Document Title	Specification of Timing Extensions
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	411

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History

Date	Release	Changed by	Description
2018-10-31	4.4.0	AUTOSAR Release Management	• Added support for Logical Execution Time • Added element SynchronizationPointConstraint • Removed constraint [constr_4535] from specification. • Added element BswCompositionTiming
2017-12-08	4.3.1	AUTOSAR Release Management	• Editorial changes in chapter 6 and 7.
2016-11-30	4.3.0	AUTOSAR Release Management	• Added support for conditional timing • Added support for timing constraints for Ethernet communications • Added timing function to support mode dependency • Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2015-07-31	4.2.2	AUTOSAR Release Management	• Minor corrections and editorial changes • Added appendices C and D

2014-10-31	4.2.1	AUTOSAR Release Management	• Added the capability in Execution Order Constraint to reference RTE and BSW Events • Added description about how to specify time sets • Minor corrections / clarifications / editorial changes; For details please refer to the BWCStatement
2014-03-31	4.1.3	AUTOSAR Release Management	• Revised the entire contents of chapter ”Application Notes” • Applied editorial changes to section ”Repetitive Execution Order Constraint”
2013-10-31	4.1.2	AUTOSAR Release Management	• Clarified the semantics of jitter and removed ambuiguities in the description of the Periodic Event Triggering Constraint. • Added AUTOSAR constraints in order to ensure specification of cpnsistent Execution Order Constraints. • Added capability to specify logical successor relationships between runnable entities and groups of runnable entities. • Changed the prefix of timing functions from ”ARTE” to ”TIMEX” in order to be consistend with the AUTOSAR standard definitions. • Clarified the use of event types in the various timing views defined in the specification.

2013-03-15	4.1.1	AUTOSAR Release Management	• Applied editorial changes in order to improve readability and comprehensibility of the contents of the document • Added VFB event type TDEventTrigger and extended TDEventSwcInternalBehaviorTypeEnum to indicate variable access of runnable entities • Extended the capability of SynchronizationTimingConstraint to reference timing description events • Revised and extended the capabilities of ExecutionOrderConstraint to specify hierarchical and repetitive execution order constraints • Added the capability to specify blueprints of VfbTimings • Added capabilities to reference timing description events in existing timing models and to support reuse of timing models, as well as AUTOSAR methodology
2011-12-22	4.0.3	AUTOSAR Release Management	• Added new timing constraint types AgeConstraint and ExecutionTimeConstraint • Added occurrence expression language for TimingDescriptionEvents • Improved TDEventModeDeclaration, BurstPatternEventTriggering and SwcTiming
2011-04-15	4.0.2	AUTOSAR Release Management	• Dropped InstanceRefs and replaced with ComponentInCompositionInstanceRef • Restricted the semantics of ExecutionOrderConstraint and OffsetTimingConstraint • Parameterize the observable event 'FlexRayClusterCycleStart' by defining the cycle repetition

2009-12-18	4.0.1	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 11

1.1 Overview . . . 11
1.2 Abbreviations . . . 11
1.3 Glossary of terms . . . 12
1.4 Template implications . . . 14
1.5 Scope . . . 14
1.6 Document conventions . . . 15
1.7 Requirements Traceability . . . 15

2 Timing Extensions Overview 18

3 Timing Views 22

3.1 Timing in Different Phases of the AUTOSAR Methodology . . . 22
3.2 VfbTiming . . . 23
3.3 SwcTiming . . . 25
3.4 SystemTiming . . . 26
3.5 BswModuleTiming and BswCompositionTiming . . . 28
3.5.1 BswModuleTiming . . . 28
3.5.2 BswCompositionTiming . . . 29
3.6 EcuTiming . . . 30

4 Timing Extensions Fundamentals 31

4.1 Formal Specification of Timing Behavior . . . 31
4.2 Timing Extensions and Blueprints . . . 32
4.3 Traceability of Constraints . . . 34
4.4 Specifying Time Sets . . . 34
4.4.1 Example . . . 34
4.5 Conditional Timing . . . 41
4.6 Logical Execution Time . . . 48
4.6.1 Specifying LET Interval . . . 49
4.6.2 Relation between LET Intervals . . . 52
4.6.3 Specifying Executable Entities Cluster . . . 61
4.6.4 Mapping Executable Entities Cluster to LET Interval . . . 62
4.6.5 Ignoring Execution Order . . . 65
4.6.6 Categories specified for LET . . . 65

5 Timing Description Events 66

5.1 Timing Events Related to the VFB . . . 68
5.2 Timing Events Related to SwcInternalBehavior . . . 76
5.3 Timing Events Related to Bus Communication . . . 79
5.4 Timing Events Related to the BSW . . . 86
5.5 Complex Timing Event . . . 90
5.6 Occurrence Expression Language for Timing Events . . . 91
5.6.1 Specifying an Occurrence Expression . . . 91

5.6.2 Occurrence Expression Language Syntax . . . 98
5.6.3 Interpreting an Occurrence Expression . . . 99
5.6.3.1 Interpreting a Content Filter . . . 99
5.6.3.2 Interpreting a Complex Event . . . 100

6 Timing Description Event Chains 102

6.1 Approach . . . 105
6.1.1 Decomposition . . . 105
6.1.2 Composition . . . 106
6.2 Patterns . . . 107
6.2.1 Sequence . . . 107
6.2.2 Fork . . . 108
6.2.3 Join . . . 109
6.2.4 Alternative . . . 109
6.2.5 Cycle . . . 110

7 Timing Constraints 111

7.1 EventTriggeringConstraint . . . 112
7.1.1 PeriodicEventTriggering . . . 113
7.1.1.1 Examples . . . 115
7.1.2 SporadicEventTriggering . . . 118
7.1.3 ConcretePatternEventTriggering . . . 119
7.1.4 BurstPatternEventTriggering . . . 122
7.1.5 ArbitraryEventTriggering . . . 126
7.2 LatencyTimingConstraint . . . 128
7.3 AgeConstraint . . . 131
7.4 SynchronizationTimingConstraint . . . 133
7.4.1 SynchronizationTimingConstraint on Event Chains . . . 135
7.4.2 SynchronizationTimingConstraint on Events . . . 137
7.5 SynchronizationPointConstraint . . . 139
7.6 OffsetTimingConstraint . . . 140
7.7 ExecutionOrderConstraint . . . 141
7.7.1 Ordinary Execution Order Constraint . . . 148
7.7.2 Hierarchical Execution Order Constraint . . . 149
7.7.3 Repetitive Execution Order Constraint . . . 151
7.8 ExecutionTimeConstraint . . . 154

8 Application Notes 155

8.1 Component integration . . . 156
8.1.1 VFB view . . . 160
8.1.2 ECU view . . . 162
8.2 Engine control . . . 164
8.2.1 Overview . . . 164
8.2.2 Timing Requirements . . . 166
8.2.3 Formal description of timing constraints in VFB View . . . 167
8.2.3.1 Requirement 1 . . . 167
8.2.3.2 Requirement 2 . . . 172

8.2.3.3 Requirement 3 . . . 175
8.2.4 Formal description of timing constraints in ECU View . . . 178
8.2.4.1 Requirement 4 . . . 179
8.2.5 Formal description of timing constraints in SW-C View . . . 182
8.2.5.1 Requirement 5 . . . 182
8.3 Describing and Constraining Sensor and Actuator Timing . . . 183
8.3.1 External Event of a Sensor accessed via S/R . . . 184
8.3.2 External Event of an Actuator accessed via S/R . . . 184
8.3.3 External Event of a Sensor accessed via C/S . . . 185
8.3.4 External Event of an Actuator accessed via C/S . . . 185
8.3.5 Considering hardware I/O latency of EventChains at VFB-level 185
8.3.5.1 Input latency . . . 186
8.3.5.2 Output latency . . . 186
8.3.6 Constraining Input or Output Latency . . . 186
A History of Constraints and Specification Items 186
A.1 Constraint History of this Document related to AUTOSAR R4.0.1 . . . 186
A.1.1 Changed Constraints in R4.0.1 . . . 186
A.1.2 Added Constraints in R4.0.1 . . . 186
A.1.3 Deleted Constraints in R4.0.1 . . . 187
A.2 Constraint History of this Document related to AUTOSAR R4.0.2 . . . 187
A.2.1 Changed Constraints in R4.0.2 . . . 187
A.2.2 Added Constraints in R4.0.2 . . . 187
A.2.3 Deleted Constraints in R4.0.2 . . . 187
A.3 Constraint History of this Document related to AUTOSAR R4.0.3 . . . 187
A.3.1 Changed Constraints in R4.0.3 . . . 187
A.3.2 Added Constraints in R4.0.3 . . . 187
A.3.3 Deleted Constraints in R 4.0.3 . . . 188
A.4 Constraint History of this Document related to AUTOSAR R4.1.1 . . . 188
A.4.1 Changed Constraints in R4.1.1 . . . 188
A.4.2 Added Constraints in R4.1.1 . . . 188
A.4.3 Deleted Constraints in R4.1.1 . . . 189
A.5 Constraint History of this Document related to AUTOSAR R4.1.2 . . . 189
A.5.1 Changed Constraints in R4.1.2 . . . 189
A.5.2 Added Constraints in R4.1.2 . . . 189
A.5.3 Deleted Constraints in R4.1.2 . . . 189
A.6 Constraint History of this Document related to AUTOSAR R4.1.3 . . . 190
A.6.1 Changed Constraints in R4.1.3 . . . 190
A.6.2 Added Constraints in R4.1.3 . . . 190
A.6.3 Deleted Constraints in R4.1.3 . . . 190
A.7 Constraint History of this Document related to AUTOSAR R4.2.1 . . . 190
A.7.1 Changed Constraints in R4.2.1 . . . 190
A.7.2 Added Constraints in R4.2.1 . . . 190
A.7.3 Deleted Constraints in R4.2.1 . . . 191
A.8 Constraint History of this Document related to AUTOSAR R4.2.2 . . . 191
A.8.1 Changed Constraints in R4.2.2 . . . 191

A.8.2 Added Constraints in R4.2.2 . . . 191
A.8.3 Deleted Constraints in R4.2.2 . . . 191
A.9 Constraint History of this Document related to AUTOSAR R4.3.0 . . . 191
A.9.1 Changed Constraints in R4.3.0 . . . 191
A.9.2 Added Constraints in R4.3.0 . . . 192
A.9.3 Deleted Constraints in R4.3.0 . . . 192
A.10 Constraint History of this Document related to AUTOSAR R4.3.1 . . . 192
A.10.1 Changed Constraints in R4.3.1 . . . 192
A.10.2 Added Constraints in R4.3.1 . . . 192
A.10.3 Deleted Constraints in R4.3.1 . . . 192
A.11 Constraint History of this Document related to AUTOSAR R4.4.0 . . . 192
A.11.1 Changed Constraints in R4.4.0 . . . 192
A.11.2 Added Constraints in R4.4.0 . . . 192
A.11.3 Deleted Constraints in R4.4.0 . . . 193
A.12 Added Specification Items in R4.0.3 . . . 193
A.13 Added Specification Items in R4.1.1 . . . 194
A.14 Added Specification Items in R4.1.2 . . . 194
A.15 Added Specification Items in R4.1.3 . . . 194
A.16 Added Specification Items in R4.2.1 . . . 194
A.17 Changed Specification Items in R4.2.1 . . . 195
A.18 Added Specification Items in R4.2.2 . . . 195
A.19 Changed Specification Items in R4.2.2 . . . 195
A.20 Added Specification Items in R4.3.0 . . . 195
A.21 Changed Specification Items in R4.3.0 . . . 195
A.22 Added Specification Items in R4.3.1 . . . 195
A.23 Changed Specification Items in R4.3.1 . . . 196
A.24 Added Specification Items in R4.4.0 . . . 196
A.25 Changed Specification Items in R4.4.0 . . . 196

B Mentioned Class Tables 196

C Splitable Elements in the Scope of this Document 229

D Variation Points in the Scope of this Document 230

References

[1] Methodology
AUTOSAR_TR_Methodology

[2] Requirements on Timing Extensions
AUTOSAR_RS_TimingExtensions

[3] Virtual Functional Bus
AUTOSAR_EXP_VFB

[4] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[5] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

[6] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate

1 Introduction

1.1 Overview

This AUTOSAR document contains the specification of the AUTOSAR Timing Extensions. Actually, it has been created as a supplement to the formal definition of the Timing Extensions by means of the AUTOSAR meta-model. In other words, this document in addition to the formal definition provides introductory description and rationale for the part of the AUTOSAR meta-model relevant for the creation of timing models.

1.2 Abbreviations

Abbreviation	*Meaning*
BSW	Basic Software
CAN	Controller Area Network
CC	Communication Controller
COM	Communication module
ECU	Electrical Control Unit
ID	Identifier
IPDU	Interaction Layer Protocol Data Unit
I/O	Input/Output
ISIGNAL	Interaction Layer Signal
LPDU	Data Link Layer Protocol Data Unit
PDU	Protocol Data Unit
RTE	Runtime Environment
SW-C	Software Component

TD	Timing Description
UML	Unified Modeling Language
VFB	Virtual Functional Bus

1.3 Glossary of terms

Term	*Definition*
Jitter	For a periodically occurring timing event, the jitter is defined as the maximum variation of its period with respect to a predefined standard period.
Latency	The latency of a timing event chain describes the time duration between the occurrence of the stimulus and the occurrence of the corresponding response.
Maximum interarrival time	Describes the maximum time interval between two consecutive event occurrences. In the more general case, this attribute is an array of the maximum latency between two, three, four, ... event occurrences.
Minimum interarrival time	Describes the minimum time interval between two consecutive event occurrences. In the more general case, this attribute is an array of the minimum latency between two, three, four, ... event occurrences.
Period	Describes the expected time interval between two consecutive event occurrences, neglecting variation (jitter).
Response	End point of an event chain.
Synchronization	Synchronization focuses on the occurrence of different timing events. Synchronization of timing events means that they must occur simultaneously within a certain tolerance interval.
Stimulus	Start point of an event chain.
Timing analysis	Timing analysis is a method of determining the timing behavior of the system. This includes consideration of timing relevant system behavior like task preemptions, interrupt handling, resource blocking, etc.
Timing constraint	A timing constraint may have two different interpretation alternatives. On the one hand, it may define a restriction for the timing behavior of the system (e.g. minimum (maximum) latency bound for a certain event sequence). In this case, a timing constraint is a requirement which the system must fulfill. On the other hand, a timing constraint may define a guarantee for the timing behavior of the system. In this case, the system developer guarantees that the system has a certain behavior with respect to timing (e.g. a timing event is guaranteed to occur periodically with a certain maximum variation).
Timing description	The timing description of a system, subsystem, software component or BSW consists of events and event chains. The former one describes events that can be observed and the latter one describe their causal relationship.
Timing event	A timing event is the abstract representation of a specific system behavior – that can be observed at runtime – in the AUTOSAR specification. Timing events are used to define the scope for timing constraints. Depending on the specific scope, the view on the system, and the level of abstraction different types of events are defined.

