME>runcondition</SHORT-NAME>
  <CATEGORY>DIAG_ACCESS_PERM_RUN_COND</CATEGORY>
  <DESCRIPTION>
    <P>
      <L-1 L="EN">This is a textual description of a run-
        condition</L-1>
    </P>
  </DESCRIPTION>
</STRUCTURED-REQ>
</INTRODUCTION>
<DIAGNOSTIC-SESSION-REFS>
  <DIAGNOSTIC-SESSION-REF DEST="DIAGNOSTIC-SESSION">/AUTOSAR/
    UseCase_230/ExampleSession</DIAGNOSTIC-SESSION-REF>
</DIAGNOSTIC-SESSION-REFS>
<SECURITY-LEVEL-REFS>
  <SECURITY-LEVEL-REF DEST="DIAGNOSTIC-SECURITY-LEVEL">/AUTOSAR/
    UseCase_230/ExampleSecurityLevel</SECURITY-LEVEL-REF>
</SECURITY-LEVEL-REFS>
</DIAGNOSTIC-ACCESS-PERMISSION>
    
```

#### 4.1.4 Decomposition of requirements

The [StructuredReq](#) can also be used for decomposition of requirements.

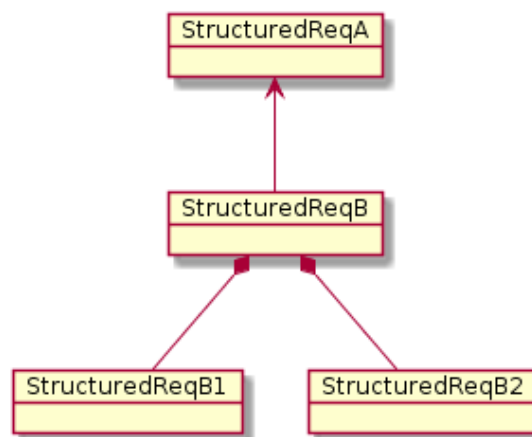


Figure 4.3: Decomposition of [StructuredReq](#)

In figure 4.3 the [StructuredReqB](#) has an upstream requirement to [StructuredReqA](#) realized by [Traceable](#) using `<TRACE-REF>`. The [StructuredReqB](#) itself is decomposed by [StructuredReqB1](#) and [StructuredReqB2](#). This is realized by the

attribute dependencies of `StructuredReq`. It contains two `<TRACE>`s which in turn use `<TRACE-REF>` again.

**Listing 4.6: Example for decomposition of `StructuredReq`**

```

<STRUCTURED-REQ>
  <SHORT-NAME>A</SHORT-NAME>
  <CATEGORY>REQUIREMENT_ITEM</CATEGORY>
  <DATE>2019-07-19</DATE>
  <ISSUED-BY>OEM</ISSUED-BY>
  <TYPE>valid</TYPE>
  <IMPORTANCE>High</IMPORTANCE>
</STRUCTURED-REQ>
<STRUCTURED-REQ>
  <SHORT-NAME>B</SHORT-NAME>
  <CATEGORY>REQUIREMENT_ITEM</CATEGORY>
  <TRACE-REFS>
    <TRACE-REF DEST="STRUCTURED-REQ">OEM/Documentation/
      DecompositionExample/A</TRACE-REF>
  </TRACE-REFS>
  <DATE>2019-07-19</DATE>
  <ISSUED-BY>OEM</ISSUED-BY>
  <TYPE>valid</TYPE>
  <IMPORTANCE>High</IMPORTANCE>
  <DEPENDENCIES>
    <TRACE>
      <SHORT-NAME>DECOMPB1</SHORT-NAME>
      <LONG-NAME>
        <L-4 L="EN">Decomposition element 1</L-4>
      </LONG-NAME>
      <CATEGORY>DECOMPOSITION</CATEGORY>
      <TRACE-REFS>
        <TRACE-REF DEST="STRUCTURED-REQ">OEM/Documentation/
          DecompositionExample/B1</TRACE-REF>
      </TRACE-REFS>
    </TRACE>
    <TRACE>
      <SHORT-NAME>DECOMPB2</SHORT-NAME>
      <LONG-NAME>
        <L-4 L="EN">Decomposition element 2</L-4>
      </LONG-NAME>
      <CATEGORY>DECOMPOSITION</CATEGORY>
      <TRACE-REFS>
        <TRACE-REF DEST="STRUCTURED-REQ">OEM/Documentation/
          DecompositionExample/B2</TRACE-REF>
      </TRACE-REFS>
    </TRACE>
  </DEPENDENCIES>
</STRUCTURED-REQ>
<STRUCTURED-REQ>
  <SHORT-NAME>B1</SHORT-NAME>
  <CATEGORY>REQUIREMENT_ITEM</CATEGORY>
  <DATE>2019-07-19</DATE>
  <ISSUED-BY>OEM</ISSUED-BY>
  <TYPE>valid</TYPE>
  <IMPORTANCE>High</IMPORTANCE>
    
```

```
</STRUCTURED-REQ>  
<STRUCTURED-REQ>  
  <SHORT-NAME>B2</SHORT-NAME>  
  <CATEGORY>REQUIREMENT_ITEM</CATEGORY>  
  <DATE>2019-07-19</DATE>  
  <ISSUED-BY>OEM</ISSUED-BY>  
  <TYPE>valid</TYPE>  
  <IMPORTANCE>High</IMPORTANCE>  
</STRUCTURED-REQ>
```

## A Mentioned Class Tables

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

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

**Table A.1: ARElement**

<b>Class</b>	<b>ARPackage</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
<b>Note</b>	AUTOSAR package, allowing to create top level packages to structure the contained ARElements. ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package. This is an extended version of MSR's SW-SYSTEM.			
<b>Base</b>	<i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>CollectableElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–





Class	ARPackage			
arPackage	<a href="#">ARPackage</a>	*	aggr	This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableElement	*	aggr	Elements that are part of this package <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20
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. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=shortLabel xml.sequenceOffset=10

**Table A.2: ARPackage**

Class	AliasNameSet			
Package	M2::AUTOSARTemplates::CommonStructure::FlatMap			
Note	This meta-class represents a set of AliasNames. The AliasNameSet can for example be an input to the A2L-Generator. <b>Tags:</b> atp.recommendedPackage=AliasNameSets			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">Multilanguage</a> <a href="#">Referrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
aliasName	AliasNameAssignment	1..*	aggr	AliasNames contained in the AliasNameSet. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortLabel vh.latestBindingTime=preCompileTime

**Table A.3: AliasNameSet**

Class	AnyInstanceRef			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::AnyInstanceRef			
Note	Describes a reference to any instance in an AUTOSAR model. This is the most generic form of an instance ref. Refer to the superclass notes for more details.			
Base	<a href="#">ARObject</a> , <a href="#">AtpInstanceRef</a>			
Attribute	Type	Mult.	Kind	Note
base	AtpClassifier	1	ref	This is the base from which navigation path begins. <b>Stereotypes:</b> atpDerived







Class		AnyInstanceRef		
contextElement	AtpFeature	*	ref	This is one step in the navigation path specified by the instance ref.
target	AtpFeature	1	ref	This is the target of the instance ref.

**Table A.4: AnyInstanceRef**

Class		ApplicationArrayType		
Package		M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes		
Note		An application data type which is an array, each element is of the same application data type. <b>Tags:</b> atp.recommendedPackage=ApplicationDataTypes		
Base		<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">ApplicationCompositeDataType</a> , <a href="#">ApplicationDataType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>		
Attribute	Type	Mult.	Kind	Note
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow if it is a variable size array.
element	<a href="#">ApplicationArrayElement</a>	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 A.5: ApplicationArrayType**

Class		ApplicationArrayElement		
Package		M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes		
Note		Describes the properties of the elements of an application array data type.		
Base		<a href="#">ARObject</a> , <a href="#">ApplicationCompositeElementDataPrototype</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">DataPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>		
Attribute	Type	Mult.	Kind	Note
arraySizeHandling	ArraySizeHandlingEnum	0..1	attr	The way how the size of the array is handled.
arraySizeSemantics	<a href="#">ArraySizeSemanticsEnum</a>	0..1	attr	This attribute controls how the information about the array size shall be interpreted.
indexDataType	<a href="#">ApplicationPrimitiveDataType</a>	0..1	ref	This reference can be taken to assign a CompuMethod of category TEXTTABLE to the array. The texttable entries associate a textual value to an index number such that the element with that index number is represented by a symbolic name.
maxNumberOfElements	PositiveInteger	0..1	attr	The maximum number of elements that the array can contain. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table A.6: ApplicationArrayElement**

<b>Class</b>	<b>ApplicationCompositeDataType</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	Abstract base class for all application data types composed of other data types.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">ApplicationDataType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">ApplicationArrayDataType</a> , <a href="#">ApplicationRecordDataType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.7: ApplicationCompositeDataType**

<b>Class</b>	<b>ApplicationCompositeElementDataPrototype</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	This class represents a data prototype which is aggregated within a composite application data type (record or array). It is introduced to provide a better distinction between target and context in instance Refs.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">DataPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">ApplicationArrayElement</a> , <a href="#">ApplicationRecordElement</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
type	<a href="#">ApplicationDataType</a>	1	tref	This represents the corresponding data type. <b>Stereotypes:</b> isOfType

**Table A.8: ApplicationCompositeElementDataPrototype**

<b>Class</b>	<b>ApplicationDataType</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	<p>ApplicationDataType defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake.</p> <p>An ApplicationDataType represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianness, etc.</p> <p>It should be possible to model the application level aspects of a VFB system by using ApplicationData Types only.</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">ApplicationCompositeDataType</a> , <a href="#">ApplicationPrimitiveDataType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.9: ApplicationDataType**

<b>Class</b>	<b>ApplicationPrimitiveDataType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	<p>A primitive data type defines a set of allowed values.</p> <p><b>Tags:</b>atp.recommendedPackage=ApplicationDataTypes</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">ApplicationDataType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			





Class	ApplicationPrimitiveDataType			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

**Table A.10: ApplicationPrimitiveDataType**

Class	ApplicationRecordDataType			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	An application data type which can be decomposed into prototypes of other application data types. <b>Tags:</b> atp.recommendedPackage=ApplicationDataTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">ApplicationCompositeDataType</a> , <a href="#">ApplicationDataType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
element (ordered)	<a href="#">ApplicationRecordElement</a>	1..*	aggr	Specifies an element of a record.  The aggregation of ApplicationRecordElement is subject to variability with the purpose to support the conditional existence of elements inside a ApplicationrecordData Type.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table A.11: ApplicationRecordDataType**

Class	ApplicationRecordElement			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	Describes the properties of one particular element of an application record data type.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">ApplicationCompositeElementDataPrototype</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">DataPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing ApplicationRecordElement as optional. This means the that, at runtime, the ApplicationRecord Element may or may not have a valid value and shall therefore be ignored.  The underlying runtime software provides means to set the ApplicationRecordElement as not valid at the sending end of a communication and determine its validity at the receiving end.

**Table A.12: ApplicationRecordElement**

Class	ApplicationSwComponentType			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	The ApplicationSwComponentType is used to represent the application software. <b>Tags:</b> atp.recommendedPackage=SwComponentTypes			





<b>Class</b>	<b>ApplicationSwComponentType</b>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtomicSwComponentType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.13: ApplicationSwComponentType**

<b>Enumeration</b>	<b>ArraySizeSemanticsEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes
<b>Note</b>	This type controls how the information about the number of elements in an ApplicationArrayDataType is to be interpreted.
<b>Literal</b>	<b>Description</b>
fixedSize	This means that the ApplicationArrayDataType will always have a fixed number of elements. <b>Tags:</b> atp.EnumerationLiteralIndex=0
variableSize	This implies that the actual number of elements in the ApplicationArrayDataType might vary at run-time. The value of arraySize represents the maximum number of elements in the array. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table A.14: ArraySizeSemanticsEnum**

<b>Class</b>	<b>AssemblySwConnector</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
<b>Note</b>	AssemblySwConnectors are exclusively used to connect SwComponentPrototypes in the context of a CompositionSwComponentType.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">SwConnector</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
provider	AbstractProvidedPort Prototype	0..1	iref	Instance of providing port.
requester	AbstractRequiredPort Prototype	0..1	iref	Instance of requiring port.