Timing event chain	A timing event chain describes the causal order for a set of functionally dependent timing events. Each event chain has a well defined stimulus and response, which describe its start and end point. Furthermore, it can be hierarchically decomposed into an arbitrary number of sub-chains, so called ”event chain segments”.
Timing event occurrence	A timing event is said to ”occur”, when a specific system behavior – represented by the timing event – can be observed. For example the timing event ”RunnableEntityStarted” occurs, when the associated `RunnableEntity` has entered the state ”started” after its activation.
Timing guarantee	see glossary entry for ”Timing constraint”.
Timing information	Superordinate concept for timing properties and timing constraints.
Timing path	A timing path defines a sequence of communication or computation activities of the system, whose timing behavior shall be examined. Timing paths can be expressed by event chains.
Timing property	A timing property defines the state or value of a timing relevant aspect within the system (e.g. the execution time bounds for a `RunnableEntity` or the priority of a task). Thus, a property does not represent a constraint for the system, but a somehow gathered (e.g. measured, estimated or determined) or defined attribute of the system.
Timing requirement	A timing requirement defines a restriction on timing that must be fulfilled to ensure proper operation of the system. Timing requirements can be expressed by using timing constraints.
Timing validation	Timing validation compares the result of timing analysis (see glossary entry for timing analysis) with the expected behavior defined by timing constraints (see glossary entry for timing constraints).

1.4 Template implications

All AUTOSAR templates use a common meta-model which is defined by using the Unified Modeling Language (UML). For the integration of timing information into the AUTOSAR meta-model we have to decide between two viable alternatives: on the one hand the extension of existing templates, and on the other hand the definition of a separate timing template.

Several discussions lead to the decision to explicitly NOT defining a separate timing template. The most valuable advantage of such an approach is addressed by the idea behind the current template composition. They are highly adapted to the AUTOSAR methodology (see [1] for more details about the AUTOSAR methodology) and the several templates handle specific process steps in the methodology. Since it is not our scope to provide a proposal for a timing augmented development process, it is as well not in our scope to define an isolated, new process step (e.g. a timing process step). For this reason, our project result has an impact to some of the existing templates. Therefore, the augmentation of the existing templates instead of the creation of a new timing template reduces dependencies in the meta-model among templates.

1.5 Scope

The primary purpose of the timing extensions is to support constructing embedded real-time systems that satisfy given timing requirements and to perform timing analysis/validations of those systems once they have build up.

The AUTOSAR Timing Extensions provide a timing model as specification basis for a contract based development process, in which the development is carried out by different organizations in different locations and time frames. The constraints entered in the early phase of the project (when corresponding solutions are not developed yet) shall be seen as extra-functional requirements agreed between the development partners. In such way the timing specification supports a top-down design methodology. However, due to the fact that a pure top-down design is not feasible in most of the cases (e.g. because of legacy code), the timing specification allows the bottom-up design methodology as well.

The resulting overall specification (AUTOSAR Model *and* Timing Extensions) shall enable the analysis of a system's timing behavior and the validation of the analysis results against timing constraints. Thus, timing properties required for the analysis must be contained in the timing augmented system model (such as the priority of a task, the activation behavior of an interrupt, the sender timing of a PDU and frame etc.). Such timing properties can be found all across AUTOSAR. For example the System Template provides means to configure and specify the timing behavior of the communication stack. Furthermore the execution time of executable entities can be specified. In addition, the overall specification must provide means to describe timing constraints. A timing constraint defines a restriction for the timing behavior of the system (e.g. bounding the maximum latency from sensor sampling to actuator access). Timing constraints

are added to the system model using the AUTOSAR Timing Extensions. Constraints, together with the result of timing analysis, are considered during the validation of a system's timing behavior, when a nominal/actual value comparison is performed.

Note: The timing specification shall enable the analysis and validation of an AUTOSAR system's timing behavior. However, the specification of analysis and validation **results** (e.g. the maximum resource load of an ECU, etc.) is not addressed in this document.

1.6 Document conventions

Technical terms (Meta Class Names) are typeset in monospaced font, e.g. `FrameTriggering`.

1.7 Requirements Traceability

The following table references the requirements specified in AUTOSAR RS Timing Extensions [2] and denotes how each of them are satisfied by the meta-model.

Requirement	Description	Satisfied by
[RS_TIMEX_00001]	Timing properties	[TPS_TIMEX_00001] [TPS_TIMEX_00002] [TPS_TIMEX_00003] [TPS_TIMEX_00004] [TPS_TIMEX_00005] [TPS_TIMEX_00006] [TPS_TIMEX_00007] [TPS_TIMEX_00008] [TPS_TIMEX_00010] [TPS_TIMEX_00011] [TPS_TIMEX_00012] [TPS_TIMEX_00013] [TPS_TIMEX_00014] [TPS_TIMEX_00015] [TPS_TIMEX_00016] [TPS_TIMEX_00017] [TPS_TIMEX_00018] [TPS_TIMEX_00019] [TPS_TIMEX_00020] [TPS_TIMEX_00021] [TPS_TIMEX_00022] [TPS_TIMEX_00023] [TPS_TIMEX_00024] [TPS_TIMEX_00025]

		[TPS_TIMEX_00026] [TPS_TIMEX_00027] [TPS_TIMEX_00028] [TPS_TIMEX_00029] [TPS_TIMEX_00030] [TPS_TIMEX_00031] [TPS_TIMEX_00032] [TPS_TIMEX_00033] [TPS_TIMEX_00034] [TPS_TIMEX_00035] [TPS_TIMEX_00036] [TPS_TIMEX_00038] [TPS_TIMEX_00039] [TPS_TIMEX_00041] [TPS_TIMEX_00042] [TPS_TIMEX_00043] [TPS_TIMEX_00044] [TPS_TIMEX_00045] [TPS_TIMEX_00046] [TPS_TIMEX_00047] [TPS_TIMEX_00048] [TPS_TIMEX_00052] [TPS_TIMEX_00053]
[RS_TIMEX_00002]	Timing constraints	[TPS_TIMEX_00003] [TPS_TIMEX_00004] [TPS_TIMEX_00006] [TPS_TIMEX_00007] [TPS_TIMEX_00010] [TPS_TIMEX_00011] [TPS_TIMEX_00012] [TPS_TIMEX_00013] [TPS_TIMEX_00014] [TPS_TIMEX_00015] [TPS_TIMEX_00038] [TPS_TIMEX_00041] [TPS_TIMEX_00046] [TPS_TIMEX_00047] [TPS_TIMEX_00048]
[RS_TIMEX_00003]	Optionality of timing constraints	[TPS_TIMEX_00009]
[RS_TIMEX_00004]	Event chains	[TPS_TIMEX_00002]
[RS_TIMEX_00005]	Structure of event chains	[TPS_TIMEX_00002]
[RS_TIMEX_00006]	Triggering behavior of event chains	[TPS_TIMEX_00003] [TPS_TIMEX_00010] [TPS_TIMEX_00011] [TPS_TIMEX_00012] [TPS_TIMEX_00013] [TPS_TIMEX_00014]
[RS_TIMEX_00007]	Synchronization of event chains	[TPS_TIMEX_00006]

[RS_TIMEX_00008]	Multiple asynchronous time bases	[TPS_TIMEX_00003] [TPS_TIMEX_00006] [TPS_TIMEX_00010] [TPS_TIMEX_00011] [TPS_TIMEX_00012] [TPS_TIMEX_00013] [TPS_TIMEX_00014] [TPS_TIMEX_00015]
[RS_TIMEX_00009]	Loop-back signal flow in sender-receiver communication	[TPS_TIMEX_00002] [TPS_TIMEX_00005]
[RS_TIMEX_00010]	Validity of timing properties and constraints	[TPS_TIMEX_00037]
[RS_TIMEX_00011]	Mode dependency	[TPS_TIMEX_00049] [TPS_TIMEX_00050] [TPS_TIMEX_00051]
[RS_TIMEX_00012]	Sensor/actuator delay	[TPS_TIMEX_00004]
[RS_TIMEX_00013]	Specification of timing resources for software-component description	[TPS_TIMEX_00008]
[RS_TIMEX_00014]	Sequence of execution of runnable entities	[TPS_TIMEX_00007] [TPS_TIMEX_00038] [TPS_TIMEX_00041] [TPS_TIMEX_00046] [TPS_TIMEX_00047] [TPS_TIMEX_00048]
[RS_TIMEX_00015]	Timing-requirements of SW-Components	[TPS_TIMEX_00004] [TPS_TIMEX_00010]
[RS_TIMEX_00016]	Some elements of the Timing Extensions shall be blueprintable	[TPS_TIMEX_00040]
[RS_TIMEX_00017]	Synchronization constraint on events	[TPS_TIMEX_00006]
[RS_TIMEX_00018]	Predefined events for port interfaces at VFB level	[TPS_TIMEX_00039]
[RS_TIMEX_00019]	AUTOSAR Methodology support	[TPS_TIMEX_00020] [TPS_TIMEX_00042] [TPS_TIMEX_00043] [TPS_TIMEX_00044] [TPS_TIMEX_00045]
[RS_TIMEX_00020]	Support for events indicating variable accesses	[TPS_TIMEX_00020] [TPS_TIMEX_00044]
[RS_TIMEX_00022]	Support for Logical Execution Time	[TPS_TIMEX_00055] [TPS_TIMEX_00056] [TPS_TIMEX_00057]
[RS_TIMEX_00023]	Support for Specifying Synchronization	[TPS_TIMEX_00054]

Table 1.1: RequirementsTracing

2 Timing Extensions Overview

The AUTOSAR Timing Extensions provide some basic means to describe and specify timing information: Timing descriptions, expressed by *events* and *event chains*, and *timing constraints* that are imposed on these events and event chains. Both means, timing descriptions and timing constraints, are organized in *timing views* for specific purposes. By and large, the purpose of the Timing Extensions are two folded: The first purpose is to provide timing requirements that guide the construction of systems which eventually shall satisfy those timing requirements. And the second purpose is to provide sufficient timing information to analyze and validate the temporal behavior of a system.

Events. Events refer to locations in systems at which the *occurrences* of events are observed. The AUTOSAR Specification of Timing Extensions defines a set of predefined event types for such *observable locations*. Those event types are used in different *timing views* and each of these timing views correspond to one of the AUTOSAR views: *VFB Timing* and Virtual Function Bus VFB View; *SW-C Timing* and Software Component View; *System Timing* and System View; *BSW Module Timing* and Basic Software Module View; as well as *ECU Timing* and ECU View.

In particular, one uses these events to specify the reading and writing of data from and to specific ports of software components, calling of services and receiving their responses (VFB, SW-C, System and ECU Timing); sending and receiving data via networks and through communication stacks (System and ECU Timing); activating, starting and terminating executable entities (SW-C Timing and Basic SW Module Timing); and last but not least calling basic software services and receiving their responses (ECU Timing and Basic SW Module Timing).

Event Chains. Event chains specify a causal relationship between events and their temporal occurrences. The notion of event chain enables one to specify the relationship between two events, for example when an event A occurs then the event B occurs, or in other words, the event B occurs if and only if the event A occurred before. In the context of an event chain the event A plays the role of the *stimulus* and the event B plays the role of the *response*. Event chains can be composed of existing event chains and decomposed into further event chains — in both cases the event chains play the role of *event chain segments*.

Timing Constraints imposed on Events. The notion of Event is used to describe that in a system specific events occur and also at which locations in this system the occurrences are observed. In addition, an Event Triggering Constraint imposes a constraint on the occurrences of an event, which means that the event triggering constraint specifies the way an event occurs in the temporal space. The AUTOSAR Specification of Timing Extensions provides means to specify periodic and sporadic event occurrences, as well as event occurrences that follow a specific pattern (burst, concrete, and arbitrary pattern).

Timing Constraints imposed on Event Chains. Like event triggering constraints impose timing constraints on events and their occurrences; the latency and synchro-

nization timing constraints impose constraints on event chains. In the former case, a constraint is used to specify a reaction and age, for example if a stimulus event occurs then the corresponding response event shall occur not later than a given amount of time. And in the latter case, the constraint is used to specify that stimuli or response events must occur within a given time interval (tolerance) to be said to occur simultaneous and synchronous respectively.

Additional Timing Constraints. In addition to the timing constraints that are imposed on events and event chains, the AUTOSAR Timing Extensions provide timing constraints which are imposed on *Executable Entities*, namely the *Execution Order Constraint* and *Execution Time Constraint*.

The concepts sketched so far are represented by the metamodel shown in Figure 2.1. And every part is described in the subsequent chapters and sections.