**Table A.15: AssemblySwConnector**

<b>Class</b>	<b>AtomicSwComponentType</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	An atomic software component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Subclasses</b>	<a href="#">ApplicationSwComponentType</a> , <a href="#">ComplexDeviceDriverSwComponentType</a> , <a href="#">EcuAbstractionSwComponentType</a> , <a href="#">NvBlockSwComponentType</a> , <a href="#">SensorActuatorSwComponentType</a> , <a href="#">ServiceProxySwComponentType</a> , <a href="#">ServiceSwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–





<b>Class</b>		<b>AtomicSwComponentType</b> (abstract)		
internalBehavior	<a href="#">SwcInternalBehavior</a>	0..1	aggr	The SwcInternalBehaviors owned by an AtomicSw ComponentType can be located in a different physical file. Therefore the aggregation is <<atpSplitable>>. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=internalBehavior, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the AtomicSw ComponentType. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=shortName

**Table A.16: AtomicSwComponentType**

<b>Class</b>		<b>AutosarDataPrototype</b> (abstract)		
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	Base class for prototypical roles of an AutosarDataType.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">DataPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">ArgumentDataPrototype</a> , <a href="#">ParameterDataPrototype</a> , <a href="#">VariableDataPrototype</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
type	AutosarDataType	1	tref	This represents the corresponding data type. <b>Stereotypes:</b> isOfType

**Table A.17: AutosarDataPrototype**

<b>Class</b>		<b>CompositionSwComponentType</b>		
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
<b>Note</b>	A CompositionSwComponentType aggregates SwComponentPrototypes (that in turn are typed by Sw ComponentTypes) 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. <b>Tags:</b> atp.recommendedPackage=SwComponentTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
component	<a href="#">SwComponentPrototype</a>	*	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 Sw ComponentPrototypes 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.





Class	CompositionSwComponentType			
				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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> 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 AssemblySw Connectors between ApplicationSwComponentTypes and ServiceSwComponentTypes during the ECU integration. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortCom Spec. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=constantValueMapping
dataType Mapping	<a href="#">DataTypeMappingSet</a>	*	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 implementors mainly because the RTE generator is not concerned about the CompositionSwComponent Types. Rationale: if the mapping of ApplicationDataTypes on the delegated and inner PortPrototype matches then the mapping to ImplementationDataTypes is not impacting compatibility. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=dataTypeMapping
instantiation RTEEventProps	InstantiationRTEEvent Props	*	aggr	This allows to define instantiation specific properties for RTE Events, in particular for instance specific scheduling. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortLabel, variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime

**Table A.18: CompositionSwComponentType**

<b>Class</b>	<b>CompuConstTextContent</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the textual content of a scale.			
<b>Base</b>	<i>ARObject, CompuConstContent</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
vt	VerbatimString	1	attr	This represents a textual constant in the computation method.

**Table A.19: CompuConstTextContent**

<b>Class</b>	<b>CompuMethod</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	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.  <b>Tags:</b> atp.recommendedPackage=CompuMethods			
<b>Base</b>	<i>ARElement, ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compuInternalToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values.  <b>Tags:</b> xml.sequenceOffset=80
compuPhysToInternal	Compu	0..1	aggr	This represents the computation from physical values to the internal values.  <b>Tags:</b> 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.  <b>Tags:</b> xml.sequenceOffset=20
unit	<a href="#">Unit</a>	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies.  <b>Tags:</b> xml.sequenceOffset=30

**Table A.20: CompuMethod**

<b>Class</b>	<b>CompuRationalCoeffs</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the ability to express a rational function by specifying the coefficients of nominator and denominator.			
<b>Base</b>	<i>ARObject</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compuDenominator	CompuNominatorDenominator	1	aggr	This is the denominator of the expression.  <b>Tags:</b> xml.sequenceOffset=30
compuNumerator	CompuNominatorDenominator	1	aggr	This is the numerator of the rational expression.  <b>Tags:</b> xml.sequenceOffset=20

**Table A.21: CompuRationalCoeffs**

<b>Class</b>	<b>CompuScale</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the ability to specify one segment of a segmented computation method.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compuInverseValue	CompuConst	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se. <b>Tags:</b> xml.sequenceOffset=60
compuScaleContents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false
desc	MultiLanguageOverviewParagraph	0..1	aggr	<desc> represents a general but brief description of the object in question. <b>Tags:</b> xml.sequenceOffset=30
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the scale. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
mask	PositiveInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap.  To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted.  The processing has to be done in order of the COMPU-SCALE elements. <b>Tags:</b> xml.sequenceOffset=35
shortLabel	Identifier	0..1	attr	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. <b>Tags:</b> xml.sequenceOffset=20
symbol	CIdentifier	0..1	attr	The symbol, if provided, is used by code generators to get a C identifier for the CompuScale. The name will be used as is for the code generation, therefore it needs to be unique within the generation context. <b>Tags:</b> xml.sequenceOffset=25
upperLimit	Limit	0..1	attr	This specifies the upper limit of a of the scale. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

**Table A.22: CompuScale**



<b>Class</b>	<b>DataConstr</b>			
<b>Package</b>	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
<b>Note</b>	This meta-class represents the ability to specify constraints on data. <b>Tags:</b> atp.recommendedPackage=DataConstrs			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataConstrRule	<a href="#">DataConstrRule</a>	*	aggr	This is one particular rule within the data constraints. <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

**Table A.23: DataConstr**

<b>Class</b>	<b>DataConstrRule</b>			
<b>Package</b>	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
<b>Note</b>	This meta-class represents the ability to express one specific data constraint rule.			
<b>Base</b>	<a href="#">ARObject</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
constrLevel	Integer	0..1	attr	This attribute describes the category of a constraint. One of its functions is in the area of constraint violation, where it can be used from a certain level, to produce error messages.  The lower the level, the more stringent the check. Used to distinguish hard or soft limits. <b>Tags:</b> xml.sequenceOffset=20
internalConstrs	<a href="#">InternalConstrs</a>	0..1	aggr	Describes the limitations applicable on the internal domain (as opposed to the physical domain). <b>Tags:</b> xml.sequenceOffset=40
physConstrs	<a href="#">PhysConstrs</a>	0..1	aggr	Describes the limitations applicable on the physical domain (as opposed to the internal domain). <b>Tags:</b> xml.sequenceOffset=30

**Table A.24: DataConstrRule**

<b>Class</b>	<b>DataPrototype</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	Base class for prototypical roles of any data type.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">ApplicationCompositeElementDataPrototype</a> , <a href="#">AutosarDataPrototype</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
swDataDef Props	<a href="#">SwDataDefProps</a>	0..1	aggr	This property allows to specify data definition properties which apply on data prototype level.

**Table A.25: DataPrototype**

<b>Class</b>	<b>DataTypeMap</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
<b>Note</b>	This class represents the relationship between ApplicationDataType and its implementing AbstractImplementationDataType.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
applicationDataType	ApplicationDataType	1	ref	This is the corresponding ApplicationDataType
implementationDataType	AbstractImplementationDataType	1	ref	This is the corresponding AbstractImplementationDataType.

**Table A.26: DataTypeMap**

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

**Table A.27: DataTypeMappingSet**

<b>Class</b>	<b>DiagnosticAccessPermission</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
<b>Note</b>	This represents the specification of whether a given service can be accessed according to the existence of meta-classes referenced by a particular DiagnosticAccessPermission.  In other words, this meta-class acts as a mapping element between several (otherwise unrelated) pieces of information that are put into context for the purpose of checking for access rights. <b>Tags:</b> atp.recommendedPackage=DiagnosticAccessPermissions			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
diagnosticSession	DiagnosticSession	*	ref	This represents the associated DiagnosticSessions
environmentalCondition	DiagnosticEnvironmentalCondition	0..1	ref	This represents the environmental conditions associated with the access permission.
securityLevel	DiagnosticSecurityLevel	*	ref	This represents the associated DiagnosticSecurityLevels

**Table A.28: DiagnosticAccessPermission**

<b>Class</b>	<b>EcuInstance</b>			
<b>Package</b>	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
<b>Note</b>	ECUInstances are used to define the ECUs used in the topology. The type of the ECU is defined by a reference to an ECU specified with the ECU resource description. <b>Tags:</b> atp.recommendedPackage=EcuInstances			
<b>Base</b>	ARObject, CollectableElement, FibexElement, <i>Identifiable</i> , MultilanguageReferrable, Packageable Element, <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
associatedComIPduGroup	ISignalIPduGroup	*	ref	With this reference it is possible to identify which ISignalIPduGroups are applicable for which Communication Connector/ ECU.  Only top level ISignalIPduGroups shall be referenced by an EcuInstance. If an ISignalIPduGroup contains other ISignalIPduGroups than these contained ISignalIPduGroups shall not be referenced by the EcuInstance. Contained ISignalIPduGroups are associated to an Ecu Instance via the top level ISignalIPduGroup.
associatedConsumedProvidedServiceInstanceGroup	ConsumedProvidedServiceInstanceGroup	*	ref	With this reference it is possible to identify which ConsumedProvidedServiceInstanceGroups are applicable for which ECUInstance.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild
associatedPdurIPduGroup	PdurIPduGroup	*	ref	With this reference it is possible to identify which PdurIPduGroups are applicable for which Communication Connector/ ECU.
clientIdRange	ClientIdRange	0..1	aggr	Restriction of the Client Identifier for this Ecu to an allowed range of numerical values. The Client Identifier of the transaction handle is generated by the client RTE for inter-Ecu Client/Server communication.
comConfigurationGwTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionRouteSignals of the AUTOSAR COM module in seconds.
comConfigurationRxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionRx of the AUTOSAR COM module in seconds.
comConfigurationTxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionTx of the AUTOSAR COM module in seconds.
comEnableMDTForCyclicTransmission	Boolean	0..1	attr	Enables for the Com module of this EcuInstance the minimum delay time monitoring for cyclic and repeated transmissions (TransmissionModeTiming has cyclic Timing assigned or eventControlledTiming with numberOfRepetitions > 0).
commController	CommunicationController	1..*	aggr	CommunicationControllers of the ECU.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild
connector	CommunicationConnector	*	aggr	All channels controlled by a single controller.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild
dltConfig	DltConfig	0..1	aggr	Describes the Dlt configuration on this EcuInstance.
dolpConfig	DolpConfig	0..1	aggr	Dolp configuration on this EcuInstance.  <b>Tags:</b> atp.Status=draft





Class	EcuInstance			
ecuInstance Props	EcuInstanceProps	0..1	aggr	Additional properties of the EcuInstance that may vary in different Variants of the Ecu. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild
ethSwitchPort Group Derivation	Boolean	0..1	attr	Defines whether the derivation of SwitchPortGroups based on VLAN and/or CouplingPort.pncMapping shall be performed for this EcuInstance. If not defined the derivation shall not be done.
partition	EcuPartition	*	aggr	Optional definition of Partitions within an Ecu.
pncPrepare SleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
pnc Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true then all available PNCs will be woken up as soon as a channel wakeup occurs. This is ensured by adding all PNCs to all channel wakeup sources during upstream mapping.
pnResetTime	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA.
sleepMode Supported	Boolean	1	attr	Specifies whether the ECU instance may be put to a "low power mode" <ul style="list-style-type: none"><li>• true: sleep mode is supported</li><li>• false: sleep mode is not supported</li></ul> Note: This flag may only be set to "true" if the feature is supported by both hardware and basic software.
v2xSupported	V2xSupportEnum	0..1	attr	This attribute is used to control the existence of the V2X stack on the given EcuInstance.
wakeUpOver BusSupported	Boolean	1	attr	Driver support for wakeup over Bus.