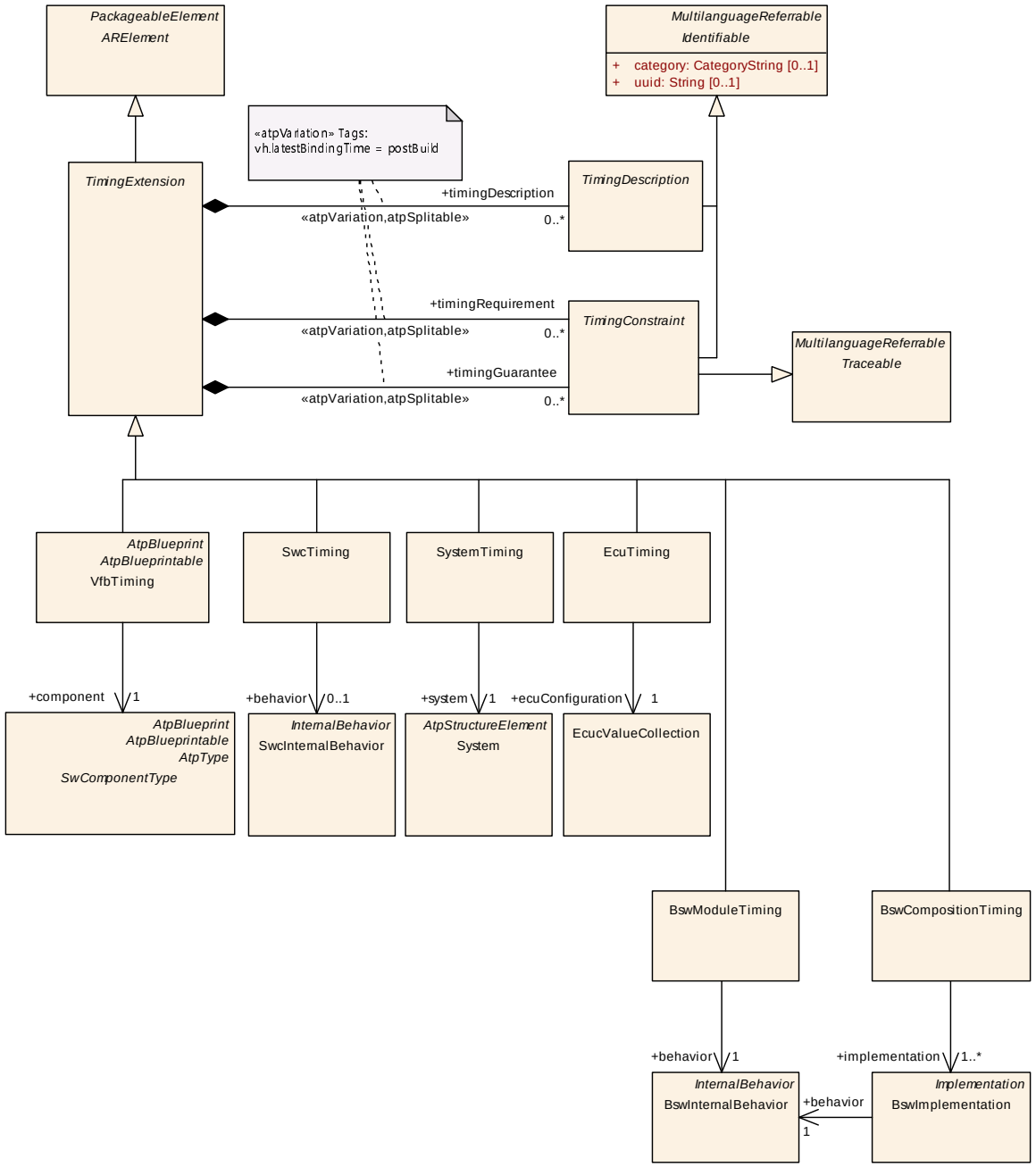


Figure 2.1: Overview of the Timing Extensions including Timing Descriptions, Timing Constraints, and Timing Views.

Class	***TimingExtension*** (abstract)
Package	M2::AUTOSARTemplates::CommonStructure::Timing
Note	The abstract parent class of the different template specific timing extensions. Depending on the specific timing extension (VfbTiming, SwcTiming, SystemTiming, BswModuleTiming, EcuTiming) the timing descriptions and timing constraints, that can be used to specify the timing behavior, are restricted.

△

Class	***TimingExtension*** (abstract)			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	BswCompositionTiming, BswModuleTiming, EcuTiming, SwcTiming, SystemTiming, VfbTiming			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
timingCondition	TimingCondition	*	aggr	The timing condition specifies a specific condition. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
timing Description	TimingDescription	*	aggr	The timing descriptions that belong to a specific timing specification. In order to support different timing description variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
timing Guarantee	TimingConstraint	*	aggr	The timing constraints that belong to a specific timing specification in the role of a timing guarantee. In order to support different timing constraint variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
timing Requirement	TimingConstraint	*	aggr	The timing constraints that belong to a specific timing specification in the role of a timing requirement. In order to support different timing constraint variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
timingResource	TimingExtension Resource	0..1	aggr	The timing resource contains all instance references referred from within a timing condition formula of a timing view. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel

Table 2.1: TimingExtension

Class	***TimingDescription*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription			
Note	The abstract parent class of the model elements that are used to define the scope of a timing constraint. **Tags:** xml.sequenceOffset=10			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*TimingDescriptionEvent*, TimingDescriptionEventChain			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 2.2: TimingDescription

Class	***TimingConstraint*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint			
Note	The abstract parent class of different timing constraints supported by the Timing extension. A concrete timing constraint is used to bound the timing behavior of the model elements in its scope. **Tags:** xml.sequenceOffset=20			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Traceable*			
Subclasses	AgeConstraint, *EventTriggeringConstraint*, ExecutionOrderConstraint, ExecutionTimeConstraint, Latency TimingConstraint, OffsetTimingConstraint, SynchronizationPointConstraint, SynchronizationTiming Constraint			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
timingCondition	TimingCondition	0..1	ref	A timing condition the timing constraint depends on. In other words it specifies the condition the timing constraint holds.

Table 2.3: TimingConstraint

3 Timing Views

The AUTOSAR Timing Extensions define five distinct timing views. Each of these views is associated with one of the AUTOSAR views, namely Virtual Function Bus-, Software Component-, System-, Basic Software Module- and ECU view. Figure 2.1 provides an overview of the AUTOSAR Timing Extensions and its basic elements including the timing views.

This chapter outlines the timing views that are used in the different phases of the AUTOSAR methodology in order to create appropriate timing models.

3.1 Timing in Different Phases of the AUTOSAR Methodology

The AUTOSAR methodology (see [1] for a general introduction) provides several well-defined process steps, and furthermore artifacts that are provided or needed by these steps. Figure 3.1 provides a simplified view of the AUTOSAR methodology, focusing on the process phases which are of interest for the use of the timing extensions. These represented steps and artifacts are grouped by boundaries in five views:

VfbTiming This view deals with timing information related to the interaction of `SwComponentType`s at VFB level.

SwcTiming This view deals with timing information related to the `SwcInternalBehavior` of `AtomicSwComponentType`s.

SystemTiming This view deals with timing information related to a `System`, utilizing information about topology, software deployment, and signal mapping.

BswModuleTiming and BswCompositionTiming The first perspective deals with timing information related to the `BswInternalBehavior` of a single `BswModuleDescription`. Whereas the latter perspective deals with timing informa-

tion related to BswInternalBehavior of more than one implementation of a BswModuleDescription.

EcuTiming This view deals with timing information related to the EcucValueCollection covering the entire software on an ECU with application software and configured basic software.

For each of these views a special focus of timing specification can be applied, depending on the availability of necessary information, the role a certain artifact is playing and the development phase, which is associated with the view.

The following sections give a detailed overview of every timing view and their relevance for timing specification. For each view it is explained what kind of timing description and timing constraints can be applied and to which AUTOSAR specification documents these can be attached to.

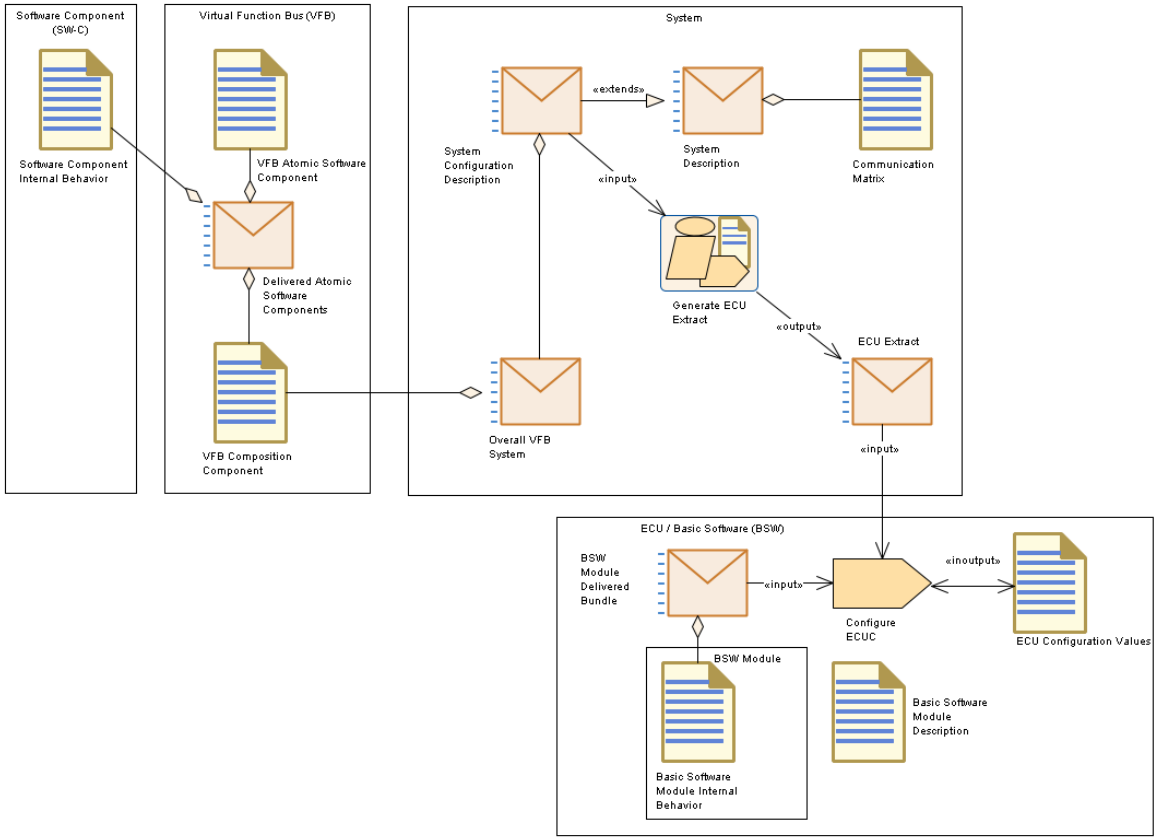


Figure 3.1: Overview of the AUTOSAR methodology and timing specification

3.2 VfbTiming

AUTOSAR defines the *Virtual Functional Bus* [3] as a composition of SwComponentPrototypes at a logical level, regardless of their physical distribution. On this logical level a special view can be applied for timing specification. This section describes what kind of timing specification can be applied at VFB level for a system or sub-system. Typ-

ically, end-to-end timing constraints, including (physical) sensors and actuators, shall be captured in this view, allowing an early formalization of those constraints.

Neglecting the physical distribution means that the VfbTiming view does not deal with the question, in which system context the prototype of a CompositionSwComponentType shall be implemented. An additional restriction of the VfbTiming view raises due to the black box treatment of software components. The SwcInternalBehavior of AtomicSwComponentTypes is not considered. For these mentioned restrictions (irrelevance of the physical distribution, black box view), TimingDescriptions at VFB level should only refer to SwComponentTypes, PortPrototypes and their connections, but not the InternalBehavior.

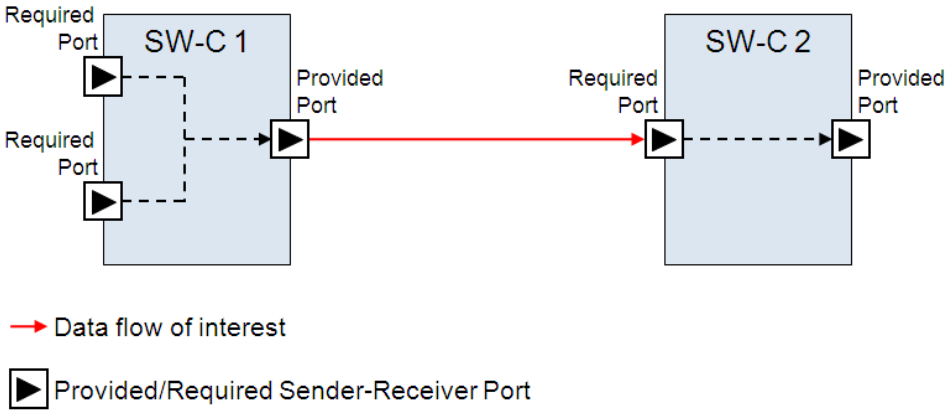


Figure 3.2: Example: Data flow in the scope of the VfbTiming view

The VfbTiming view is applicable for different system granularities. The smallest granularity is the investigation of a single SwComponentType without any contextual embedding. Here, a timing description can only refer to relations between a component's RPortPrototypes and the same component's PPortPrototypes.

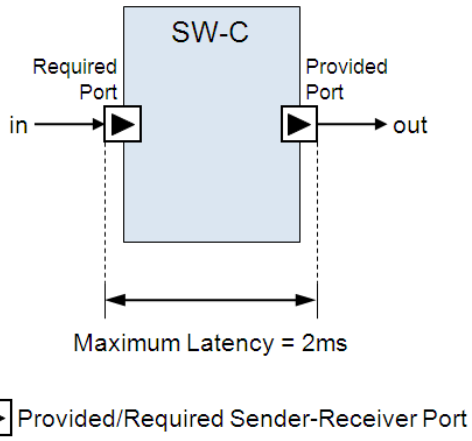


Figure 3.3: Example: Latency requirement

As an example, consider the timing constraint illustrated in Figure 3.3: ”From the point in time, where the value *in* is received by the Software Component named *SW-C*, until the point in time, where the newly calculated value *out* is sent, there shall be a maximum latency of 2 ms”. This would be attached to the timing description that refers to an `AtomicSwComponentType` *SW-C* (see Figure 3.1).

In case of a `CompositionSwComponentType` that itself contains other `SwComponent Prototype`s, the timing interrelation between different components, e.g. from one component's `PPortPrototype` to another component's `RPortPrototype`, could be of interest.

[TPS_TIMEX_00032] Purpose of `VfbTiming` ⌈ The element `VfbTiming` aggregates all timing information, timing descriptions and timing constraints, that is related to the VFB View. ⌋*(RS_TIMEX_00001)*

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

Table 3.1: VfbTiming

3.3 SwcTiming

In contrast to the `VfbTiming` view, a specification engineer might especially be interested in the `SwcInternalBehavior` of `AtomicSwComponentType`s that are represented as black boxes at VFB level. The `SwcInternalBehavior` specifies a component's behavioral decomposition into runnable entities, which are executed at runtime. Thus, in `SwcTiming` view, a timing description is attached to the `SwcInternalBehavior` of a `SwComponentType` (see Figure 3.1). It can refer to the activation, start, and termination (see section 5.2) of the execution of a `RunnableEntity`.