**Table A.29: EcuInstance**

Class	EcucModuleConfigurationValues			
Package	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
Note	Head of the configuration of one Module. A Module can be a BSW module as well as the RTE and ECU Infrastructure. As part of the BSW module description, the EcucModuleConfigurationValues element has two different roles: The recommendedConfiguration contains parameter values recommended by the BSW module vendor. The preconfiguredConfiguration contains values for those parameters which are fixed by the implementation and cannot be changed. These two EcucModuleConfigurationValues are used when the base EcucModuleConfigurationValues (as part of the base ECU configuration) is created to fill parameters with initial values. <b>Tags:</b> atp.recommendedPackage=EcucModuleConfigurationValues			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note





Class	EcucModuleConfigurationValues			
container	EcucContainerValue	1..*	aggr	Aggregates all containers that belong to this module configuration. atpVariation: [RS_ECUC_00078] <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=definition, shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild xml.sequenceOffset=10
definition	EcucModuleDef	1	ref	Reference to the definition of this EcucModule ConfigurationValues element. Typically, this is a vendor specific module configuration. <b>Tags:</b> xml.sequenceOffset=-10
ecucDefEdition	RevisionLabelString	1	attr	This is the version info of the ModuleDef ECUC Parameter definition to which this values conform to / are based on.  For the Definition of ModuleDef ECUC Parameters the AdminData shall be used to express the semantic changes. The compatibility rules between the definition and value revision labels is up to the module's vendor.
implementation ConfigVariant	EcucConfiguration VariantEnum	1	attr	Specifies the kind of deliverable this EcucModule ConfigurationValues element provides. If this element is not used in a particular role (e.g. preconfigured Configuration or recommendedConfiguration) then the value shall be one of VariantPreCompile, VariantLink Time, VariantPostBuild.
module Description	BswImplementation	0..1	ref	Referencing the BSW module description, which this EcucModuleConfigurationValues element is configuring. This is optional because the EcucModuleConfiguration Values element is also used to configure the ECU infrastructure (memory map) or Application SW-Cs. However in case the EcucModuleConfigurationValues are used to configure the module, the reference is mandatory in order to fetch module specific "common" published information.
postBuildVariant Used	Boolean	0..1	attr	Indicates whether a module implementation has or plans to have (i.e., introduced at link or post-build time) new post-build variation points. TRUE means yes, FALSE means no. If the attribute is not defined, FALSE semantics shall be assumed.

**Table A.30: EcucModuleConfigurationValues**

<b>Class</b>	<b>EcucValueCollection</b>			
<b>Package</b>	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
<b>Note</b>	This represents the anchor point of the ECU configuration description. <b>Tags:</b> atp.recommendedPackage=EcucValueCollections			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	EcucValueCollection			
ecucValue	<a href="#">EcucModuleConfigurationValues</a>	1..*	ref	References to the configuration of individual software modules that are present on this ECU. atpVariation: [RS_ECUC_00079] <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
ecuExtract	<a href="#">System</a>	1	ref	Represents the extract of the System Configuration that is relevant for the ECU configured with that ECU Configuration Description.

**Table A.31: EcucValueCollection**

Class	FlatInstanceDescriptor			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::FlatMap			
<b>Note</b>	Represents exactly one node (e.g. a component instance or data element) of the instance tree of a software system. The purpose of this element is to map the various nested representations of this instance to a flat representation and assign a unique name (shortName) to it.  Use cases: <ul style="list-style-type: none"> <li>Specify unique names of measurable data to be used by MCD tools</li> <li>Specify unique names of calibration data to be used by MCD tool</li> <li>Specify a unique name for an instance of a component prototype in the ECU extract of the system description</li> </ul> Note that in addition it is possible to assign alias names via AliasNameAssignment.			
<b>Base</b>	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
Attribute	Type	Mult.	Kind	Note
ecuExtractReference	AtpFeature	0..1	iref	Refers to the instance in the ECU extract. This is valid only, if the FlatMap is used in the context of an ECU extract.  The reference shall be such that it uniquely defines the object instance. For example, if a data prototype is declared as a role within an SwcInternalBehavior, it is not enough to state the SwcInternalBehavior as context and the aggregated data prototype as target. In addition, the reference shall also include the complete path identifying instance of the component prototype and the Atomic SoftwareComponentType, which is referred by the particular SwcInternalBehavior. <b>Tags:</b> xml.sequenceOffset=40
role	Identifier	0..1	attr	The role denotes the particular role of the downstream memory location described by this FlatInstanceDescriptor.  It applies to use case where one upstream object results in multiple downstream objects, e.g. ModeDeclaration GroupPrototypes which are measurable. In this case the RTE will provide locations for current mode, previous mode and next mode.
rtePluginProps	RtePluginProps	0..1	aggr	The properties of a communication graph with respect to the utilization of RTE Implementation Plug-in. <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=rtePluginProps
swDataDefProps	<a href="#">SwDataDefProps</a>	0..1	aggr	The properties of this FlatInstanceDescriptor.





Class	FlatInstanceDescriptor			
upstream Reference	AtpFeature	0..1	iref	<p>Refers to the instance in the context of an "upstream" descriptions, wich could be the system or system extract description, the basic software module description or (if a flat map is used in preliminary context) a description of an atomic component or composition. This reference is optional in case the flat map is used in ECU context.</p> <p>The reference shall be such that it uniquely defines the object instance in the given context. For example, if a data prototype is declared as a role within an SwcInternal Behavior, it is not enough to state the SwcInternal Behavior as context and the aggregated data prototype as target. In addition, the reference shall also include the complete path identifying the instance of the component prototype that contains the particular instance of Swc InternalBehavior.</p> <p><b>Tags:</b>xml.sequenceOffset=20</p>

**Table A.32: FlatInstanceDescriptor**

Class	FlatMap			
Package	M2::AUTOSARTemplates::CommonStructure::FlatMap			
Note	<p>Contains a flat list of references to software objects. This list is used to identify instances and to resolve name conflicts. The scope is given by the RootSwCompositionPrototype for which it is used, i.e. it can be applied to a system, system extract or ECU-extract.</p> <p>An instance of FlatMap may also be used in a preliminary context, e.g. in the scope of a software component before integration into a system. In this case it is not referred by a RootSwComposition Prototype.</p> <p><b>Tags:</b>atp.recommendedPackage=FlatMaps</p>			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">Multilanguage Referrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
instance	<a href="#">FlatInstanceDescriptor</a>	1..*	aggr	<p>A descriptor instance aggregated in the flat map.</p> <p>The variation point accounts for the fact, that the system in scope can be subject to variability, and thus the existence of some instances is variable.</p> <p>The aggregation has been made splitable because the content might be contributed by different stakeholders at different times in the workflow. Plus, the overall size might be so big that eventually it becomes more manageable if it is distributed over several files.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild</p>

**Table A.33: FlatMap**



<b>Class</b>	<i>Identifiable</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
<b>Note</b>	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.			
<b>Base</b>	<i>ARObject</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Subclasses</b>	<p><i>ARPackage</i>, <i>AbstractEvent</i>, <i>AbstractImplementationDataTypeElement</i>, <i>AbstractServiceInstance</i>, <i>ApplicationEndpoint</i>, <i>ApplicationError</i>, <i>ApplicationPartitionToEcuPartitionMapping</i>, <i>AsynchronousServerCallResultPoint</i>, <i>AtpBlueprint</i>, <i>AtpBlueprintable</i>, <i>AtpClassifier</i>, <i>AtpFeature</i>, <i>AutosarOperationArgumentInstance</i>, <i>AutosarVariableInstance</i>, <i>BswInternalTriggeringPoint</i>, <i>BswModuleDependency</i>, <i>BuildActionEntity</i>, <i>BuildActionEnvironment</i>, <i>CanTpAddress</i>, <i>CanTpChannel</i>, <i>CanTpNode</i>, <i>Chapter</i>, <i>ClassContentConditional</i>, <i>ClientIdDefinition</i>, <i>ClientServerOperation</i>, <i>Code</i>, <i>CollectableElement</i>, <i>ComManagementMapping</i>, <i>CommConnectorPort</i>, <i>CommunicationConnector</i>, <i>CommunicationController</i>, <i>Compiler</i>, <i>ConsistencyNeeds</i>, <i>ConsumedEventGroup</i>, <i>CouplingPort</i>, <i>CouplingPortStructuralElement</i>, <i>CryptoServiceMapping</i>, <i>DataPrototypeGroup</i>, <i>DataTransformation</i>, <i>DependencyOnArtifact</i>, <i>DiagEventDebounceAlgorithm</i>, <i>DiagnosticConnectedIndicator</i>, <i>DiagnosticDataElement</i>, <i>DiagnosticFunctionInhibitSource</i>, <i>DiagnosticMasterToSlaveEventMapping</i>, <i>DiagnosticRoutineSubfunction</i>, <i>DitArgument</i>, <i>DitLogChannel</i>, <i>DitMessage</i>, <i>DolpInterface</i>, <i>DolpLogicAddress</i>, <i>ECUMapping</i>, <i>EOCExecutableEntityRefAbstract</i>, <i>EcuPartition</i>, <i>EcucContainerValue</i>, <i>EcucDefinitionElement</i>, <i>EcucDestinationUriDef</i>, <i>EcucEnumerationLiteralDef</i>, <i>EcucQuery</i>, <i>EcucValidationCondition</i>, <i>EndToEndProtection</i>, <i>ExclusiveArea</i>, <i>ExecutableEntity</i>, <i>ExecutionTime</i>, <i>FMAttributeDef</i>, <i>FMFeatureMapAssertion</i>, <i>FMFeatureMapCondition</i>, <i>FMFeatureMapElement</i>, <i>FMFeatureRelation</i>, <i>FMFeatureRestriction</i>, <i>FMFeatureSelection</i>, <i>FlatInstanceDescriptor</i>, <i>FlexrayArTpNode</i>, <i>FlexrayTpConnectionControl</i>, <i>FlexrayTpNode</i>, <i>FlexrayTpPduPool</i>, <i>FrameTriggering</i>, <i>GeneralParameter</i>, <i>GlobalTimeGateway</i>, <i>GlobalTimeMaster</i>, <i>GlobalTimeSlave</i>, <i>HeapUsage</i>, <i>HwAttributeDef</i>, <i>HwAttributeLiteralDef</i>, <i>HwPin</i>, <i>HwPinGroup</i>, <i>IPSecRule</i>, <i>IPv6ExtHeaderFilterList</i>, <i>ISignalToIPduMapping</i>, <i>ISignalTriggering</i>, <i>IdentCaption</i>, <i>InternalTriggeringPoint</i>, <i>J1939SharedAddressCluster</i>, <i>J1939TpNode</i>, <i>Keyword</i>, <i>LifeCycleState</i>, <i>LinScheduleTable</i>, <i>LinTpNode</i>, <i>Linker</i>, <i>MacMulticastGroup</i>, <i>McDataInstance</i>, <i>MemorySection</i>, <i>ModeDeclaration</i>, <i>ModeDeclarationMapping</i>, <i>ModeSwitchPoint</i>, <i>NetworkEndpoint</i>, <i>NmCluster</i>, <i>NmEcu</i>, <i>NmNode</i>, <i>NvBlockDescriptor</i>, <i>PackageableElement</i>, <i>ParameterAccess</i>, <i>PduToFrameMapping</i>, <i>PduTriggering</i>, <i>PerInstanceMemory</i>, <i>PhysicalChannel</i>, <i>PortGroup</i>, <i>PortInterfaceMapping</i>, <i>PossibleErrorReaction</i>, <i>ResourceConsumption</i>, <i>RootSwCompositionPrototype</i>, <i>RptComponent</i>, <i>RptContainer</i>, <i>RptExecutableEntity</i>, <i>RptExecutableEntityEvent</i>, <i>RptExecutionContext</i>, <i>RptProfile</i>, <i>RptServicePoint</i>, <i>RunnableEntityGroup</i>, <i>SdgAttribute</i>, <i>SdgClass</i>, <i>SecureCommunicationAuthenticationProps</i>, <i>SecureCommunicationFreshnessProps</i>, <i>ServerCallPoint</i>, <i>ServiceNeeds</i>, <i>SignalServiceTranslationElementProps</i>, <i>SignalServiceTranslationEventProps</i>, <i>SignalServiceTranslationProps</i>, <i>SocketAddress</i>, <i>SomeipTpChannel</i>, <i>SpecElementReference</i>, <i>StackUsage</i>, <i>StaticSocketConnection</i>, <i>StructuredReq</i>, <i>SwGenericAxisParamType</i>, <i>SwServiceArg</i>, <i>SwcServiceDependency</i>, <i>SwcToApplicationPartitionMapping</i>, <i>SwcToEcuMapping</i>, <i>SwcToImpiMapping</i>, <i>SystemMapping</i>, <i>TcpOptionFilterList</i>, <i>TimingCondition</i>, <i>TimingConstraint</i>, <i>TimingDescription</i>, <i>TimingExtensionResource</i>, <i>TimingModelInstance</i>, <i>TlsCryptoCipherSuite</i>, <i>Topic1</i>, <i>TpAddress</i>, <i>TraceableTable</i>, <i>TraceableText</i>, <i>TracedFailure</i>, <i>TransformationProps</i>, <i>TransformationTechnology</i>, <i>Trigger</i>, <i>VariableAccess</i>, <i>VariationPointProxy</i>, <i>ViewMap</i>, <i>VlanConfig</i>, <i>WaitPoint</i></p>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. <b>Tags:</b> 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. <b>Tags:</b> xml.sequenceOffset=-25
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. <b>Tags:</b> xml.sequenceOffset=-50







<b>Class</b>	<b>Identifiable</b> (abstract)			
desc	MultiLanguageOverviewParagraph	0..1	aggr	<p>This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question.</p> <p>More elaborate documentation, (in particular how the object is built or used) should go to "introduction".</p> <p><b>Tags:</b>xml.sequenceOffset=-60</p>
introduction	DocumentationBlock	0..1	aggr	<p>This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock.</p> <p><b>Tags:</b>xml.sequenceOffset=-30</p>
uuid	String	0..1	attr	<p>The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp.</p> <p><b>Tags:</b>xml.attribute=true</p>

**Table A.34: Identifiable**

<b>Class</b>	<b>ImplementationDataType</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
<b>Note</b>	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. <b>Tags:</b> atp.recommendedPackage=ImplementationDataTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">AObject</a> , <a href="#">AbstractImplementationDataType</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">AutosarDataType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
isStructWithOptionalElement	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE.  If set to True, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional.