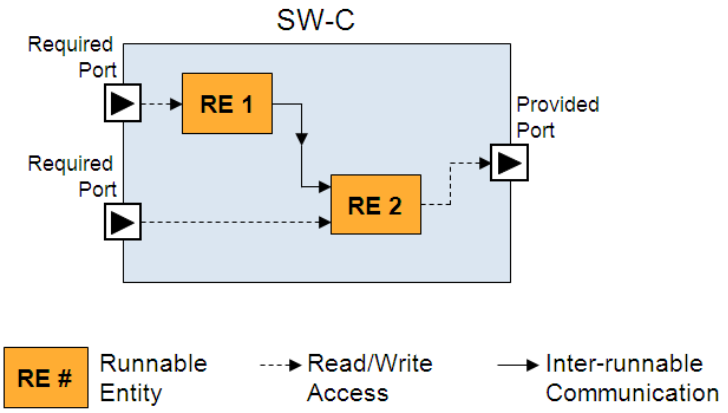


Figure 3.4: Example: Data flow in the scope of the SW-C Timing view

[TPS_TIMEX_00033] Purpose of `SwcTiming` ⌈ The element `SwcTiming` aggregates all timing information, timing descriptions and timing constraints, that is related to the Software Component View. ⌋*(RS_TIMEX_00001)*

Class	**SwcTiming**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing			
Note	The SwcTiming is used to describe the timing of an atomic software component. TimingDescriptions aggregated by SwcTiming are restricted to event chains referring to events which are derived from the classes TDEventVfb and TDEventSwcInternalBehavior. **Tags:** atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *TimingExtension*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
behavior	SwcInternalBehavior	0..1	ref	This defines the scope of a SwcTiming. All corresponding timing descriptions and constraints must be defined within this scope. Note! The reason for the cardinality of 0..1 is to ensure backward compatibility.

Table 3.2: SwcTiming

3.4 SystemTiming

At system level a special prototype of a `CompositionSwComponentType`—the `RootSwCompositionPrototype`—is instantiated. This prototype, the chosen hardware topology and other artifacts are used as input to the task dealing with the deployment of software components onto ECUs in order to configure the system. The main configuration result is the mapping of software components to ECUs and in further steps the resulting communication matrix is created. This information is aggregated in the `System` description (see Figure 3.1).

The `SystemTiming` view is used to provide timing informations at system level. As an extension, it can be attached to a `System`. As the `System` description aggregates all the information about `SwComponentType`s and their corresponding `SwcInter-`

nalBehavior, it is possible to use the same concepts that are available in the views VfbTiming and SwcTiming also in this timing view. The difference is the specific system context that defines the validity of timing informations at system level. Without knowledge of the mapping of software components to a target hardware respectively ECU, only a generic platform independent description can be provided.

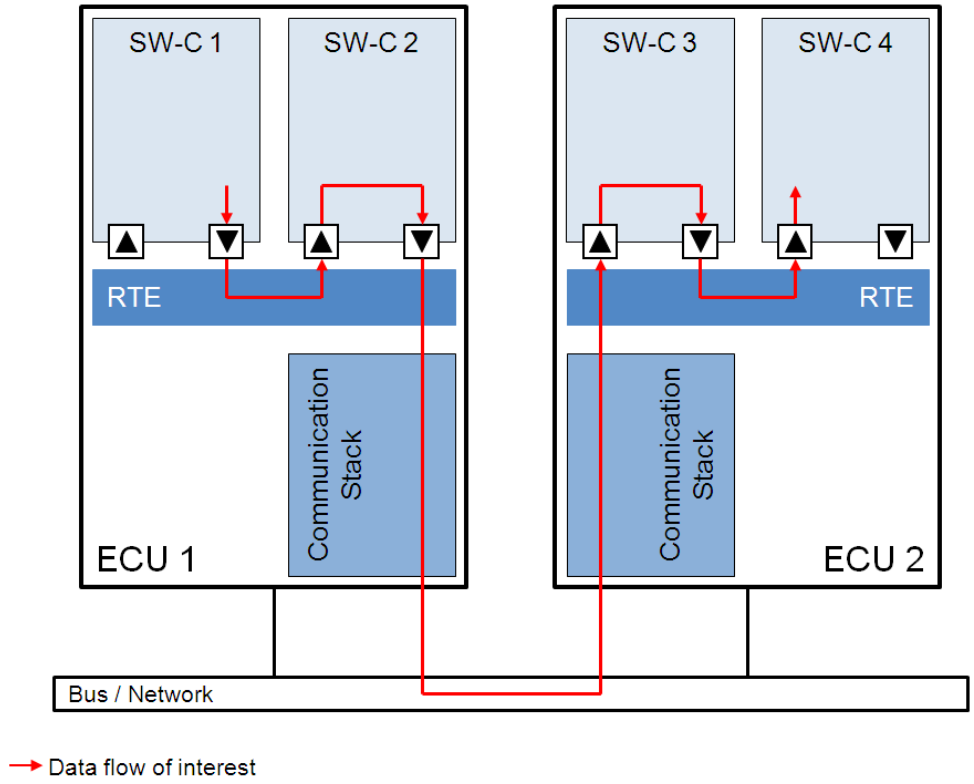


Figure 3.5: Example: Data flow in the scope of System Timing view

In addition, a timing description in system view refers to the concrete communication of software components that only was represented as abstract connectors in VfbTiming view. Due to the software mapping, now communication is either local communication over the RTE (both software components on same ECU) or remote communication over the RTE, through the communication stack of the BSW and a communication bus. A system-specific timing description thus can refer to signals (RTE), I-PDUs (COM) and frames (communication driver and bus).

[TPS_TIMEX_00034] Purpose of SystemTiming ⌈ The element SystemTiming aggregates all timing information, timing descriptions and timing constraints, that is related to the System View. ⌋*(RS_TIMEX_00001)*

Class	**SystemTiming**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing			
Note	A model element used to refine timing descriptions and constraints (from a VfbTiming) at System level, utilizing information about topology, software deployment, and signal mapping described in the System Template. TimingDescriptions aggregated by SystemTiming are restricted to events which are derived from the class TDEventVfb, TDEventSwcInternalBehavior and TDEventCom. **Tags:** atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *TimingExtension*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
system	System	1	ref	This defines the scope of a SystemTiming. All corresponding timing descriptions and constraints must be defined within this scope.

Table 3.3: SystemTiming

3.5 BswModuleTiming and BswCompositionTiming

The Basic Software Timing consists of two perspectives, which are described in the following subsections.

3.5.1 BswModuleTiming

According to Figure 3.1, a `BswModuleDescription` is generated for each BSW module as part of the ECU configuration phase. For every module its internals, the `BswInternalBehavior`, must be defined, i.e. structuring any `BswModuleEntity`. Similar to the timing view on the `SwcInternalBehavior` of an `AtomicSwComponentType` as described in section 3.3, the BSW module timing view focuses on the activation, start and end of the execution of any `BswModuleEntity`. The timing description for each module can be attached to the `BswModuleDescription`.

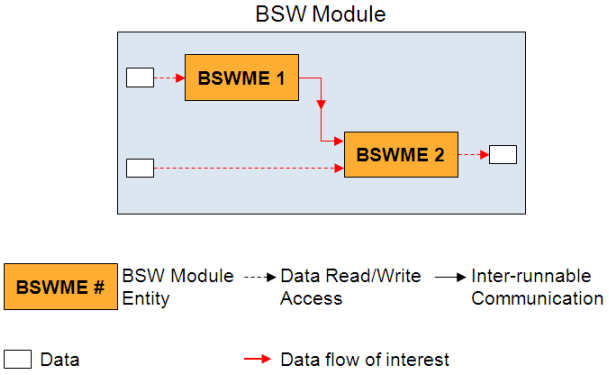


Figure 3.6: Example: Data flow in scope of BSW Module Timing view

[TPS_TIMEX_00035] Purpose of `BswModuleTiming` ⌈ The element `BswModuleTiming` aggregates all timing information, timing descriptions and timing constraints, that is related to the Basic Software Module View. ⌋*(RS_TIMEX_00001)*

Class	**BswModuleTiming**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing			
Note	A model element used to define timing descriptions and constraints for the BswInternalBehavior of one BSW Module. Thereby, for each BswInternalBehavior a separate timing can be specified. A constraint defined at this level holds true for all Implementations of that BswInternalBehavior. TimingDescriptions aggregated by BswModuleTiming are restricted to event chains referring to events which are derived from the class TDEventBswInternalBehavior. **Tags:** atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *TimingExtension*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
behavior	BswInternalBehavior	1	ref	This defines the scope of a BswModuleTiming. All corresponding timing descriptions and constraints must be defined within this scope.

Table 3.4: BswModuleTiming

3.5.2 BswCompositionTiming

In contrast to the element `BswModuleTiming`, which describes the timing information of a single `BswInternalBehavior`, the `BswCompositionTiming` describes the timing information for more than one implementation of a basic software module — a *composition* of basic software modules. Typically, this is the case when the timing information of a basic software stack, consisting of several basic software modules, shall be described or the timing information of a basic software module that is deployed onto different cores shall be described.

[TPS_TIMEX_00053] Purpose of `BswCompositionTiming` ⌈ The element `BswCompositionTiming` describes the timing information related to `BswInternalBehavior` of more than one implementation of a `BswModuleDescription`. ⌋ *(RS_TIMEX_00001)*

Class	**BswCompositionTiming**
Package	M2::AUTOSARTemplates::CommonStructure::Timing
Note	A model element used to define timing descriptions and constraints for a set of BswImplementations representing a BSW composition. A constraint defined at this level holds true for all referenced Bsw Implementations. Note, that multiple implementations of the same basic software module could be involved. TimingDescriptions aggregated by BswCompositionTiming are restricted to event chains referring to events which are derived from the class TDEventBswInternalBehavior and TDEventBsw. **Tags:** atp.recommendedPackage=TimingExtensions
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *TimingExtension*

▽

△

Class	**BswCompositionTiming**			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
implementation	BswImplementation	1..*	ref	This defines the scope of a BswCompositionTiming. All corresponding timing descriptions and constraints must be defined within this scope.

Table 3.5: BswCompositionTiming

3.6 EcuTiming

A result of the ECU configuration phase is the complete `EcucValueCollection` representing the ECU's configuration description (see Figure 3.1). During ECU configuration, this artifact is filled amongst others with ...

- ... the ECU Extract of System Configuration, where the needed part of the overall system description for the respective ECU is extracted.
- ... references to information about all BSW modules present on the ECU. Such BSW modules are described by `BswModuleDescription`s, providing for instance information about the interfaces that the modules offer or require.

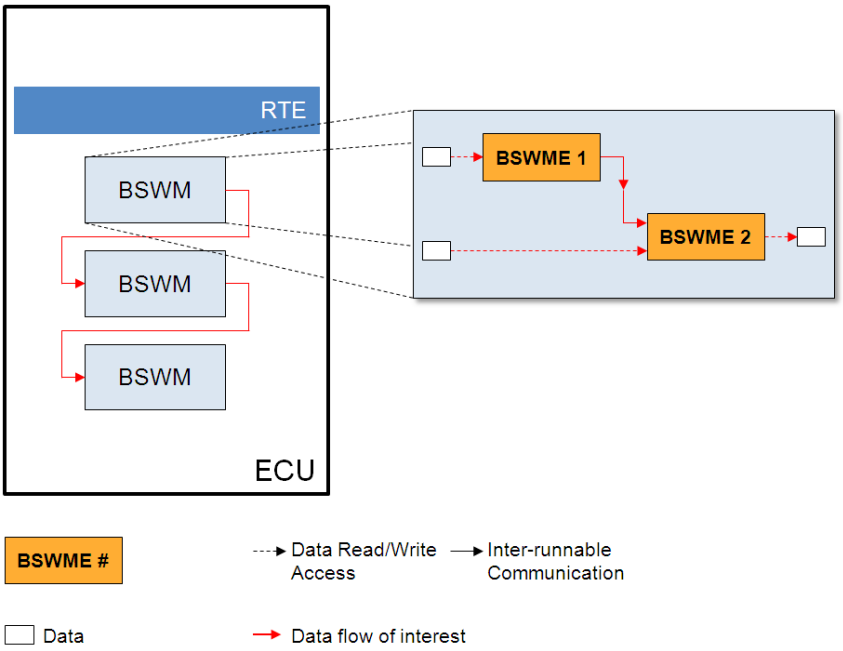


Figure 3.7: An example of data flow, whose timing behavior is in scope of ECU view

In this view, timing can reference all the ECU-relevant information: The deployed software component instances, the ECU related interactions including bus communication, Basic Software, etc. In other words, the `EcuTiming` has the same expressivity as the System Timing view but only focusing on one specific ECU in the system's topology. In addition, the entire BSW can be considered during timing modeling, because the

complete composition and internal structure of the BSW modules are known. The internals of BSW modules and the inter-relations between BS modules are of interest in this timing view. The information is attached to the EcucValueCollection.

[TPS_TIMEX_00036] Purpose of EcuTiming ⌈ The element EcuTiming aggregates all timing information, timing descriptions and timing constraints, that is related to the ECU View. ⌋*(RS_TIMEX_00001)*

Class	**EcuTiming**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing			
Note	A model element used to define timing descriptions and constraints within the scope of one ECU configuration. TimingDescriptions aggregated by EcuTiming are allowed to use all events derived from the class TimingDescriptionEvent. **Tags:** atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *TimingExtension*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
ecu Configuration	EcucValueCollection	1	ref	This defines the scope of an EcuTiming. All corresponding timing descriptions and constraints must be defined within this scope.

Table 3.6: EcuTiming

4 Timing Extensions Fundamentals

This section explains the fundamentals that the timing extensions, described in the following sections, are based upon.

4.1 Formal Specification of Timing Behavior

Compared to the specification of a system's functional behavior, the specification of its timing behavior requires additional information to be captured. Not only the eventual occurrence of events but also their exact timing or the concurrency of various events become important. Therefore, in the specification of timing extensions for AUTOSAR, the *event* is the basic entity. This event is used to refer to an observable behavior within a system (e.g. the activation of a RunnableEntity, the transmission of a frame etc.) at a certain point in time.