Class	ImplementationDataType			
subElement (ordered)	ImplementationDataTypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of ImplementationDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a ImplementationDataType representing a structure. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the ImplementationDataType. <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=shortName
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

**Table A.35: ImplementationDataType**

Class	InstantiationDataDefProps			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::InstantiationDataDefProps			
<b>Note</b>	This is a general class allowing to apply additional SwDataDefProps to particular instantiations of a Data Prototype. Typically the accessibility and further information like alias names for a particular data is modeled on the level of DataPrototypes (especially VariableDataPrototypes, ParameterDataPrototypes). But due to the recursive structure of the meta-model concerning data types (a composite (data) type consists out of data prototypes) a part of the MCD information is described in the data type (in case of Application CompositeDataType). This is a strong restriction in the reuse of data typed because the data type should be re-used for different VariableDataPrototypes and ParameterDataPrototypes to guarantee type compatibility on C-implementation level (e.g. data of a Port is stored in PIM or a ParameterDataPrototype used as ROM Block and shall be typed by the same data type as NVRAM Block). This class overcomes such a restriction if applied properly.			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
parameter Instance	AutosarParameterRef	0..1	aggr	This is the particular ParameterDataPrototypes on which the swDataDefProps shall be applied.
swDataDef Props	<a href="#">SwDataDefProps</a>	1	aggr	These are the particular data definition properties which shall be applied
variableInstance	AutosarVariableRef	0..1	aggr	This is the particular VariableDataPrototypes on which the swDataDefProps shall be applied.

**Table A.36: InstantiationDataDefProps**

<b>Class</b>	<b>InternalBehavior</b> (abstract)
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::InternalBehavior
<b>Note</b>	Common base class (abstract) for the internal behavior of both software components and basic software modules/clusters.
<b>Base</b>	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>





Class	InternalBehavior (abstract)			
Subclasses	BswInternalBehavior, SvcInternalBehavior			
Attribute	Type	Mult.	Kind	Note
constantMemory	ParameterDataPrototype	*	aggr	<p>Describes a read only memory object containing characteristic value(s) implemented by this Internal Behavior.</p> <p>The shortName of ParameterDataPrototype has to be equal to the 'C' identifier of the described constant.</p> <p>The characteristic value(s) might be shared between Sw ComponentPrototypes of the same SwComponentType.</p> <p>The aggregation of constantMemory is subject to variability with the purpose to support variability in the software component or module implementations. Typically different algorithms in the implementation are requiring different number of memory objects.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=shortName, variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
constantValueMapping	ConstantSpecificationMappingSet	*	ref	<p>Reference to the ConstanSpecificationMapping to be applied for the particular InternalBehavior</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b>atp.Splitkey=constantValueMapping</p>
dataTypeMapping	DataTypeMappingSet	*	ref	<p>Reference to the DataTypeMapping to be applied for the particular InternalBehavior</p> <p><b>Stereotypes:</b> atpSplitable  <b>Tags:</b>atp.Splitkey=dataTypeMapping</p>
exclusiveArea	ExclusiveArea	*	aggr	<p>This specifies an ExclusiveArea for this InternalBehavior. The exclusiveArea is local to the component resp. module. The aggregation of ExclusiveAreas is subject to variability. Note: the number of ExclusiveAreas might vary due to the conditional existence of RunnableEntities or BswModuleEntities.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=shortName, variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
exclusiveAreaNestingOrder	ExclusiveAreaNestingOrder	*	aggr	<p>This represents the set of ExclusiveAreaNestingOrder owned by the InternalBehavior.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=shortName, variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
staticMemory	VariableDataPrototype	*	aggr	<p>Describes a read and writeable static memory object representing measurement variables implemented by this software component. The term "static" is used in the meaning of "non-temporary" and does not necessarily specify a linker encapsulation. This kind of memory is only supported if supportsMultipleInstantiation is FALSE.</p> <p>The shortName of the VariableDataPrototype has to be equal with the 'C' identifier of the described variable.</p>





<b>Class</b>	<b>InternalBehavior</b> (abstract)			
				<p style="text-align: center;">△</p> <p>The aggregation of staticMemory is subject to variability with the purpose to support variability in the software component's implementations.</p> <p>Typically different algorithms in the implementation are requiring different number of memory objects.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                  atp.Splitkey=shortName, variationPoint.shortLabel                  vh.latestBindingTime=preCompileTime</p>

**Table A.37: InternalBehavior**

<b>Class</b>	<b>McDataInstance</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport			
<b>Note</b>	<p>Describes the specific properties of one data instance in order to support measurement and/or calibration of this data instance.</p> <p>The most important attributes are:</p> <ul style="list-style-type: none"> <li>• Its shortName is copied from the ECU Flat map (if applicable) and will be used as identifier and for display by the MC system.</li> <li>• The category is copied from the corresponding data type (ApplicationDataType if defined, otherwise ImplementationDataType) as far as applicable.</li> <li>• The symbol is the one used in the programming language. It will be used to find out the actual memory address by the final generation tool with the help of linker generated information.</li> </ul> <p>It is assumed that in the M1 model this part and all the aggregated and referred elements (with the exception of the Flat Map and the references from ImplementationElementInParameterInstanceRef and McAccessDetails) are completely generated from "upstream" information. This means, that even if an element like e.g. a CompuMethod is only used via reference here, it will be copied into the M1 artifact which holds the complete McSupportData for a given Implementation.</p>			
<b>Base</b>	ARObject, <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
arraySize	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 number of elements.
displayIdentifier	McdIdentifier	0..1	attr	An optional attribute to be used to set the ASAM ASAP2 DISPLAY_IDENTIFIER attribute.
flatMapEntry	<a href="#">FlatInstanceDescriptor</a>	0..1	ref	<p>Reference to the corresponding entry in the ECU Flat Map. This allows to trace back to the original specification of the generated data instance. This link shall be added by the RTE generator mainly for documentation purposes.</p> <p>The reference is optional because</p> <ul style="list-style-type: none"> <li>• The McDataInstance may represent an array or struct in which only the subElements correspond to FlatMap entries.</li> <li>• The McDataInstance may represent a task local buffer for rapid prototyping access which is different from the "main instance" used for measurement access.</li> </ul>





Class	McDataInstance			
instanceInMemory	ImplementationElement InParameterInstance Ref	0..1	aggr	Reference to the corresponding data instance in the description of calibration data structures published by the RTE generator. This is used to support emulation methods inside the ECU, it is not required for A2L generation.
mcDataAccessDetails	McDataAccessDetails	0..1	aggr	Refers to "upstream" information on how the RTE uses this data instance. Use Case: Rapid Prototyping
mcDataAssignment	RoleBasedMcDataAssignment	*	aggr	An assignment between McDataInstances. This supports the indication of related McDataElement implementing the of "RP global buffer", "RP global measurement buffer", "RP enabler flag".
resultingProperties	<a href="#">SwDataDefProps</a>	0..1	aggr	These are the generated properties resulting from decisions taken by the RTE generator for the actually implemented data instance. Only those properties are relevant here, which are needed for the measurement and calibration system.
resultingRptSwPrototypingAccess	RptSwPrototypingAccess	0..1	aggr	Describes the implemented accessibility of data and modes by the rapid prototyping tooling.
role	Identifier	0..1	attr	An optional attribute to be used for additional information on the role of this data instance, for example in the context of rapid prototyping.
rptImplPolicy	RptImplPolicy	0..1	aggr	Describes the implemented code preparation for rapid prototyping at data accesses for a hook based bypassing.
subElement (ordered)	<a href="#">McDataInstance</a>	*	aggr	This relation indicates, that the target element is part of a "struct" which is given by the source element. This information will be used by the final generator to set up the correct addressing scheme.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
symbol	SymbolString	0..1	attr	This String is used to determine the memory address during final generation of the MC configuration data (e.g. "A2L" file) . It shall be the name of the element in the programming language such that it can be identified in linker generated information.  In case the McDataInstance is part of composite data in the programming language, the symbol String may include parts denoting the element context, unless the context is given by the symbol attribute of an enclosing McDataInstance. This means in particular for the C language that the "." character shall be used as a separator between the name of a "struct" variable the name of one of its elements.  The symbol can differ from the shortName in case of generated C data declarations.  It is an optional attribute since it may be missing in case the instance represents an element (e.g. a single array element) which has no name in the linker map.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=symbol

**Table A.38: McDataInstance**

<i>Enumeration</i>	<b>MemoryAllocationKeywordPolicyType</b>
<b>Package</b>	M2::MSR::DataDictionary::AuxillaryObjects
<b>Note</b>	Enumeration to specify the name pattern of the Memory Allocation Keyword.
<b>Literal</b>	<b>Description</b>
addrMethodShort Name	The MemorySection shortNames of referring MemorySections and therefore the belonging Memory Allocation Keywords in the code are build with the shortName of the SwAddrMethod. This is the default value if the attribute does not exist. <b>Tags:</b> atp.EnumerationLiteralIndex=0
addrMethodShort NameAndAlignment	The MemorySection shortNames of referring MemorySections and therefore the belonging Memory Allocation Keywords in the code are build with the shortName of the SwAddrMethod and a variable alignment postfix. Thereby the alignment postfix needs to be consistent with the alignment attribute of the related MemorySection. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table A.39: MemoryAllocationKeywordPolicyType**

<i>Enumeration</i>	<b>MemorySectionType</b>
<b>Package</b>	M2::MSR::DataDictionary::AuxillaryObjects
<b>Note</b>	Enumeration to specify the essential nature of the data which can be allocated in a common memory class by the means of the AUTOSAR Memory Mapping.
<b>Literal</b>	<b>Description</b>
calibrationVariables	This memory section is reserved for "virtual variables" that are computed by an MCD system during a measurement session but do not exist in the ECU memory. <b>Tags:</b> atp.EnumerationLiteralIndex=2
calprm	To be used for calibratable constants of ECU-functions. <b>Tags:</b> atp.EnumerationLiteralIndex=3
code	To be used for mapping code to application block, boot block, external flash etc. <b>Tags:</b> atp.EnumerationLiteralIndex=4
configData	Constants with attributes that show that they reside in one segment for module configuration. <b>Tags:</b> atp.EnumerationLiteralIndex=5
const	To be used for global or static constants. <b>Tags:</b> atp.EnumerationLiteralIndex=6
excludeFromFlash	This memory section is reserved for "virtual parameters" that are taken for computing the values of so-called dependent parameter of an MCD system. Dependent Parameters that are not at the same time "virtual parameters" are allocated in the ECU memory. Virtual parameters, on the other hand, are not allocated in the ECU memory. Virtual parameters exist in the ECU Hex file for the purpose of being considered (for computing the values of dependent parameters) during an offline-calibration session. <b>Tags:</b> atp.EnumerationLiteralIndex=7
var	To be used for global or static variables. The expected initialization is specified with the attribute sectionInitializationPolicy. <b>Tags:</b> atp.EnumerationLiteralIndex=9

**Table A.40: MemorySectionType**

<b>Class</b>	<b>ModeDeclaration</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
<b>Note</b>	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model. <b>Tags:</b> atp.ManifestKind=ExecutionManifest,MachineManifest			
<b>Base</b>	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this Mode Declaration.