Having to deal with different abstraction levels and views (see chapter 3), and in order to avoid semantic confusion with existing concepts, a new abstract type TimingDescriptionEvent (see section 5) is introduced as a formal basis for the timing extensions. Depending on the model entity and the associated observable behavior, specific timing events are defined and linked to the different views.

For the analysis of a system's timing behavior usually not only single events but also the correlation of different events is of fundamental importance. To relate timing events to each other, a further concept called TimingDescriptionEventChain (see section 6) is introduced. Hereby, it is important to note that for the events referred to within an event chain a functional dependency is implicitly assumed. This means that an event of a chain somehow causes subsequent chain events. An example for an end-to-end event chain with bus communication is depicted in Figure 3.5 in chapter 3. This event chain describes the path from software component instance "SWC1" to software component instance "SWC3".

Based on events and event chains, it is possible to express various specific timing constraints derived from the abstract type TimingConstraint. These timing constraints specify the expected timing behavior. As timing constraints shall be valid independently from implementation details, they are also expressed on a abstract level by referencing the above introduced formal basis of TimingDescriptionEvents and TimingDescriptionEventChains.

Thus, by means of events, event chains and timing constraints defined on top of these, a separate central timing specification can be provided, decoupling the expected timing behavior from the actually implemented behavior. This approach supports timing contracts for AUTOSAR systems in a top-down as well as bottom-up approach.

[TPS_TIMEX_00009] Optional use of timing extensions ⌈ The elements TimingExtension, TimingDescription, and TimingConstraint of the timing extensions are derived from the element ARElement. This enables one to deliver timing extensions in a separate document. In addition, there are no external references from any template that point to timing extensions elements. ⌋*(RS_TIMEX_00003)*

4.2 Timing Extensions and Blueprints

[TPS_TIMEX_00040] Blueprinting VfbTiming ⌈ VfbTiming can be blueprinted. ⌋ *(RS_TIMEX_00016)*

The primary purpose of blueprinting VfbTiming is to annotate Application Interfaces and attach timing constraints, like age- and periodic event triggering constraints, to events of type TDEventVfb which reference port prototype blueprints. The concept of Blueprints and its details are described in [4].

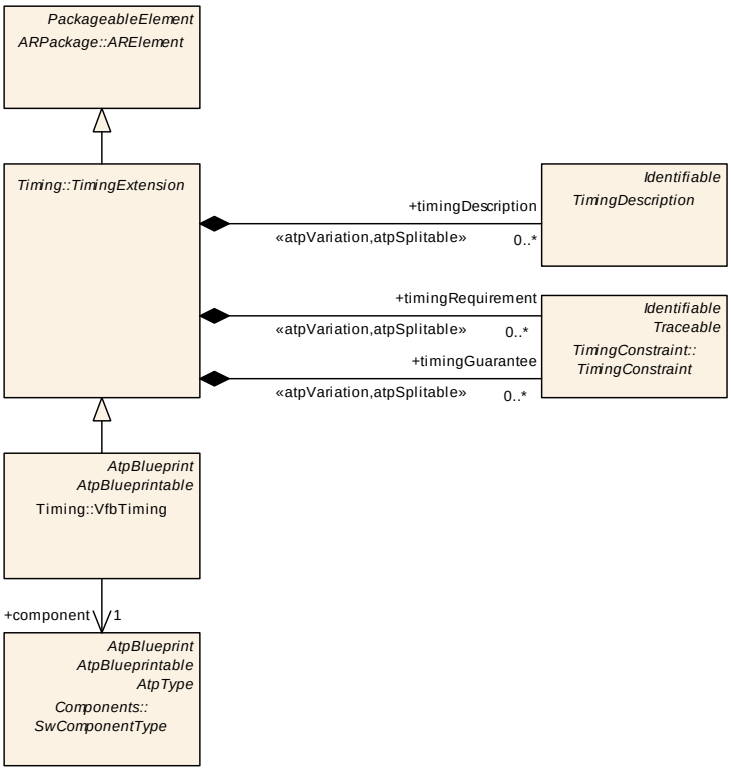


Figure 4.1: VFB Timing Blueprint

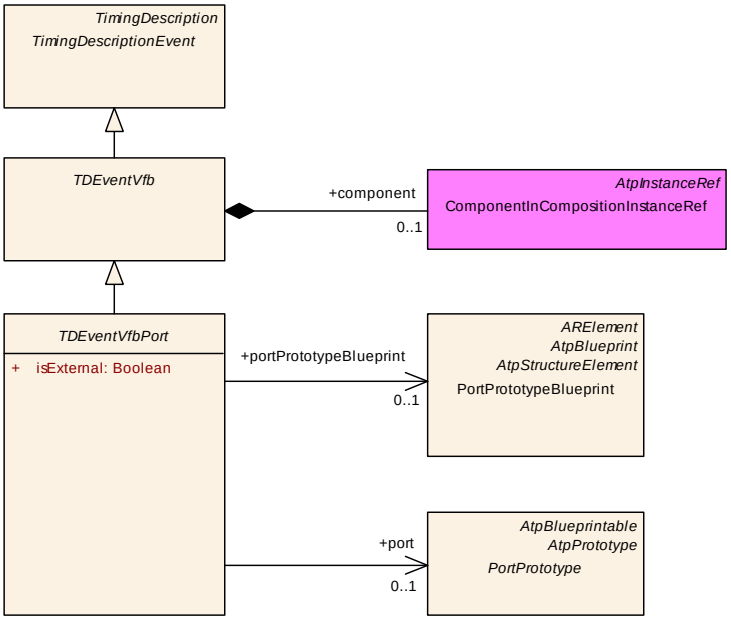


Figure 4.2: `TDEventVfb` Blueprint

[constr_4508] `TDEventVfb` shall reference `PortPrototypeBlueprint` only in Blueprints ⌈ An event type `TDEventVfb` only shall reference `PortPrototype-Blueprint` in blueprints. ⌋*()*

[constr_4509] Only `VfbTiming` shall be a Blueprint ⌈ Only the `VfbTiming` is blueprintable. ⌋*()*

4.3 Traceability of Constraints

[TPS_TIMEX_00037] `TimingConstraint` is a `Traceable` ⌈ The element `TimingConstraint` and all of its specializations, commonly called timing constraints, are traceable. ⌋*(RS_TIMEX_00010)*

The support for traceability [4] enables one to specify for example a relationship between an RTE event activating a runnable entity and a given timing constraint. A system integrator has chosen the RTE event `TimingEvent` with a period of 20ms, because of a given timing requirement respectively constraint `PeriodicEventTriggering` requiring the periodic activation of a runnable entity every 20ms. In this case, a trace from the RTE event's document section to the corresponding timing constraint can be set. In addition this capability ensures validity between constraints and properties.

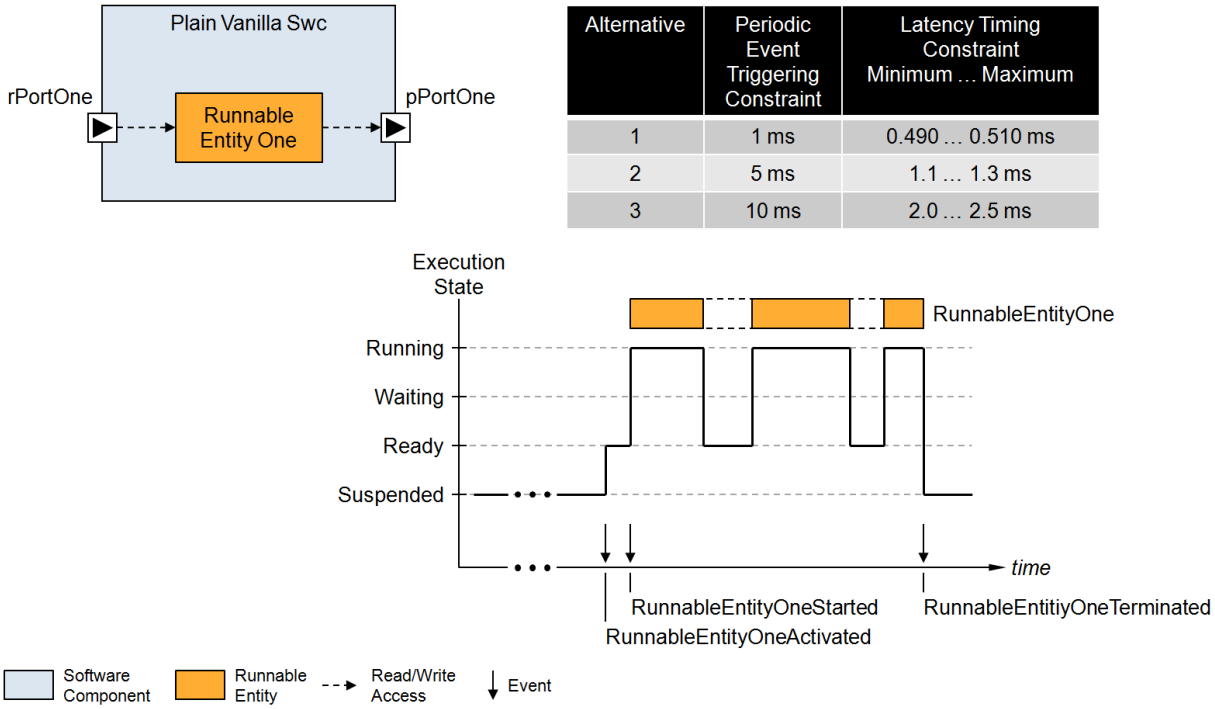
4.4 Specifying Time Sets

Sometimes one likes to specify that there are several alternatives with regard to timing requirements. For example, quite often it is reasonable to specify that a runnable entity shall be periodically activated either at 1ms, 2ms, 5ms, 8ms, or 10ms. In other words, it is perfectly fine to decide that the runnable entity is activated every 8ms. Indeed, it is allowed to activate the runnable entity either at 1ms, 2ms, 5ms, 8ms, or 10ms. Hence, there should be a means to specify such time sets which contain all allowed timings, like in case of activating a runnable entity at {1, 2, 5, 8, 10} ms.

For the purpose of specifying time sets the timing extensions utilize the ”Variant Handling” capabilities specified and described in [5]. In the following sub-section an example is given describing how to specify time sets with the mentioned capabilities.

4.4.1 Example

As shown in Figure 4.3 a software component called ”PlainVanillaSwc” consists of one runnable entity named ”RunnableEntityOne”. This runnable entity calculates the value of the variable ”DataElementTwo” based on the value of the variable ”DataElementOne”. The latter variable is received via the required port called ”rPortOne” and the former one is written to the provided port named ”pPortOne”. In the example three alternatives for activating the runnable entity are specified and shown in the table on the right hand side in Figure 4.3.



Alternative	Periodic Event Triggering Constraint	Latency Timing Constraint Minimum ... Maximum
1	1 ms	0.490 ... 0.510 ms
2	5 ms	1.1 ... 1.3 ms
3	10 ms	2.0 ... 2.5 ms

Figure 4.3: Example of a set of timings specified for a Runnable Entity.

In listing 4.1 the part of the AUTOSAR XML is given representing the software component description. It describes the software component including ports, interfaces, data elements, and its internal behavior respectively the runnable entity and the data accesses carried out by the runnable entity; as shown in Figure 4.3.

Listing 4.1: Software component description of the software component ”PlainVanillaSwc”

```
<AR-PACKAGE>
  <SHORT-NAME>TimeSetExample</SHORT-NAME>
  <ELEMENTS>
    <APPLICATION-SW-COMPONENT-TYPE>
      <SHORT-NAME>PlainVanillaSwc</SHORT-NAME>
      <SHORT-NAME-PATTERN />
      <PORTS>
        <R-PORT-PROTOTYPE>
          <SHORT-NAME>rPortOne</SHORT-NAME>
          <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
              TimeSetExample/srInterfaceOne</REQUIRED-INTERFACE-TREF>
        </R-PORT-PROTOTYPE>
        <P-PORT-PROTOTYPE>
          <SHORT-NAME>pPortOne</SHORT-NAME>
          <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
              TimeSetExample/srInterfaceTwo</PROVIDED-INTERFACE-TREF>
        </P-PORT-PROTOTYPE>
      </PORTS>
      <INTERNAL-BEHAVIORS>
        <SWC-INTERNAL-BEHAVIOR>
          <SHORT-NAME>InternalBehaviorPlainVanillaSwc</SHORT-NAME>
          <RUNNABLES>
```

```
            <RUNNABLE-ENTITY>
              <SHORT-NAME>RunnableEntityOne</SHORT-NAME>
              <DATA-READ-ACCESSS>
                <VARIABLE-ACCESS>
                  <SHORT-NAME>draDataElementOne</SHORT-NAME>
                </VARIABLE-ACCESS>
              </DATA-READ-ACCESSS>
              <DATA-WRITE-ACCESSS>
                <VARIABLE-ACCESS>
                  <SHORT-NAME>dwaDataElementTwo</SHORT-NAME>
                </VARIABLE-ACCESS>
              </DATA-WRITE-ACCESSS>
            </RUNNABLE-ENTITY>
          </RUNNABLES>
        </SWC-INTERNAL-BEHAVIOR>
      </INTERNAL-BEHAVIORS>
    </APPLICATION-SW-COMPONENT-TYPE>
    <SENDER-RECEIVER-INTERFACE>
      <SHORT-NAME>srInterfaceOne</SHORT-NAME>
      <DATA-ELEMENTS>
        <VARIABLE-DATA-PROTOTYPE>
          <SHORT-NAME>dataElementOne</SHORT-NAME>
        </VARIABLE-DATA-PROTOTYPE>
      </DATA-ELEMENTS>
    </SENDER-RECEIVER-INTERFACE>
    <SENDER-RECEIVER-INTERFACE>
      <SHORT-NAME>srInterfaceTwo</SHORT-NAME>
      <DATA-ELEMENTS>
        <VARIABLE-DATA-PROTOTYPE>
          <SHORT-NAME>dataElementTwo</SHORT-NAME>
        </VARIABLE-DATA-PROTOTYPE>
      </DATA-ELEMENTS>
    </SENDER-RECEIVER-INTERFACE>
  </ELEMENTS>
</AR-PACKAGE>
```

The second part of the AUTOSAR XML, shown in listing 4.2, represents the software component timing called ”swctPlainVanillaSwc”. It specifies the timing descriptions and timing requirements. The timing descriptions describe the events that are related to the runnable entity of the software component ”PlainVanillaSwc”, in particular the activation event ”RunnableEntityOneActivated”, start event ”RunnableEntityOneStarted” and termination event ”RunnableEntityOneTerminated” of the runnable entity ”RunnableEntityOne”. In addition, a timing description event chain ”CalculateValueOfDataElementTwo” is specified that describes the causal relationship between the events ”RunnableEntityOneActivated” and ”RunnableEntityOneTerminated”.