**Table A.41: ModeDeclaration**

<b>Class</b>	<b>ModeDeclarationGroup</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
<b>Note</b>	A collection of Mode Declarations. Also, the initial mode is explicitly identified. <b>Tags:</b> atp.ManifestKind=ExecutionManifest,MachineManifest atp.recommendedPackage=ModeDeclarationGroups			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
initialMode	<a href="#">ModeDeclaration</a>	1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred.
mode Declaration	<a href="#">ModeDeclaration</a>	1..*	aggr	The ModeDeclarations collected in this ModeDeclaration Group. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=blueprintDerivationTime
modeManager ErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the ability to define the error behavior expected by the mode manager in case of errors on the mode user side (e.g. terminated mode user).
modeTransition	ModeTransition	*	aggr	This represents the available ModeTransitions of the ModeDeclarationGroup
modeUserError Behavior	ModeErrorBehavior	0..1	aggr	This represents the definition of the error behavior expected by the mode user in case of errors on the mode manager side (e.g. terminated mode manager).
onTransition Value	PositiveInteger	0..1	attr	The value of this attribute shall be taken into account by the RTE generator for programmatically representing a value used for the transition between two statuses.

**Table A.42: ModeDeclarationGroup**

<b>Class</b>	<b>ModeDeclarationGroupPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
<b>Note</b>	The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context. <b>Tags:</b> atp.ManifestKind=ExecutionManifest,MachineManifest			
<b>Base</b>	ARObject, AtpFeature, AtpPrototype, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	ModeDeclarationGroupPrototype			
swCalibration Access	<a href="#">SwCalibrationAccess Enum</a>	0..1	attr	This allows for specifying whether or not the enclosing ModeDeclarationGroupPrototype can be measured at run-time.
type	<a href="#">ModeDeclarationGroup</a>	1	tref	The "collection of ModeDeclarations" (= ModeDeclaration Group) supported by a component <b>Stereotypes:</b> isOfType

**Table A.43: ModeDeclarationGroupPrototype**

Class	ModeSwitchInterface			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A mode switch interface declares a ModeDeclarationGroupPrototype to be sent and received. <b>Tags:</b> atp.recommendedPackage=PortInterfaces			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
modeGroup	<a href="#">ModeDeclarationGroup Prototype</a>	1	aggr	The ModeDeclarationGroupPrototype of this mode interface.

**Table A.44: ModeSwitchInterface**

Class	NumericalValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	A numerical ValueSpecification which is intended to be assigned to a Primitive data element. Note that the numerical value is a variant, it can be computed by a formula.			
Base	<a href="#">ARObject</a> , <a href="#">ValueSpecification</a>			
Attribute	Type	Mult.	Kind	Note
value	Numerical	1	attr	This is the value itself. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table A.45: NumericalValueSpecification**

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

**Table A.46: PPortPrototype**



<b>Class</b>	<b>ParameterAccess</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements			
<b>Note</b>	The presence of a ParameterAccess implies that a RunnableEntity needs access to a ParameterData Prototype.			
<b>Base</b>	ARObject, AbstractAccessPoint, AtpClassifier, AtpFeature, AtpStructureElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
accessed Parameter	AutosarParameterRef	1	aggr	Reference to the accessed calibration parameter.
swDataDef Props	<a href="#">SwDataDefProps</a>	0..1	aggr	This allows denote instance and access specific properties, mainly input values and common axis.

**Table A.47: ParameterAccess**

<b>Class</b>	<b>ParameterDataPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
<b>Note</b>	A parameter element used for parameter interface and internal behavior, supporting signal like parameter and characteristic value communication patterns and parameter and characteristic value definition.			
<b>Base</b>	ARObject, AtpFeature, AtpPrototype, <a href="#">AutosarDataPrototype</a> , <a href="#">DataPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
initValue	<a href="#">ValueSpecification</a>	0..1	aggr	Specifies initial value(s) of the ParameterDataPrototype

**Table A.48: ParameterDataPrototype**

<b>Class</b>	<b>ParameterInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	A parameter interface declares a number of parameter and characteristic values to be exchanged between parameter components and software components. <b>Tags:</b> atp.recommendedPackage=PortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, <a href="#">CollectableElement</a> , <a href="#">DataInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
parameter	<a href="#">ParameterDataPrototype</a>	1..*	aggr	The ParameterDataPrototype of this ParameterInterface.

**Table A.49: ParameterInterface**

<b>Class</b>	<b>ParameterProvideComSpec</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
<b>Note</b>	"Communication" specification that applies to parameters on the provided side of a connection.			
<b>Base</b>	ARObject, PPortComSpec			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
initValue	<a href="#">ValueSpecification</a>	0..1	aggr	The initial value applicable for the corresponding ParameterDataPrototype.





<b>Class</b>	<b>ParameterProvideComSpec</b>			
parameter	<a href="#">ParameterData Prototype</a>	1	ref	The ParameterDataPrototype to which the Parameter ComSpec applies.

**Table A.50: ParameterProvideComSpec**

<b>Class</b>	<b>ParameterSwComponentType</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	The ParameterSwComponentType defines parameters and characteristic values accessible via provided Ports. The provided values are the same for all connected SwComponentPrototypes <b>Tags:</b> atp.recommendedPackage=SwComponentTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a> , <a href="#">SwComponentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
constant Mapping	ConstantSpecification MappingSet	*	ref	Reference to the ConstanSpecificationMapping to be applied for the particular ParameterSwComponentType <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=constantMapping
dataType Mapping	<a href="#">DataTypeMappingSet</a>	*	ref	Reference to the DataTypeMapping to be applied for the particular ParameterSwComponentType <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=dataTypeMapping
instantiation DataDefProps	<a href="#">InstantiationDataDef Props</a>	*	aggr	The purpose of this is that within the context of a given SwComponentType some data def properties of individual instantiations can be modified. The aggregation of InstantiationDataDefProps is subject to variability with the purpose to support the conditional existence of PortPrototypes <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table A.51: ParameterSwComponentType**

<b>Class</b>	<b>PhysConstrs</b>			
<b>Package</b>	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
<b>Note</b>	This meta-class represents the ability to express physical constraints. Therefore it has (in opposite to InternalConstrs) a reference to a Unit.			
<b>Base</b>	<a href="#">ARObject</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the constraint. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
maxDiff	Numerical	0..1	attr	Maximum difference that is permitted between two consecutive values if the constraint is applied to an axis. <b>Tags:</b> xml.sequenceOffset=60





Class	PhysConstrs			
maxGradient	Numerical	0..1	attr	This element specifies the maximum slope that may be used in curves and maps. <b>Tags:</b> xml.sequenceOffset=50
monotony	MonotonyEnum	0..1	attr	This specifies the monotony constraints on the data object. Note that this applies only to curves and maps. <b>Tags:</b> xml.sequenceOffset=70
scaleConstr (ordered)	ScaleConstr	*	aggr	This is one particular scale which contributes to the data constraints. <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=40 xml.typeElement=false xml.typeWrapperElement=false
unit	<a href="#">Unit</a>	0..1	ref	This is the unit to which the physical constraints relate to. In particular, it is the physical unit of the specified limits. <b>Tags:</b> xml.sequenceOffset=80
upperLimit	Limit	0..1	attr	This specifies the upper limit of the constraint. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

**Table A.52: PhysConstrs**

Class	PhysicalDimension			
<b>Package</b>	M2::MSR::AsamHdo::Units			
<b>Note</b>	<p>This class represents a physical dimension. If the physical dimension of two units is identical, then a conversion between them is possible. The conversion between units is related to the definition of the physical dimension.</p> <p>Note that the equivalence of the exponents does not per se define the convertibility. For example Energy and Torque share the same exponents (Nm).</p> <p>Please note further the value of an exponent does not necessarily have to be an integer number. It is also possible that the value yields a rational number, e.g. to compute the square root of a given physical quantity. In this case the exponent value would be a rational number where the numerator value is 1 and the denominator value is 2.</p> <p><b>Tags:</b>atp.recommendedPackage=PhysicalDimensions</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
currentExp	Numerical	0..1	attr	This attribute represents the exponent of the physical dimension "electric current". <b>Tags:</b> xml.sequenceOffset=50
lengthExp	Numerical	0..1	attr	The exponent of the physical dimension "length". <b>Tags:</b> xml.sequenceOffset=20
luminous IntensityExp	Numerical	0..1	attr	The exponent of the physical dimension "luminous intensity". <b>Tags:</b> xml.sequenceOffset=80





<b>Class</b>	<b>PhysicalDimension</b>			
massExp	Numerical	0..1	attr	The exponent of the physical dimension "mass". <b>Tags:</b> xml.sequenceOffset=30
molarAmountExp	Numerical	0..1	attr	The exponent of the physical dimension "quantity of substance". <b>Tags:</b> xml.sequenceOffset=70
temperatureExp	Numerical	0..1	attr	The exponent of the physical dimension "temperature". <b>Tags:</b> xml.sequenceOffset=60
timeExp	Numerical	0..1	attr	The exponent of the physical dimension "time". <b>Tags:</b> xml.sequenceOffset=40

**Table A.53: PhysicalDimension**

<b>Class</b>	<b>PortInterface</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	Abstract base class for an interface that is either provided or required by a port of a software component.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	ClientServerInterface, <a href="#">DataInterface</a> , <a href="#">ModeSwitchInterface</a> , TriggerInterface			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
isService	Boolean	1	attr	This flag is set if the PortInterface is to be used for communication between an <ul style="list-style-type: none"> <li>• ApplicationSwComponentType or</li> <li>• ServiceProxySwComponentType or</li> <li>• SensorActuatorSwComponentType or</li> <li>• ComplexDeviceDriverSwComponentType</li> <li>• ServiceSwComponentType</li> <li>• EcuAbstractionSwComponentType</li> </ul> and a ServiceSwComponentType (namely an AUTOSAR Service) located on the same ECU. Otherwise the flag is not set.
serviceKind	ServiceProviderEnum	0..1	attr	This attribute provides further details about the nature of the applied service.

**Table A.54: PortInterface**

<b>Class</b>	<b>PortPrototype</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">AbstractProvidedPortPrototype</a> , <a href="#">AbstractRequiredPortPrototype</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	PortPrototype (abstract)			
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non volatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

**Table A.55: PortPrototype**

Class	RPortPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	ARObject, AbstractRequiredPortPrototype, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, PortPrototype, Referrable			
Attribute	Type	Mult.	Kind	Note
required Interface	PortInterface	1	tref	The interface that this port requires. <b>Stereotypes:</b> isOfType

**Table A.56: RPortPrototype**

Class	RTEEvent (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	Abstract base class for all RTE-related events			
Base	ARObject, AbstractEvent, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, Multilanguage Referrable, Referrable			
Subclasses	AsynchronousServerCallReturnsEvent, BackgroundEvent, DataReceiveErrorEvent, DataReceivedEvent, DataSendCompletedEvent, DataWriteCompletedEvent, ExternalTriggerOccurredEvent, InitEvent, InternalTriggerOccurredEvent, ModeSwitchedAckEvent, OperationInvokedEvent, SwcModeManagerError Event, SwcModeSwitchEvent, TimingEvent, TransformerHardErrorEvent			
Attribute	Type	Mult.	Kind	Note
disabledMode	ModeDeclaration	*	iref	Reference to the Modes that disable the Event. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=contextPort, contextModeDeclaration GroupPrototype, targetModeDeclaration





<b>Class</b>	<b>RTEEvent</b> (abstract)			
startOnEvent	<a href="#">RunnableEntity</a>	0..1	ref	RunnableEntity starts when the corresponding RTEEvent occurs.

**Table A.57: RTEEvent**

<b>Primitive</b>	<b>Ref</b>			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
<b>Note</b>	This primitive denotes a name based reference. For detailed syntax see the xsd.pattern. <ul style="list-style-type: none"> <li>• first slash (relative or absolute reference) [optional]</li> <li>• Identifier [required]</li> <li>• a sequence of slashes and Identifiers [optional]</li> </ul> This primitive is used by the meta-model tools to create the references. <b>Tags:</b> xml.xsd.customType=REF xml.xsd.pattern=/?[a-zA-Z][a-zA-Z0-9_]{0,127}/([a-zA-Z][a-zA-Z0-9_]{0,127})* xml.xsd.type=string			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
base	Identifier	0..1	attr	This attribute reflects the base to be used for this reference. <b>Tags:</b> xml.attribute=true
blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint. <b>Tags:</b> atp.Status=draft xml.attribute=true
index	PositiveInteger	0..1	attr	This attribute supports the use case to point on specific elements in an array. This is in particular required if arrays are used to implement particular data objects. <b>Tags:</b> xml.attribute=true

**Table A.58: Ref**

<b>Class</b>	<b>Referrable</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
<b>Note</b>	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
<b>Base</b>	<i>ARObject</i>			
<b>Subclasses</b>	<i>AtpDefinition, BswDistinguishedPartition, BswModuleCallPoint, BswModuleClientServerEntry, BswVariableAccess, CouplingPortTrafficClassAssignment, DiagnosticDebounceAlgorithmProps, DiagnosticEnvModeElement, EthernetPriorityRegeneration, EventHandler, ExclusiveAreaNestingOrder, HwDescriptionEntity, ImplementationProps, LinSlaveConfigIdent, ModeTransition, MultilanguageReferrable, PduActivationRoutingGroup, PncMappingIdent, SingleLanguageReferrable, SoConIPduIdentifier, SocketConnectionBundle, TimeSyncServerConfiguration, TpConnectionIdent</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	Referrable (abstract)			
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.  <b>Tags:</b> xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments.  <b>Tags:</b> xml.sequenceOffset=-90

**Table A.59: Referrable**

Class	RootSwCompositionPrototype			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	<p>The RootSwCompositionPrototype represents the top-level-composition of software components within a given System. According to the use case of the System, this may for example be the a more or less complete VFB description, the software of a System Extract or the software of a flat ECU Extract with only atomic SWCs.</p> <p>Therefore the RootSwComposition will only occasionally contain all atomic software components that are used in a complete VFB System. The OEM is primarily interested in the required functionality and the interfaces defining the integration of the Software Component into the System. The internal structure of such a component contains often substantial intellectual property of a supplier. Therefore a top-level software composition will often contain empty compositions which represent subsystems.</p> <p>The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including Port Prototypes, PortInterfaces, VariableDataPrototypes, SwcInternalBehavior etc.), and their ports are interconnected using SwConnectorPrototypes.</p>			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	Type	Mult.	Kind	Note
calibration ParameterValue Set	CalibrationParameter ValueSet	*	ref	Used CalibrationParameterValueSet for instance specific initialization of calibration parameters.  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=calibrationParameterValueSet
flatMap	FlatMap	0..1	ref	The FlatMap used in the scope of this RootSw CompositionPrototype.  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=flatMap
software Composition	CompositionSw ComponentType	1	tref	We assume that there is exactly one top-level composition that includes all Component instances of the system  <b>Stereotypes:</b> isOfType

**Table A.60: RootSwCompositionPrototype**

Class	RunnableEntity			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
Note	A RunnableEntity represents the smallest code-fragment that is provided by an AtomicSwComponent Type and are executed under control of the RTE. RunnableEntities are for instance set up to respond to data reception or operation invocation on a server.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, ExecutableEntity, Identifiable, Multilanguage Referrable, Referrable			





<b>Class</b>	<b>RunnableEntity</b>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
argument (ordered)	RunnableEntity Argument	*	aggr	This represents the formal definition of a an argument to a RunnableEntity.
asynchronous ServerCall ResultPoint	AsynchronousServer CallResultPoint	*	aggr	<p>The server call result point admits a runnable to fetch the result of an asynchronous server call.</p> <p>The aggregation of AsynchronousServerCallResultPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes and the variant existence of server call result points in the implementation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=shortName, variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
canBelInvoked Concurrently	Boolean	1	attr	If the value of this attribute is set to "true" the enclosing RunnableEntity can be invoked concurrently (even for one instance of the corresponding AtomicSwComponent Type). This implies that it is the responsibility of the implementation of the RunnableEntity to take care of this form of concurrency. Note that the default value of this attribute is set to "false".
dataRead Access	VariableAccess	*	aggr	<p>RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.</p> <p>The aggregation of dataReadAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataReadAccess in the implementation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=shortName, variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
dataReceive PointBy Argument	VariableAccess	*	aggr	<p>RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of an argument in the function signature.</p> <p>The aggregation of dataReceivePointByArgument is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data receive points in the implementation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=shortName, variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
dataReceive PointByValue	VariableAccess	*	aggr	<p>RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.</p> <p>The result is passed back to the application by means of the return value. The aggregation of dataReceivePointBy Value is subject to variability with the purpose to support the conditional existence of sender receiver ports or the</p>







Class	RunnableEntity			
				variant existence of data receive points in the implementation. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataSendPoint	VariableAccess	*	aggr	RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataSendPoint is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data send points in the implementation. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataWrite Access	VariableAccess	*	aggr	RunnableEntity has implicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataWriteAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataWriteAccess in the implementation. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
external TriggeringPoint	ExternalTriggeringPoint	*	aggr	The aggregation of ExternalTriggeringPoint is subject to variability with the purpose to support the conditional existence of trigger ports or the variant existence of external triggering points in the implementation. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=ident.shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
internal TriggeringPoint	InternalTriggeringPoint	*	aggr	The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
modeAccess Point	ModeAccessPoint	*	aggr	The runnable has a mode access point. The aggregation of ModeAccessPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode access points in the implementation. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=ident.shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime





Class	RunnableEntity			
modeSwitch Point	ModeSwitchPoint	*	aggr	<p>The runnable has a mode switch point. The aggregation of ModeSwitchPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode switch points in the implementation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=shortName, variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
parameter Access	ParameterAccess	*	aggr	<p>The presence of a ParameterAccess implies that a RunnableEntity needs read only access to a Parameter DataPrototype which may either be local or within a Port Prototype.</p> <p>The aggregation of ParameterAccess is subject to variability with the purpose to support the conditional existence of parameter ports and component local parameters as well as the variant existence of Parameter Access (points) in the implementation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=shortName, variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
readLocal Variable	VariableAccess	*	aggr	<p>The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable.</p> <p>The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of read LocalVariable (points) in the implementation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=shortName, variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
serverCallPoint	ServerCallPoint	*	aggr	<p>The RunnableEntity has a ServerCallPoint. The aggregation of ServerCallPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes or the variant existence of server call points in the implementation.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation  <b>Tags:</b>                      atp.Splitkey=shortName, variationPoint.shortLabel                      vh.latestBindingTime=preCompileTime</p>
symbol	CIdentifier	1	attr	<p>The symbol describing this RunnableEntity's entry point. This is considered the API of the RunnableEntity and is required during the RTE contract phase.</p>
waitPoint	WaitPoint	*	aggr	<p>The WaitPoint associated with the RunnableEntity.</p>
writtenLocal Variable	VariableAccess	*	aggr	<p>The presence of a writtenLocalVariable implies that a RunnableEntity needs write access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable.</p> <p>The aggregation of writtenLocalVariable is subject to variability with the purpose to support the conditional</p>





<b>Class</b>	<b>RunnableEntity</b>			
				existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of written LocalVariable (points) in the implementation. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

**Table A.61: RunnableEntity**

<b>Class</b>	<b>SenderReceiverInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	A sender/receiver interface declares a number of data elements to be sent and received. <b>Tags:</b> atp.recommendedPackage=PortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DataInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataElement	<a href="#">VariableDataPrototype</a>	1..*	aggr	The data elements of this SenderReceiverInterface.
invalidation Policy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement
metaDataItem Set	MetaDataItemSet	*	aggr	This aggregation defines fixed sets of meta-data items associated with dataElements of the enclosing Sender ReceiverInterface

**Table A.62: SenderReceiverInterface**

<b>Class</b>	<b>StructuredReq</b>			
<b>Package</b>	M2::MSR::Documentation::BlockElements::RequirementsTracing			
<b>Note</b>	This represents a structured requirement. This is intended for a case where specific requirements for features are collected. Note that this can be rendered as a labeled list.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">DocumentViewSelectable</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Paginateable</a> , <a href="#">Referrable</a> , <a href="#">Traceable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
appliesTo	standardNameEnum	*	attr	This attribute represents the platform the requirement is assigned to. <b>Tags:</b> xml.namePlural=APPLIES-TO-DEPENDENCIES xml.sequenceOffset=25
conflicts	DocumentationBlock	0..1	aggr	This represents an informal specification of conflicts. <b>Tags:</b> xml.sequenceOffset=40
date	DateTime	1	attr	This represents the date when the requirement was initiated. <b>Tags:</b> xml.sequenceOffset=5





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

**Table A.63: StructuredReq**

<b>Class</b>	<b>SwAddrMethod</b>			
<b>Package</b>	M2::MSR::DataDictionary::AuxillaryObjects			
<b>Note</b>	Used to assign a common addressing method, e.g. common memory section, to data or code objects. These objects could actually live in different modules or components. <b>Tags:</b> atp.recommendedPackage=SwAddrMethods			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	SwAddrMethod			
memory Allocation KeywordPolicy	<a href="#">MemoryAllocationKeywordPolicyType</a>	0..1	attr	Enumeration to specify the name pattern of the Memory Allocation Keyword.
option	Identifier	*	attr	<p>This attribute introduces the ability to specify further intended properties of the MemorySection in with the related objects shall be placed.</p> <p>These properties are handled as to be selected. The intended options are mentioned in the list.</p> <p>In the Memory Mapping configuration, this option list is used to determine an appropriate MemMapAddressing ModeSet.</p>
section Initialization Policy	SectionInitializationPolicyType	0..1	attr	<p>Specifies the expected initialization of the variables (inclusive those which are implementing VariableData Prototypes). Therefore this is an implementation constraint for initialization code of BSW modules (especially RTE) as well as the start-up code which initializes the memory segment to which the AutosarData Prototypes referring to the SwAddrMethod's are later on mapped.</p> <p>If the attribute is not defined it has the identical semantic as the attribute value "INIT"</p>
sectionType	<a href="#">MemorySectionType</a>	0..1	attr	Defines the type of memory sections which can be associated with this addressing method.

**Table A.64: SwAddrMethod**

Class	SwAxisGrouped			
<b>Package</b>	M2::MSR::DataDictionary::Axis			
<b>Note</b>	An SwAxisGrouped is an axis which is shared between multiple calibration parameters.			
<b>Base</b>	<i>ARObject</i> , <i>SwCalprmAxisTypeProps</i>			
Attribute	Type	Mult.	Kind	Note
sharedAxisType	<a href="#">ApplicationPrimitiveDataType</a>	0..1	ref	This is the datatype of the calibration parameter providing the shared axis.
swAxisIndex	AxisIndexType	0..1	attr	<p>Describes which axis of the referenced calibration parameter provides the values for the group axis. The index satisfies the following convention:</p> <ul style="list-style-type: none"> <li>0 = value axis. in this case, the interpolation result of the referenced parameter is used as a base point index.</li> <li>The index should only be specified if the parameter under swCalprm contains more than one axis. It is standard practice for the axis index of parameters with more than one axis, to be set to 1, if data has not been assigned to swAxis Index.</li> </ul> <p><b>Tags:</b>xml.sequenceOffset=20</p>





Class	SwAxisGrouped			
swCalprmRef	SwCalprmRefProxy	1	aggr	This property specifies the calibration parameter which serves as the input axis. In AUTOSAR, the type of the referenced Calibration parameter shall be compatible to the type specified by sharedAxisType.  <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

**Table A.65: SwAxisGrouped**

Class	SwAxisIndividual			
<b>Package</b>	M2::MSR::DataDictionary::Axis			
<b>Note</b>	This meta-class describes an axis integrated into a parameter (field etc.). The integration makes this individual to each parameter. The so-called grouped axis represents the counterpart to this. It is conceived as an independent parameter (see class SwAxisGrouped).			