The timing requirements specify three variants of the periodic event triggering constraint named ”PeriodicActivationRunnableEntityOne” describing the *periodic* activation of the runnable entity ”RunnableEntityOne”. The following alternatives are specified: 1ms, 5ms and 10ms. In addition, three alternatives for latency timing constraints, named ”ResponseTimeForCalculatingDataElementTwo”, are specified that are imposed on the response time of the runnable entity. Technically, the scope of

this timing constraint is the event chain ”CalculateValueOfDataElementTwo” mentioned before.

Listing 4.2: Software component timing of the software component ”PlainVanillaSwc”

```
<AR-PACKAGE>
  <SHORT-NAME>TimingExtensions</SHORT-NAME>
  <ELEMENTS>
    <SWC-TIMING>
      <SHORT-NAME>swctPlainVanillaSwc</SHORT-NAME>
      <TIMING-DESCRIPTIONS>
        <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
          <SHORT-NAME>RunnableEntityOneActivated</SHORT-NAME>
          <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/TimeSetExample/
              PlainVanillaSwc/InternalBehaviorPlainVanillaSwc/
              RunnableEntityOne</RUNNABLE-REF>
          <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-
              ACTIVATED</TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
        </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
        <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
          <SHORT-NAME>RunnableEntityOneTerminated</SHORT-NAME>
          <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/TimeSetExample/
              PlainVanillaSwc/InternalBehaviorPlainVanillaSwc/
              RunnableEntityOne</RUNNABLE-REF>
          <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-
              TERMINATED</TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
        </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
        <TIMING-DESCRIPTION-EVENT-CHAIN>
          <SHORT-NAME>CalculateValueOfDataElementTwo</SHORT-NAME>
          <STIMULUS-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/
              TimingExtensions/swctPlainVanillaSwc/
              RunnableEntityOneActivated</STIMULUS-REF>
          <RESPONSE-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/
              TimingExtensions/swctPlainVanillaSwc/
              RunnableEntityOneTerminated</RESPONSE-REF>
          <SEGMENT-REFS>
            <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/
                TimingExtensions/swctPlainVanillaSwc/
                CalculateValueOfDataElementTwo</SEGMENT-REF>
          </SEGMENT-REFS>
        </TIMING-DESCRIPTION-EVENT-CHAIN>
      </TIMING-DESCRIPTIONS>
      <TIMING-REQUIREMENTS>
        <PERIODIC-EVENT-TRIGGERING>
          <SHORT-NAME>PeriodicActivationRunnableEntityOne</SHORT-NAME>
          <VARIATION-POINT>
            <SHORT-LABEL>VpPetTiming1</SHORT-LABEL>
            <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF
                DEST="SW-SYSTEMCONST">/SystemConstants/
                ScTimingRunnableEntityOne</SYSC-REF>==1)</SW-SYSCOND>
          </VARIATION-POINT>
          <EVENT-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/
              TimingExtensions/swctPlainVanillaSwc/
              RunnableEntityOneActivated</EVENT-REF>
          <PERIOD>
            <CSE-CODE>3</CSE-CODE>
```

```
      <CSE-CODE-FACTOR>1</CSE-CODE-FACTOR>
    </PERIOD>
  </PERIODIC-EVENT-TRIGGERING>
  <PERIODIC-EVENT-TRIGGERING>
    <SHORT-NAME>PeriodicActivationRunnableEntityOne</SHORT-NAME>
    <VARIATION-POINT>
      <SHORT-LABEL>VpPetTiming2</SHORT-LABEL>
      <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF
          DEST="SW-SYSTEMCONST">/SystemConstants/
          ScTimingRunnableEntityOne</SYSC-REF>==2)</SW-SYSCOND>
    </VARIATION-POINT>
    <EVENT-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/
        TimingExtensions/swctPlainVanillaSwc/
        RunnableEntityOneActivated</EVENT-REF>
    <PERIOD>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>5</CSE-CODE-FACTOR>
    </PERIOD>
  </PERIODIC-EVENT-TRIGGERING>
  <PERIODIC-EVENT-TRIGGERING>
    <SHORT-NAME>PeriodicActivationRunnableEntityOne</SHORT-NAME>
    <VARIATION-POINT>
      <SHORT-LABEL>VpPetTiming3</SHORT-LABEL>
      <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF
          DEST="SW-SYSTEMCONST">/SystemConstants/
          ScTimingRunnableEntityOne</SYSC-REF>==3)</SW-SYSCOND>
    </VARIATION-POINT>
    <EVENT-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/
        TimingExtensions/swctPlainVanillaSwc/
        RunnableEntityOneActivated</EVENT-REF>
    <PERIOD>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
    </PERIOD>
  </PERIODIC-EVENT-TRIGGERING>
  <LATENCY-TIMING-CONSTRAINT>
    <SHORT-NAME>ResponseTimeForCalculatingDataElementTwo</SHORT-
        NAME>
    <VARIATION-POINT>
      <SHORT-LABEL>VpLtcTiming1</SHORT-LABEL>
      <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF
          DEST="SW-SYSTEMCONST">/SystemConstants/
          ScTimingRunnableEntityOne</SYSC-REF>==1)</SW-SYSCOND>
    </VARIATION-POINT>
    <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
    <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/
        TimingExtensions/swctPlainVanillaSwc/
        CalculateValueOfDataElementTwo</SCOPE-REF>
    <MINIMUM>
      <CSE-CODE>2</CSE-CODE>
      <CSE-CODE-FACTOR>490</CSE-CODE-FACTOR>
    </MINIMUM>
    <MAXIMUM>
      <CSE-CODE>2</CSE-CODE>
      <CSE-CODE-FACTOR>510</CSE-CODE-FACTOR>
    </MAXIMUM>
```

```
        </LATENCY-TIMING-CONSTRAINT>
        <LATENCY-TIMING-CONSTRAINT>
          <SHORT-NAME>ResponseTimeForCalculatingDataElementTwo</SHORT-
              NAME>
          <VARIATION-POINT>
            <SHORT-LABEL>VpLtcTiming2</SHORT-LABEL>
            <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF
                DEST="SW-SYSTEMCONST">/SystemConstants/
                ScTimingRunnableEntityOne</SYSC-REF>==2)</SW-SYSCOND>
          </VARIATION-POINT>
          <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
          <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/
              TimingExtensions/swctPlainVanillaSwc/
              CalculateValueOfDataElementTwo</SCOPE-REF>
          <MINIMUM>
            <CSE-CODE>2</CSE-CODE>
            <CSE-CODE-FACTOR>1100</CSE-CODE-FACTOR>
          </MINIMUM>
          <MAXIMUM>
            <CSE-CODE>2</CSE-CODE>
            <CSE-CODE-FACTOR>1300</CSE-CODE-FACTOR>
          </MAXIMUM>
        </LATENCY-TIMING-CONSTRAINT>
        <LATENCY-TIMING-CONSTRAINT>
          <SHORT-NAME>ResponseTimeForCalculatingDataElementTwo</SHORT-
              NAME>
          <VARIATION-POINT>
            <SHORT-LABEL>VpLtcTiming3</SHORT-LABEL>
            <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF
                DEST="SW-SYSTEMCONST">/SystemConstants/
                ScTimingRunnableEntityOne</SYSC-REF>==3)</SW-SYSCOND>
          </VARIATION-POINT>
          <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
          <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/
              TimingExtensions/swctPlainVanillaSwc/
              CalculateValueOfDataElementTwo</SCOPE-REF>
          <MINIMUM>
            <CSE-CODE>2</CSE-CODE>
            <CSE-CODE-FACTOR>2000</CSE-CODE-FACTOR>
          </MINIMUM>
          <MAXIMUM>
            <CSE-CODE>2</CSE-CODE>
            <CSE-CODE-FACTOR>2500</CSE-CODE-FACTOR>
          </MAXIMUM>
        </LATENCY-TIMING-CONSTRAINT>
      </TIMING-REQUIREMENTS>
      <BEHAVIOR-REF DEST="SWC-INTERNAL-BEHAVIOR">/TimeSetExample/
          PlainVanillaSwc/InternalBehaviorPlainVanillaSwc</BEHAVIOR-REF>
    </SWC-TIMING>
  </ELEMENTS>
</AR-PACKAGE>
```

The third and last part of the AUTOSAR XML, shown in listing 4.3, specifies the system constant called ”ScTimingRunnableEntityOne” that eventually is used to select the specific variant of the required timing—the activation frequency and response time of

the runnable entity. The selected variant is specified by the system constant's value in the SwSystemconstantValueSet. In the given example the first variant "Timing1" is selected.

Listing 4.3: Description of the timing variations of the given example

```
<AR-PACKAGE>
  <SHORT-NAME>SystemConstants</SHORT-NAME>
  <ELEMENTS>
    <SW-SYSTEMCONST>
      <SHORT-NAME>ScTimingRunnableEntityOne</SHORT-NAME>
    </SW-SYSTEMCONST>
  </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
  <SHORT-NAME>Variants</SHORT-NAME>
  <ELEMENTS>
    <SW-SYSTEMCONSTANT-VALUE-SET>
      <SHORT-NAME>TIMEX_EXP_Time_Set_VariantValues</SHORT-NAME>
      <SW-SYSTEMCONSTANT-VALUES>
        <SW-SYSTEMCONST-VALUE>
  <!-- The values of the system constant ScTimingRunnableEntityOne may
      range from 1 to 3. -->
          <SW-SYSTEMCONST-REF DEST="SW-SYSTEMCONST">/SystemConstants/
              ScTimingRunnableEntityOne</SW-SYSTEMCONST-REF>
          <VALUE>1</VALUE>
        </SW-SYSTEMCONST-VALUE>
      </SW-SYSTEMCONSTANT-VALUES>
    </SW-SYSTEMCONSTANT-VALUE-SET>
    <POST-BUILD-VARIANT-CRITERION-VALUE-SET>
      <SHORT-NAME>TIMEX_EXP_Time_Set_VariantPostBuildValues</SHORT-NAME
          >
    </POST-BUILD-VARIANT-CRITERION-VALUE-SET>
    <PREDEFINED-VARIANT>
      <SHORT-NAME>TIMEX_EXP_Time_Set_Variant</SHORT-NAME>
      <POST-BUILD-VARIANT-CRITERION-VALUE-SET-REFS>
        <POST-BUILD-VARIANT-CRITERION-VALUE-SET-REF DEST="POST-BUILD-
            VARIANT-CRITERION-VALUE-SET">/Variants/
            TIMEX_EXP_Time_Set_VariantPostBuildValues</POST-BUILD-
            VARIANT-CRITERION-VALUE-SET-REF>
      </POST-BUILD-VARIANT-CRITERION-VALUE-SET-REFS>
      <SW-SYSTEMCONSTANT-VALUE-SET-REFS>
        <SW-SYSTEMCONSTANT-VALUE-SET-REF DEST="SW-SYSTEMCONSTANT-VALUE-
            SET">/Variants/TIMEX_EXP_Time_Set_VariantValues</SW-
            SYSTEMCONSTANT-VALUE-SET-REF>
      </SW-SYSTEMCONSTANT-VALUE-SET-REFS>
    </PREDEFINED-VARIANT>
  </ELEMENTS>
</AR-PACKAGE>
```

4.5 Conditional Timing

In almost all cases, systems, and the application software executed in those systems, operate under various conditions, like normal condition, error condition, start-up condition, etc. During the operation of such systems the conditions may change at any time. As a consequence timing constraints imposed on the system may vary depending on these conditions, too. The AUTOSAR Timing Extensions provide means to support the description of timing constraints depending on such conditions by *Conditional Timing* as described in this section.

In the following some examples are given that demonstrate the necessity for conditional timing constraints.

Almost all software management systems controlling an internal combustion engine must maintain a constant temperature of the coolant to ensure the optimal operation under specific conditions. For example, critical thermal conditions may occur that may lead to severe damage of the engine's mechanical components. In order to prevent the engine's mechanical components from being damaged the software application shall respond faster than in the nominal case. Therefore, different timing constraints are imposed on a given event chain depending on a known condition.

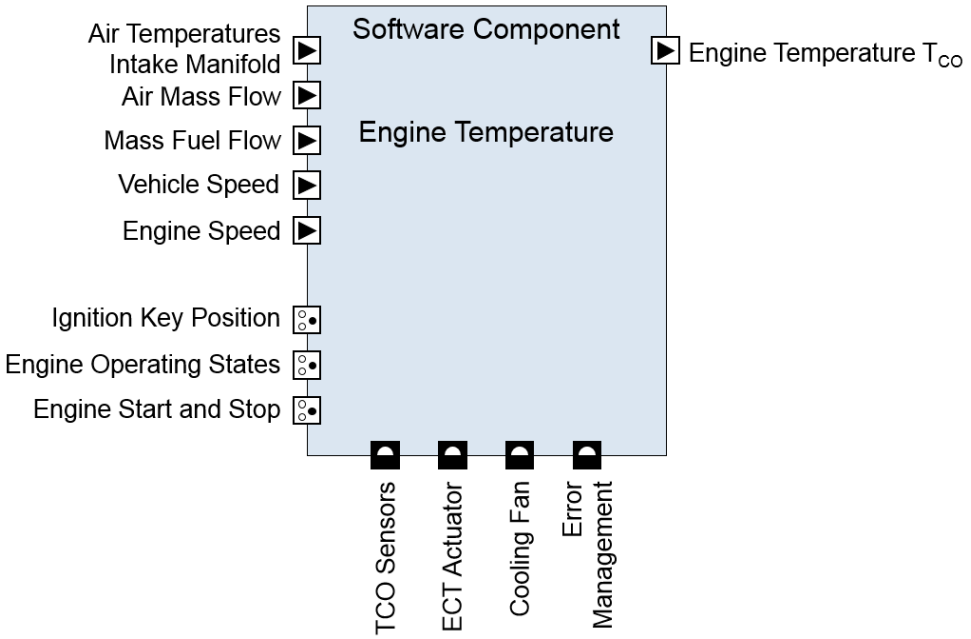


Figure 4.4: Rough sketch of an AUTOSAR software component (SW-C) controlling the coolant temperature (TCO) of an internal combustion engine. This SW-C must consider various modes of operation as shown in the lower left corner of the components, namely the mode ports called "Ignition Key Position", "Engine Operating States", and "Engine Start and Stop".