<b>Base</b>	ARObject, SwCalprmAxisTypeProps			
Attribute	Type	Mult.	Kind	Note
compuMethod	<a href="#">CompuMethod</a>	0..1	ref	This is the compuMethod which is expected for the axis. It is used in early stages if the particular input-value is not yet available.  <b>Tags:</b> xml.sequenceOffset=30
dataConstr	<a href="#">DataConstr</a>	0..1	ref	Refers to constraints, e.g. for plausibility checks.  <b>Tags:</b> xml.sequenceOffset=80
inputVariableType	<a href="#">ApplicationPrimitive</a> <a href="#">DataType</a>	0..1	ref	This is the datatype of the input value for the axis. This allows to define e.g. a type of curve, where the input value is finalized at the access point.
swAxisGeneric	SwAxisGeneric	0..1	aggr	this specifies the properties of a generic axis if applicable.  <b>Tags:</b> xml.sequenceOffset=90
swMaxAxisPoints	Integer	1	attr	Maximum number of base points contained in the axis of a map or curve.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
swMinAxisPoints	Integer	1	attr	Minimum number of base points contained in the axis of a map or curve.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=70
swVariableRef (ordered)	SwVariableRefProxy	*	aggr	Refers to input variables of the axis. It is possible to specify more than one variable. Here the following is valid: <ul style="list-style-type: none"> <li>The variable with the highest priority shall be given first. It is used in the generation of the code and is also displayed first in the application system.</li> </ul>





Class	SwAxisIndividual			
				<p style="text-align: center;">△</p> <ul style="list-style-type: none"> <li>All variables referenced shall be of the same physical nature. This is usually detected in that the conversion formulae affected refer back to the same SI-units.</li> </ul> <p>In AUTOSAR this ensured by the constraint, that the referenced input variables shall use a type compatible to "inputVariableType".</p> <ul style="list-style-type: none"> <li>This multiple referencing allows a base point distribution for more than one input variable to be used. One example of this are the temperature curves which can depend both on the induction air temperature and the engine temperature.</li> </ul> <p>These variables can be displayed simultaneously by MCD systems (adjustment systems), enabling operating points to be shown in the curves.</p> <p><b>Tags:</b>                      xml.roleElement=false                      xml.roleWrapperElement=true                      xml.sequenceOffset=20                      xml.typeElement=false                      xml.typeWrapperElement=false</p>
unit	<a href="#">Unit</a>	0..1	ref	<p>This represents the physical unit of the input value of the axis. It is provided to support the case that the particular input variable is not yet known.</p> <p><b>Tags:</b>xml.sequenceOffset=40</p>

**Table A.66: SwAxisIndividual**

Class	SwBaseType			
<b>Package</b>	M2::MSR::AsamHdo::BaseTypes			
<b>Note</b>	This meta-class represents a base type used within ECU software. <b>Tags:</b> atp.recommendedPackage=BaseTypes			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">BaseType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.67: SwBaseType**

Enumeration	SwCalibrationAccessEnum			
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties			
<b>Note</b>	Determines the access rights to a data object w.r.t. measurement and calibration.			
<b>Literal</b>	<b>Description</b>			
notAccessible	The element will not be accessible via MCD tools, i.e. will not appear in the ASAP file. <b>Tags:</b> atp.EnumerationLiteralIndex=0			
readOnly	The element will only appear as read-only in an ASAP file. <b>Tags:</b> atp.EnumerationLiteralIndex=1			





<b>Enumeration</b>	<b>SwCalibrationAccessEnum</b>
readWrite	The element will appear in the ASAP file with both read and write access. <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table A.68: SwCalibrationAccessEnum**

<b>Class</b>	<b>SwCalprmAxis</b>			
<b>Package</b>	M2::MSR::DataDictionary::CalibrationParameter			
<b>Note</b>	This element specifies an individual input parameter axis (abscissa).			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
baseType	SwBaseType	0..1	ref	The SwBaseType to be used for the axis. Note that this is not applicable for ApplicationDataTypes. The value shall be ignored. <b>Tags:</b> atp.Status=removed xml.sequenceOffset=110
category	CalprmAxisCategory Enum	0..1	attr	This property specifies the category of a particular axis. <b>Tags:</b> xml.sequenceOffset=30
displayFormat	DisplayFormatString	0..1	attr	This property specifies how the axis values shall be displayed e.g. in documents or in measurement and calibration tools. <b>Tags:</b> xml.sequenceOffset=100
swAxisIndex	AxisIndexType	0..1	attr	This attribute specifies which axis is specified by the containing SwCalprmAxis. For example in a curve this is usually "1". In a map this is "1" or "2". <b>Tags:</b> xml.sequenceOffset=20
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	Describes the applicability of parameters and variables. <b>Tags:</b> xml.sequenceOffset=90
swCalprmAxis TypeProps	SwCalprmAxisType Props	1	aggr	specific properties depending on the type of the axis. <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=40 xml.typeElement=true xml.typeWrapperElement=false

**Table A.69: SwCalprmAxis**

<b>Class</b>	<b>SwCalprmAxisSet</b>			
<b>Package</b>	M2::MSR::DataDictionary::CalibrationParameter			
<b>Note</b>	This element specifies the input parameter axes (abscissas) of parameters (and variables, if these are used adaptively).			
<b>Base</b>	ARObject			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>







Class	SwCalprmAxisSet			
swCalprmAxis	<a href="#">SwCalprmAxis</a>	*	aggr	One axis belonging to this SwCalprmAxisSet <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

**Table A.70: SwCalprmAxisSet**

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

**Table A.71: SwComponentPrototype**

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





Class	SwComponentType (abstract)			
swComponentDocumentation	SwComponentDocumentation	0..1	aggr	This adds a documentation to the SwComponentType. <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=swComponentDocumentation, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10
unitGroup	UnitGroup	*	ref	This allows for the specification of which UnitGroups are relevant in the context of referencing SwComponentType.

**Table A.72: SwComponentType**

Class	<<atpVariation>> SwDataDefProps			
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	<p>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.</p> <p>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.</p> <p>SwDataDefProps covers various aspects:</p> <ul style="list-style-type: none"> <li>• 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 Data Types in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet</li> <li>• Implementation aspects, mainly expressed by swImpIPolicy, swVariableAccessImpIPolicy, swAddrMethod, swPointerTargetProps, baseType, implementationDataType and additionalNativeTypeQualifier</li> <li>• Access policy for the MCD system, mainly expressed by swCalibrationAccess</li> <li>• Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalidValue</li> <li>• Code generation policy provided by swRecordLayout</li> </ul> <p><b>Tags:</b>vh.latestBindingTime=codeGenerationTime</p>			
Base	ARObject			
Attribute	Type	Mult.	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. <b>Tags:</b> xml.sequenceOffset=235
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false





Class	<<atpVariation>> SwDataDefProps			
baseType	<a href="#">SwBaseType</a>	0..1	ref	Base type associated with the containing data object. <b>Tags:</b> xml.sequenceOffset=50
compuMethod	<a href="#">CompuMethod</a>	0..1	ref	Computation method associated with the semantics of this data object. <b>Tags:</b> xml.sequenceOffset=180
dataConstr	<a href="#">DataConstr</a>	0..1	ref	Data constraint for this data object. <b>Tags:</b> 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. <b>Tags:</b> xml.sequenceOffset=210
display Presentation	DisplayPresentation Enum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.
implementation DataType	AbstractImplementation DataType	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 <ul style="list-style-type: none"> <li>• redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype</li> <li>• the target type of a pointer (see SwPointerTarget Props), if it does not refer to a base type directly</li> <li>• the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly</li> <li>• the data type of an SwServiceArg, if it does not refer to a base type directly</li> </ul> <b>Tags:</b> xml.sequenceOffset=215
invalidValue	<a href="#">ValueSpecification</a>	0..1	aggr	Optional value to express invalidity of the actual data element. <b>Tags:</b> 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	<a href="#">SwAddrMethod</a>	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. <b>Tags:</b> 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 memory AllocationKeywordPolicy of the referenced SwAddr Method. <b>Tags:</b> xml.sequenceOffset=33
swBit Representation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. <b>Tags:</b> xml.sequenceOffset=60





Class	<<atpVariation>> SwDataDefProps			
swCalibrationAccess	<a href="#">SwCalibrationAccessEnum</a>	0..1	attr	Specifies the read or write access by MCD tools for this data object. <b>Tags:</b> xml.sequenceOffset=70
swCalprmAxisSet	<a href="#">SwCalprmAxisSet</a>	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. <b>Tags:</b> xml.sequenceOffset=90
swComparisonVariable	<a href="#">SwVariableRefProxy</a>	*	aggr	Variables used for comparison in an MCD process. <b>Tags:</b> xml.sequenceOffset=170 xml.typeElement=false
swDataDependency	<a href="#">SwDataDependency</a>	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). <b>Tags:</b> xml.sequenceOffset=200
swHostVariable	<a href="#">SwVariableRefProxy</a>	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. <b>Tags:</b> xml.sequenceOffset=220 xml.typeElement=false
swImplPolicy	<a href="#">SwImplPolicyEnum</a>	0..1	attr	Implementation policy for this data object. <b>Tags:</b> 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". <b>Tags:</b> 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. <b>Tags:</b> 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 . <b>Tags:</b> xml.sequenceOffset=260
swPointerTargetProps	<a href="#">SwPointerTargetProps</a>	0..1	aggr	Specifies that the containing data object is a pointer to another data object. <b>Tags:</b> xml.sequenceOffset=280





Class	<<atpVariation>> SwDataDefProps			
swRecordLayout	SwRecordLayout	0..1	ref	Record layout for this data object. <b>Tags:</b> 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. <b>Tags:</b> xml.sequenceOffset=300
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. <b>Tags:</b> xml.sequenceOffset=120
swValueBlockSize	Numerical	0..1	attr	This represents the size of a Value Block <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
swValueBlockSizeMult (ordered)	Numerical	*	attr	This attribute is used to specify the dimensions of a value block (VAL_BLK) for the case that that value block has more than one dimension.  The dimensions given in this attribute are ordered such that the first entry represents the first dimension, the second entry represents the second dimension, and so on.  For one-dimensional value blocks the attribute swValueBlockSize shall be used and this attribute shall not exist. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
unit	<a href="#">Unit</a>	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. <b>Tags:</b> xml.sequenceOffset=350
valueAxisDataType	<a href="#">ApplicationPrimitiveDataType</a>	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. <b>Tags:</b> xml.sequenceOffset=355

**Table A.73: SwDataDefProps**

<b>Enumeration</b>	SwImpPolicyEnum
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties
<b>Note</b>	Specifies the implementation strategy with respect to consistency mechanisms of variables.
<b>Literal</b>	<b>Description</b>
const	forced implementation such that the running software within the ECU shall not modify it. For example implemented with the "const" modifier in C. This can be applied for parameters (not for those in NVRAM) as well as argument data prototypes. <b>Tags:</b> atp.EnumerationLiteralIndex=0





Enumeration	SwImplPolicyEnum
fixed	This data element is fixed. In particular this indicates, that it might also be implemented e.g. as in place data, (#DEFINE). <b>Tags:</b> atp.EnumerationLiteralIndex=1
measurementPoint	The data element is created for measurement purposes only. The data element is never read directly within the ECU software. In contrast to a "standard" data element in an unconnected provide port is, this unconnection is guaranteed for measurementPoint data elements. <b>Tags:</b> atp.EnumerationLiteralIndex=2
queued	The content of the data element is queued and the data element has 'event' semantics, i.e. data elements are stored in a queue and all data elements are processed in 'first in first out' order. The queuing is intended to be implemented by RTE Generator. This value is not applicable for parameters. <b>Tags:</b> atp.EnumerationLiteralIndex=3
standard	This is applicable for all kinds of data elements. For variable data prototypes the 'last is best' semantics applies. For parameter there is no specific implementation directive. <b>Tags:</b> atp.EnumerationLiteralIndex=4

**Table A.74: SwImplPolicyEnum**

Class	SwcImplementation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcImplementation			
Note	This meta-class represents a specialization of the general Implementation meta-class with respect to the usage in application software. <b>Tags:</b> atp.recommendedPackage=SwcImplementations			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">Implementation</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
behavior	<a href="#">SwcInternalBehavior</a>	1	ref	The internal behavior implemented by this Implementation.
perInstanceMemorySize	PerInstanceMemorySize	*	aggr	Allows a definition of the size of the per-instance memory for this implementation. The aggregation of PerInstanceMemorySize is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects, in this case PerInstanceMemory. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
requiredRTEVendor	String	0..1	attr	Identify a specific RTE vendor. This information is potentially important at the time of integrating (in particular: linking) the application code with the RTE. The semantics is that (if the association exists) the corresponding code has been created to fit to the vendor-mode RTE provided by this specific vendor. Attempting to integrate the code with another RTE generated in vendor mode is in general not possible.

**Table A.75: SwcImplementation**

<b>Class</b>	<b>SwcInternalBehavior</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
<b>Note</b>	The SwcInternalBehavior of an AtomicSwComponentType describes the relevant aspects of the software-component with respect to the RTE, i.e. the RunnableEntities and the RTEEvents they respond to.			
<b>Base</b>	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, <a href="#">Identifiable</a> , <a href="#">InternalBehavior</a> , <a href="#">Multilanguage</a> , <a href="#">Referrable</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
arTypedPerInstanceMemory	<a href="#">VariableDataPrototype</a>	*	aggr	<p>Defines an AUTOSAR typed memory-block that needs to be available for each instance of the SW-component.</p> <p>This is typically only useful if supportsMultipleInstantiation is set to "true" or if the component defines NVRAM access via permanent blocks.</p> <p>The aggregation of arTypedPerInstanceMemory is subject to variability with the purpose to support variability in the software component's implementations. Typically different algorithms in the implementation are requiring different number of memory objects.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
event	<a href="#">RTEEvent</a>	*	aggr	<p>This is a RTEEvent specified for the particular Swc InternalBehavior.</p> <p>The aggregation of RTEEvent is subject to variability with the purpose to support the conditional existence of RTE events. Note: the number of RTE events might vary due to the conditional existence of PortPrototypes using Data ReceivedEvents or due to different scheduling needs of algorithms.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
exclusiveAreaPolicy	SwcExclusiveAreaPolicy	*	aggr	<p>Options how to generate the ExclusiveArea related APIs. When no SwcExclusiveAreaPolicy is specified for an ExclusiveArea the default values apply.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=exclusiveAreaPolicy, variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>
explicitInterRunnableVariable	<a href="#">VariableDataPrototype</a>	*	aggr	<p>Implement state message semantics for establishing communication among runnables of the same component. The aggregation of explicitInterRunnableVariable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime</p>





Class	SwcInternalBehavior			
handleTerminationAndRestart	HandleTerminationAndRestartEnum	1	attr	This attribute controls the behavior with respect to stopping and restarting. The corresponding AtomicSwComponentType may either not support stop and restart, or support only stop, or support both stop and restart.
implicitInterRunnableVariable	<a href="#">VariableDataPrototype</a>	*	aggr	Implement state message semantics for establishing communication among runnables of the same component. The aggregation of implicitInterRunnableVariable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
includedDataTypeSet	IncludedDataTypeSet	*	aggr	The includedDataTypeSet is used by a software component for its implementation. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=includedDataTypeSet
includedModeDeclarationGroupSet	IncludedModeDeclarationGroupSet	*	aggr	This aggregation represents the included ModeDeclarationGroups <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=includedModeDeclarationGroupSet
instantiationDataDefProps	<a href="#">InstantiationDataDefProps</a>	*	aggr	The purpose of this is that within the context of a given SwComponentType some data def properties of individual instantiations can be modified. The aggregation of InstantiationDataDefProps is subject to variability with the purpose to support the conditional existence of Port Prototypes and component local memories like "per InstanceParameter" or "arTypedPerInstanceMemory". <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=instantiationDataDefProps, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
perInstanceMemory	PerInstanceMemory	*	aggr	Defines a per-instance memory object needed by this software component. The aggregation of PerInstanceMemory is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
perInstanceParameter	<a href="#">ParameterDataPrototype</a>	*	aggr	Defines parameter(s) or characteristic value(s) that needs to be available for each instance of the software-component. This is typically only useful if supportsMultipleInstantiation is set to "true". The aggregation of perInstanceParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects.







Class	SwcInternalBehavior			
				△ <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
portAPIOption	PortAPIOption	*	aggr	Options for generating the signature of port-related calls from a runnable to the RTE and vice versa. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=portAPIOption, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
runnable	<a href="#">RunnableEntity</a>	*	aggr	This is a RunnableEntity specified for the particular Swc InternalBehavior.  The aggregation of RunnableEntity is subject to variability with the purpose to support the conditional existence of RunnableEntities. Note: the number of RunnableEntities might vary due to the conditional existence of Port Prototypes using DataReceivedEvents or due to different scheduling needs of algorithms.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
service Dependency	SwcService Dependency	*	aggr	Defines the requirements on AUTOSAR Services for a particular item.  The aggregation of SwcServiceDependency is subject to variability with the purpose to support the conditional existence of ports as well as the conditional existence of ServiceNeeds.  The SwcServiceDependency owned by an SwcInternal Behavior can be located in a different physical file in order to support that SwcServiceDependency might be provided in later development steps or even by different expert domain (e.g OBD expert for Obd related Service Needs) tools. Therefore the aggregation is <<atp Splitable>>.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
shared Parameter	<a href="#">ParameterData Prototype</a>	*	aggr	Defines parameter(s) or characteristic value(s) shared between SwComponentPrototypes of the same Sw ComponentType The aggregation of sharedParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime





Class	SwcInternalBehavior			
supports Multiple Instantiation	Boolean	1	attr	Indicate whether the corresponding software-component can be multiply instantiated on one ECU. In this case the attribute will result in an appropriate component API on programming language level (with or without instance handle).
variationPoint Proxy	VariationPointProxy	*	aggr	Proxy of a variation points in the C/C++ implementation. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=shortName

**Table A.76: SwcInternalBehavior**

Class	System			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	<p>The top level element of the System Description. The System description defines five major elements: Topology, Software, Communication, Mapping and Mapping Constraints.</p> <p>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.</p> <p><b>Tags:</b> atp.recommendedPackage=Systems</p>			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
clientId DefinitionSet	ClientIdDefinitionSet	*	ref	Set of Client Identifiers that are used for inter-ECU client-server communication in the System.
containerIPdu HeaderByte Order	ByteOrderEnum	0..1	attr	Defines the byteOrder of the header in ContainerIPdus.
ecuExtract Version	RevisionLabelString	0..1	attr	Version number of the Ecu Extract.
fibexElement	FibexElement	*	ref	<p>Reference to ASAM FIBEX elements specifying Communication and Topology.</p> <p>All Fibex Elements used within a System Description shall be referenced from the System Element.</p> <p>atpVariation: In order to describe a product-line, all Fibex Elements can be optional.</p> <p><b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=postBuild</p>
j1939Shared AddressCluster	J1939SharedAddress Cluster	*	aggr	<p>Collection of J1939Clusters that share a common address space for the routing of messages.</p> <p><b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild</p>
mapping	SystemMapping	*	aggr	<p>Aggregation of all mapping aspects (mapping of SW components to ECUs, mapping of data elements to signals, and mapping constraints).</p> <p>In order to support OEM / Tier 1 interaction and shared development for one common System this aggregation is atpSplitable and atpVariation. The content of System</p>





Class	System			
				Mapping 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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> 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	<a href="#">RootSwComposition Prototype</a>	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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> 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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=-10
systemVersion	RevisionLabelString	1	attr	Version number of the System Description.

**Table A.77: System**

Class	TimingEvent			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
<b>Note</b>	TimingEvent references the RunnableEntity that need to be started in response to the TimingEvent			
<b>Base</b>	ARObject, AbstractEvent, AtpClassifier, AtpFeature, AtpStructureElement, <a href="#">Identifiable</a> , <a href="#">Multilanguage Referrable</a> , <a href="#">RTEEvent</a> , <a href="#">Referrable</a>			
Attribute	Type	Mult.	Kind	Note
offset	TimeValue	0..1	attr	The value makes an assumption about the time offset of the first activation of the RunnableEntity triggered by the mapped TimingEvent relative to the periodic activation of the time base of this TimingEvent. Unit: second.
period	TimeValue	1	attr	Period of timing event in seconds. The value of this attribute shall be greater than zero.

**Table A.78: TimingEvent**

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

**Table A.79: Traceable**

<b>Class</b>	<b>Unit</b>			
<b>Package</b>	M2::MSR::AsamHdo::Units			
<b>Note</b>	<p>This is a physical measurement unit. All units that might be defined should stem from SI units. In order to convert one unit into another factor and offset are defined.</p> <p>For the calculation from SI-unit to the defined unit the factor (<i>factorSiToUnit</i>) and the offset (<i>offsetSiToUnit</i>) are applied as follows: <math>x \text{ [unit]} := y * \text{[siUnit]} * \text{factorSiToUnit} \text{ [unit]/[siUnit]} + \text{offsetSiToUnit} \text{ [unit]}</math></p> <p>For the calculation from a unit to SI-unit the reciprocal of the factor (<i>factorSiToUnit</i>) and the negation of the offset (<i>offsetSiToUnit</i>) are applied. <math>y \text{ [siUnit]} := (x * \text{unit} - \text{offsetSiToUnit} \text{ [unit]}) / (\text{factorSiToUnit} \text{ [unit]/[siUnit]})</math></p> <p><b>Tags:</b>atp.recommendedPackage=Units</p>			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
displayName	SingleLanguageUnit Names	0..1	aggr	<p>This specifies how the unit shall be displayed in documents or in user interfaces of tools. The displayName corresponds to the Unit.Display in an ASAM MCD-2MC file.</p> <p><b>Tags:</b>xml.sequenceOffset=20</p>
factorSiToUnit	Float	0..1	attr	<p>This is the factor for the conversion from SI Units to units. The inverse is used for conversion from units to SI Units.</p> <p><b>Tags:</b>xml.sequenceOffset=30</p>
offsetSiToUnit	Float	0..1	attr	<p>This is the offset for the conversion from and to siUnits.</p> <p><b>Tags:</b>xml.sequenceOffset=40</p>
physical Dimension	<a href="#">PhysicalDimension</a>	0..1	ref	<p>This association represents the physical dimension to which the unit belongs to. Note that only values with units of the same physical dimensions might be converted.</p> <p><b>Tags:</b>xml.sequenceOffset=50</p>

**Table A.80: Unit**

<b>Class</b>	<b>UnitGroup</b>			
<b>Package</b>	M2::MSR::AsamHdo::Units			
<b>Note</b>	<p>This meta-class represents the ability to specify a logical grouping of units. The category denotes the unit system that the referenced units are associated to.</p> <p>In this way, e.g. country-specific unit systems (CATEGORY="COUNTRY") can be defined as well as specific unit systems for certain application domains.</p> <p>In the same way a group of equivalent units, can be defined which are used in different countries, by setting CATEGORY="EQUIV_UNITS". KmPerHour and MilesPerHour could such be combined to one group named "vehicle_speed". The unit MeterPerSec would not belong to this group because it is normally not used for vehicle speed. But all of the mentioned units could be combined to one group named "speed".</p> <p>Note that the UnitGroup does not ensure the physical compliance of the units. This is maintained by the physical dimension.</p> <p><b>Tags:</b>atp.recommendedPackage=UnitGroups</p>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
unit	<a href="#">Unit</a>	*	ref	This represents one particular unit in the UnitGroup. <b>Tags:</b> xml.sequenceOffset=20

**Table A.81: UnitGroup**

<b>Class</b>	<b>ValueSpecification</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Constants			
<b>Note</b>	Base class for expressions leading to a value which can be used to initialize a data object.			
<b>Base</b>	<a href="#">ARObject</a>			
<b>Subclasses</b>	<a href="#">AbstractRuleBasedValueSpecification</a> , <a href="#">ApplicationValueSpecification</a> , <a href="#">CompositeValueSpecification</a> , <a href="#">ConstantReference</a> , <a href="#">NotAvailableValueSpecification</a> , <a href="#">NumericalValueSpecification</a> , <a href="#">ReferenceValueSpecification</a> , <a href="#">TextValueSpecification</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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 A.82: ValueSpecification**

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

**Table A.83: VariableDataPrototype**