As shown in Figure 4.4 the software component controlling the coolant temperature requires information on the current operating conditions of the internal combustion engine. Depending on these conditions different timing constraints are imposed on an event chain requiring different reaction time constraints. In one situation, the reac-

tion time must be much faster than in other situation. For example, the mode port called ”Engine Operating States” indicates a specific condition respectively *mode* the internal combustion engine operates in. In this specific mode it may happen that the internal combustion engine faces a critical thermal condition. In order to prevent the engine from being damaged the Electronically Controlled Thermostat (ETC) must be actuated much faster (client port called ”ECT Actuator”) than in other cases. This is accomplished by specifying a tighter reaction time for this condition; and under any other condition the reaction time may be more relaxed. Besides this particular example, there are a lot more cases where the timing constraints imposed on given event chains change depending on the internal combustion engines mode.

In dependable systems an application is typically designed in a fault tolerant manner. It either operates under normal condition — normal mode — or under failure condition — failure mode. As sketched in the upper part of Figure 4.5 in such an application a number of runnable entities process the values of three sensors. In a first step the sensor values are corrected by a runnable entity and are passed to another runnable entity of the application that utilizes a control algorithm to calculate an output. This output value is used to control an actuator or is processed by other runnable entities in the system. In the same application another runnable entity checks the plausibility of the values received from the sensors. The purpose of this runnable entity is to determine any unexpected and implausible deviation from the expected values of each sensor. If such an deviation is detected by the runnable entity checking the plausibility of the sensor values then a runnable entity correcting the sensor values is notified about this condition, namely indicating a failure of one of the sensors. The runnable entity correcting the sensor values is capable of deriving the value of the erroneous sensor based on the values of the other sensors which monitor other physical properties. Thus, the application is operating under different conditions — normal and failure mode — and with regard to the software implementation this results in different worst case execution times of the runnable entity correcting the sensor values.

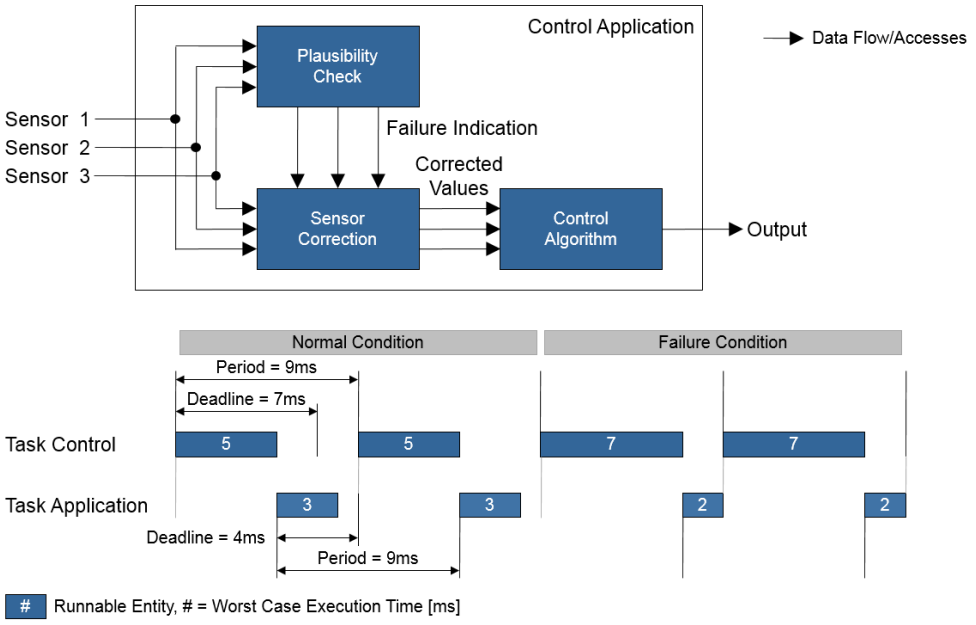


Figure 4.5: Generic fault tolerant control application

Assumed the application executing the three runnable entities is mapped to an ECU executing already runnable entities of another application. The runnable entities of these two applications are executed in the context of two tasks called ”Task Control” and ”Task Application”. The former task executes the runnable entities of the given fault tolerant application and the latter task executes the application already being mapped to the ECU and consisting of a number of runnable entities. Both tasks respectively applications are activated periodically every 9ms. The task ”Task Control” has a deadline of 7ms and the task ”Task Application” has a deadline of 4ms. As mentioned above the first application operates under two different conditions: Under normal condition the execution time of the runnable entities does not exceed 5ms, and under failure condition the execution time of the runnable entities increases to 7ms due to the additional time required for executing the algorithm to derive the value of the erroneous sensor.

A rough sketch of the task schedule containing both tasks is given in the lower part of Figure 4.5. On the left hand side the fault tolerant application is operating under normal conditions and on the right hand side this application is operating under failure condition. In this case — failure condition — the execution time (3ms) of the task ”Task Application” leads to a violation of the schedule, which means that the two applications cannot be integrated onto the same ECU unless both applications are mapped onto a more powerful ECU — resulting in a decrease of execution times. Another possibility would be that the application executed by the task ”Task Application” provides capabilities to operate under degraded conditions resulting in a shorter execution time, for example 2ms.

However, in the cases described above timing constraints imposed on timing properties of software components and/or runnable entities depend on specific conditions a system must operate properly. The AUTOSAR Timing Extensions are capable of specifying timing constraints for specific conditions as shown in Figure 4.6. Such a timing constraint is valid if and only if the given timing condition holds.

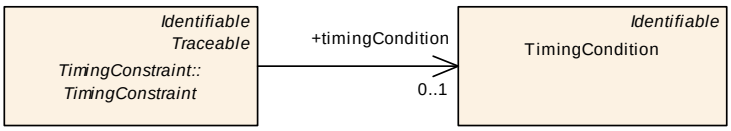


Figure 4.6: Conditional Timing Constraint

Since a `TimingConstraint` plays either the role of a timing requirement or timing guarantee the dependency on a condition can be specified on both types of timing information, namely timing constraint and timing property.

[TPS_TIMEX_00049] Purpose of `TimingCondition` ⌈ The purpose of the `TimingCondition` is to describe a condition a timing constraint is depending on. ⌋ *(RS_TIMEX_00011)*

[TPS_TIMEX_00050] Purpose of `TimingConditionFormula` ⌈ The purpose of the `TimingConditionFormula` is to specify an expression describing a dependency on a specific condition. ⌋ *(RS_TIMEX_00011)*

[TPS_TIMEX_00051] Purpose of `TimingExtensionResource` ⌈ The purpose of the `TimingExtensionResource` is to subsume a number of re-usable elements, like instance references to various timing relevant elements, that are referenced from within a `TimingConditionFormula`. These elements of a `TimingExtensionResource` are related to the context of the aggregating — *parent* — `TimingExtension`. ⌋*(RS_TIMEX_00011)*

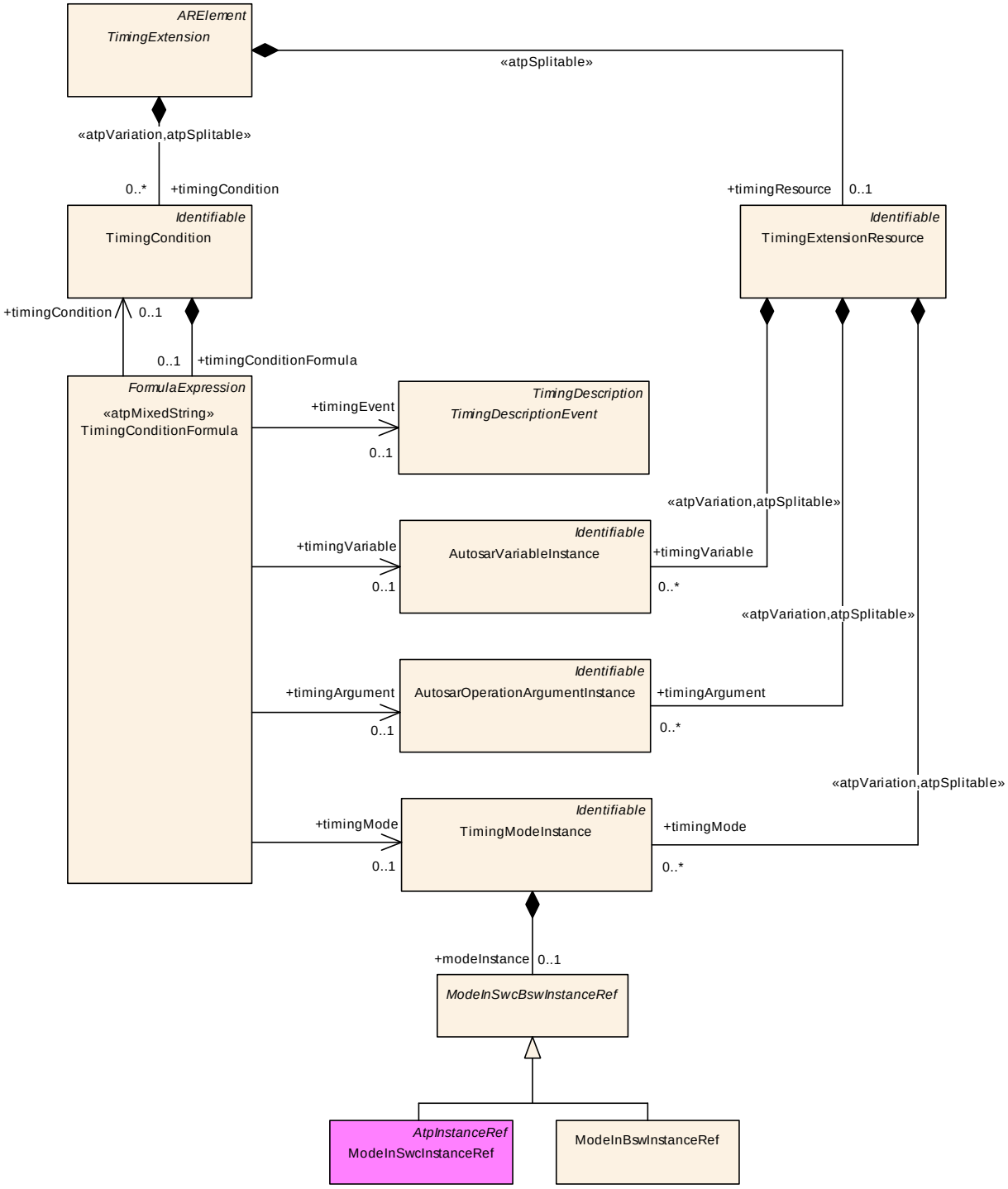


Figure 4.7: Conditional Timing

Class	**TimingCondition**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConditional			
Note	A TimingCondition describes a dependency on a specific condition. The element owns an expression which describes the timing condition dependency.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
timingCondition Formula	TimingCondition Formula	0..1	aggr	This is the expression describing the dependency on a specific condition.

Table 4.1: TimingCondition

Class	«atpMixedString» **TimingConditionFormula**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConditional			
Note	A TimingConditionFormula describes a specific dependency. The expression shall be a boolean expression addressing modes, variables, arguments, and/or events.			
Base	*ARObject*, *FormulaExpression*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
timingArgument	AutosarOperation ArgumentInstance	0..1	ref	This refers to an argument of an operation call.
timingCondition	TimingCondition	0..1	ref	This refers to a timing condition that is part of an expression describing the dependency on a specific condition.
timingEvent	TimingDescriptionEvent	0..1	ref	This refers to a timing event.
timingMode	TimingModeInstance	0..1	ref	This refers to a mode declaration.
timingVariable	AutosarVariable Instance	0..1	ref	This refers to a variable.

Table 4.2: TimingConditionFormula

Class	**TimingExtensionResource**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConditional			
Note	A TimingExtensionResource provides the capability to contain instance references referred from within a timing condition formula.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
timingArgument	AutosarOperation ArgumentInstance	*	aggr	This refers to an instance reference of an argument of an operation call. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
timingMode	TimingModeInstance	*	aggr	This refers to an instance reference of a mode declaration. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
timingVariable	AutosarVariable Instance	*	aggr	This refers to an instance reference of a variable. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild

Table 4.3: TimingExtensionResource

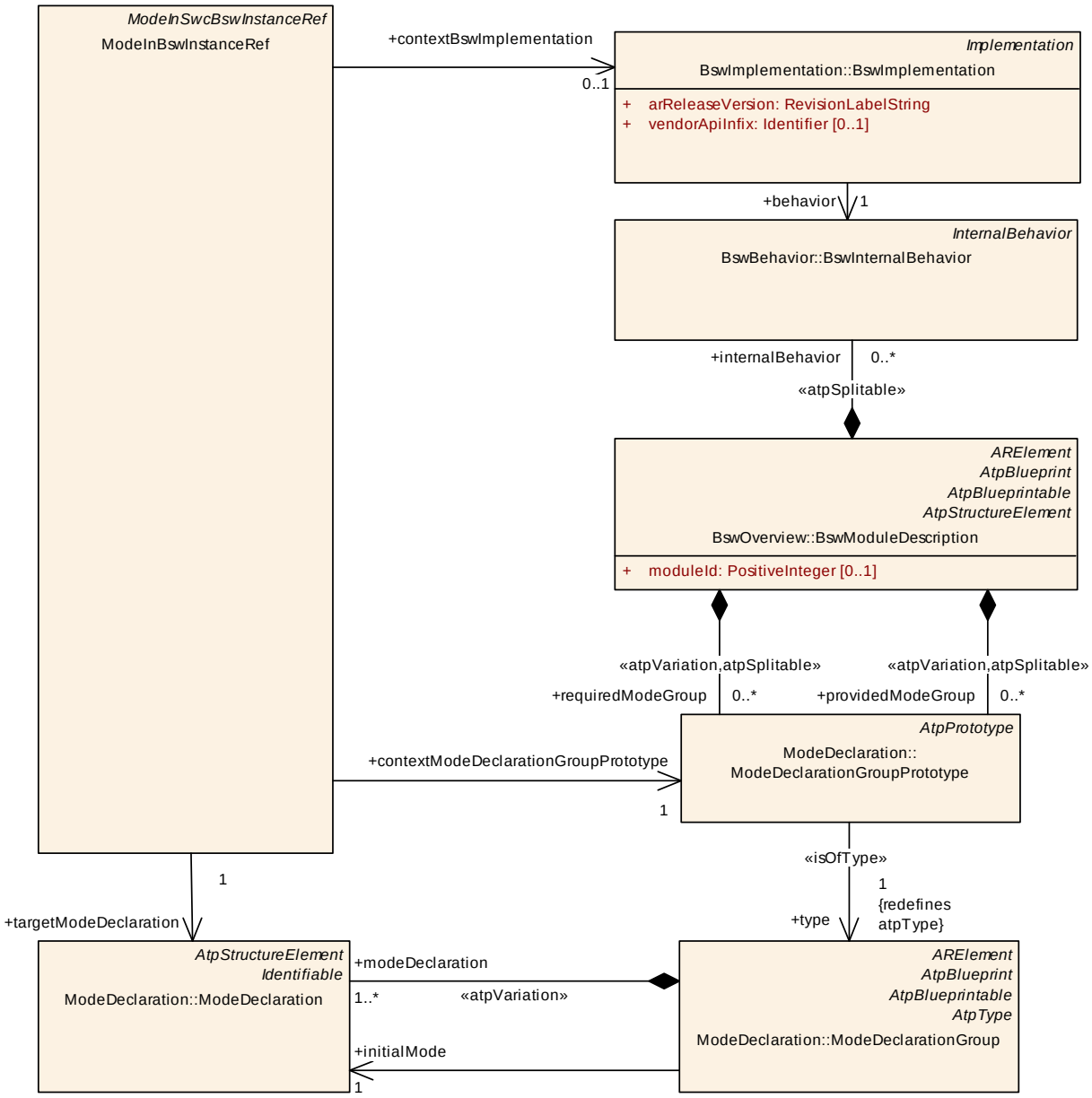


Figure 4.8: Instance Reference Mode for Basic Software Module

Since the notion of ”Type and Prototype” is not supported by the Basic Sofware Module Description Template, the element `ModeInBswInstanceRef` is not a specialization of `AtpInstanceRef`. Therefore, the directed association playing the role of ”base” is not present.

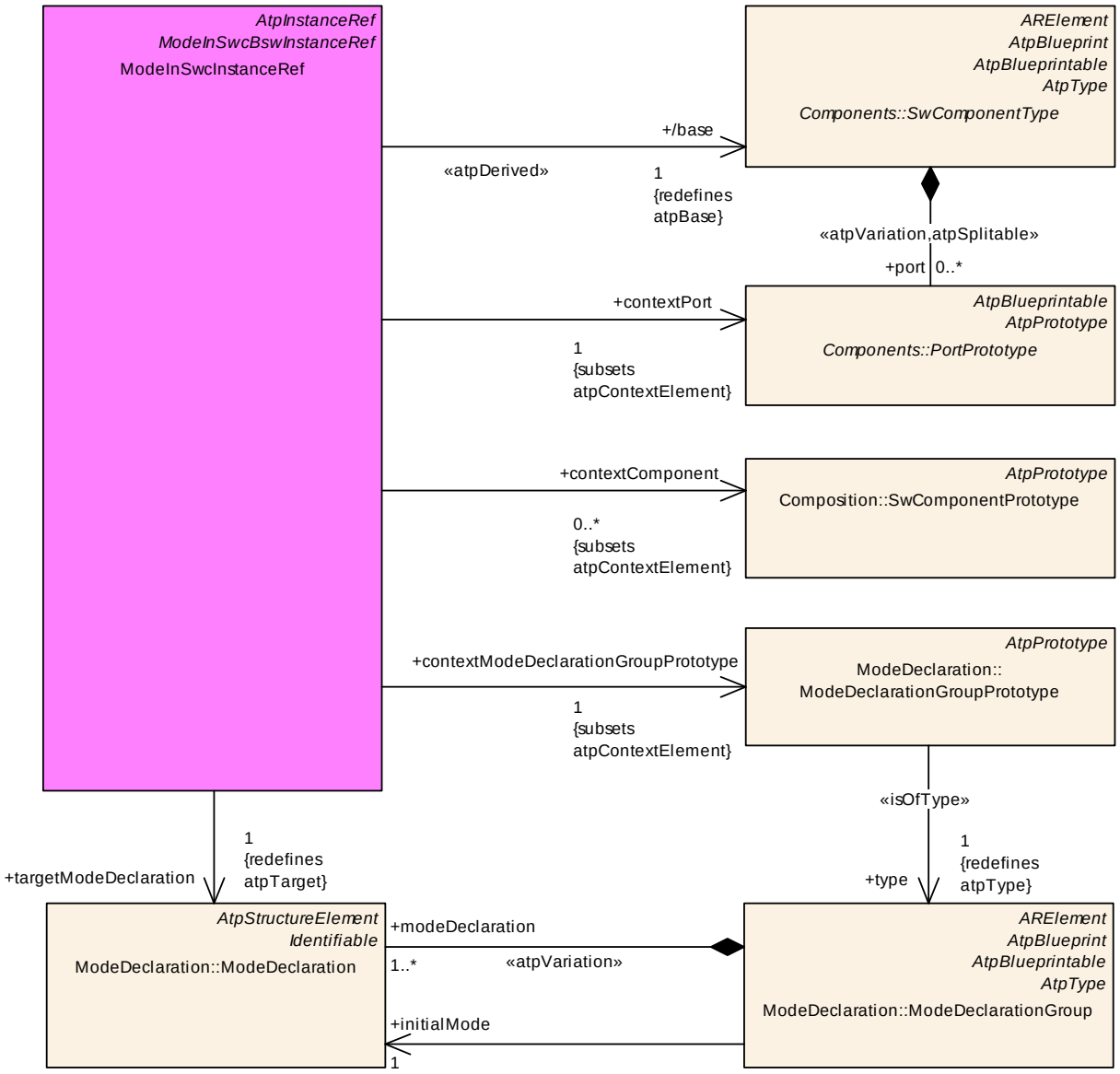


Figure 4.9: Instance Reference Mode for Software Component

Class	**TimingModeInstance**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConditional			
Note	This class specifies the mode declaration to be checked in a specific instance of a mode declaration group. This is used in a timing condition formula as an operand of the unary timing function TIMEX_modeActive to check whether the mode declaration is active at the point in time this expression is evaluated.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
modeInstance	ModeInSwcBsw InstanceRef	0..1	aggr	This refers to a specific mode declaration in the given context.

Table 4.4: TimingModeInstance

Class	**ModeInBswInstanceRef**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConditional			
Note	Instance reference to be capable of referencing a specific ModeDeclaration of a ModeDeclarationGroup Prototype utilized in a BSW module.			
Base	*ARObject*, *ModeInSwcBswInstanceRef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
contextBsw Implementation	BswImplementation	0..1	ref	Specifies the BSW implementation that manifests the context. **Tags:** xml.sequenceOffset=10
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	1	ref	Specifies the mode declaration group prototype that manifests the context. **Tags:** xml.sequenceOffset=20
targetMode Declaration	ModeDeclaration	1	ref	Specifies the specific mode declaration in the given context. **Tags:** xml.sequenceOffset=30

Table 4.5: ModeInBswInstanceRef

Class	**ModeInSwcInstanceRef**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConditional			
Note	Instance reference to be capable of referencing a ModeDeclaration at a specific Mode Switch Port of a SW-C.			
Base	*ARObject*, *AtpInstanceRef*, *ModeInSwcBswInstanceRef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	SwComponentType	1	ref	Specifies the SW component representing the base of the context. **Stereotypes:** atpDerived **Tags:** xml.sequenceOffset=10
context Component	SwComponent Prototype	*	ref	Specifies the SW component prototype representing the context. **Tags:** xml.sequenceOffset=20
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	1	ref	Specifies the mode declaration group prototype that manifests the context. **Tags:** xml.sequenceOffset=40
contextPort	PortPrototype	1	ref	Specifies the port prototype representing the context. **Tags:** xml.sequenceOffset=30
targetMode Declaration	ModeDeclaration	1	ref	Specifies the specific mode declaration in the given context. **Tags:** xml.sequenceOffset=50

Table 4.6: ModeInSwcInstanceRef

4.6 Logical Execution Time

The *Logical Execution Time* (LET) is a real-time programming *abstraction*. It abstracts from actual — *physical* — execution time of a real-time program. LET determines the time it takes from reading program input to writing program output regardless of the time it takes to execute the program which processes the input and writes output. LET is motivated by the observation that the relevant behavior of real-time programs

is determined by when input is read and output is written and not when programs just execute any code. This *logical* notion is shown in the upper part of Figure 4.10. At the *Release* point input is read and at the *Terminate* point output is written. Between these two points — the logical execution time — the program is executed, and writing output is delayed until the logical execution time elapsed even if the program completes execution before the logical execution time elapsed. This is sketched in the lower part of Figure 4.10 and shows the *physical* — actual — execution of a program.

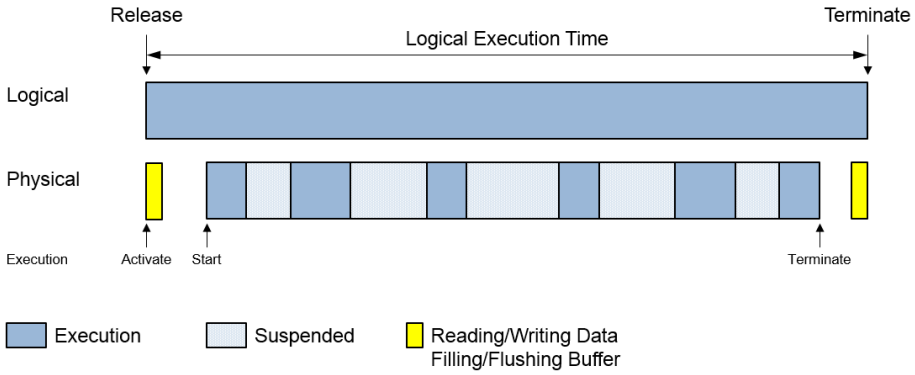


Figure 4.10: LET Approach

[constr_4558] Applicability of LET semantics ⌈ The LET semantics applies only to *implicit* sender-receiver communication. ⌋*()*

4.6.1 Specifying LET Interval

As sketched in Figure 4.11 a LET Interval is specified by a timing description event chain `TimingDescriptionEventChain` and the value of the element's attribute `category` is `LET_INTERVAL`. The timing description event chain references two timing description events `TimingDescriptionEvent` representing the release point (stimulus event of the timing description event chain) and the terminate point (response event of the timing description event chain). The value of the first element's attribute `category` is `LET_RELEASE`; and the value of the latter element's attribute `category` is `LET_TERMINATE`. The complex timing event `TDEventComplex` is used for the purpose to indicate the release and terminate point. The complex timing event is used because both timing description events are not referencing an *observable* location, like the other types of timing description events. The event occurrence expression of the complex timing event may be used to specify the particular event supposed to occur in case of the release and terminate point respectively.

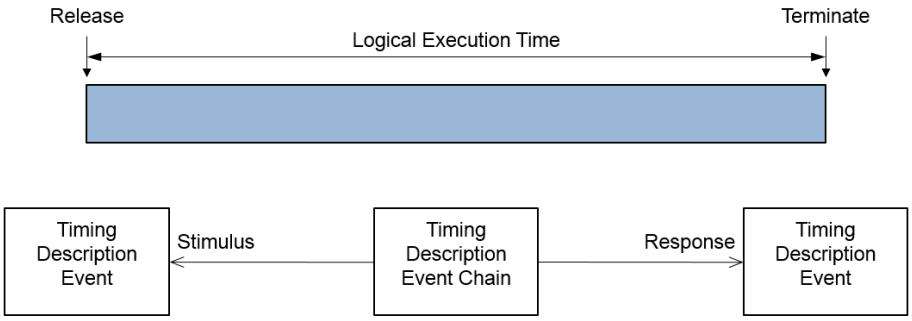


Figure 4.11: Specifying LET Interval

The following listing 4.4 shows an XML fragment that matches what is shown in Figure 4.11. It specifies a timing description event chain, and two timing description events representing the release and terminate point of this LET interval. The former timing description event plays the role *stimulus* and the latter timing description event plays the role *response* of the timing description event chain representing the LET interval.

Listing 4.4: Specifying LET Interval

```
<TD-EVENT-COMPLEX>
  <SHORT-NAME>LetIntervalRelease</SHORT-NAME>
  <CATEGORY>LET_RELEASE</CATEGORY>
</TD-EVENT-COMPLEX>
<TD-EVENT-COMPLEX>
  <SHORT-NAME>LetIntervalTerminate</SHORT-NAME>
  <CATEGORY>LET_TERMINATE</CATEGORY>
</TD-EVENT-COMPLEX>
<TIMING-DESCRIPTION-EVENT-CHAIN>
  <SHORT-NAME>tdecLetInterval</SHORT-NAME>
  <CATEGORY>LET_INTERVAL</CATEGORY>
  <STIMULUS-REF DEST="TD-EVENT-COMPLEX">/
     LogicalExecutionTime/TimingExtensions/
     LogicalExecutionTime02/LetIntervalRelease</STIMULUS-
     REF>
  <RESPONSE-REF DEST="TD-EVENT-COMPLEX">/
     LogicalExecutionTime/TimingExtensions/
     LogicalExecutionTime02/LetIntervalTerminate</RESPONSE-
     REF>
  <SEGMENT-REFS>
    <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/
       LogicalExecutionTime/TimingExtensions/
       LogicalExecutionTime02/tdecLetInterval</SEGMENT-REF>
  </SEGMENT-REFS>
</TIMING-DESCRIPTION-EVENT-CHAIN>
```

The timing parameters of a LET interval are specified using timing constraints as described in the following. The latency timing constraint `LatencyTimingConstraint` is used to specify the *length* of a LET interval, as shown in Figure 4.12. The type of the latency timing constraint shall be set to `REACTION`. The recurrence of a LET interval

is specified by the periodic event triggering constraint PeriodicEventTriggering which is described later in this section.

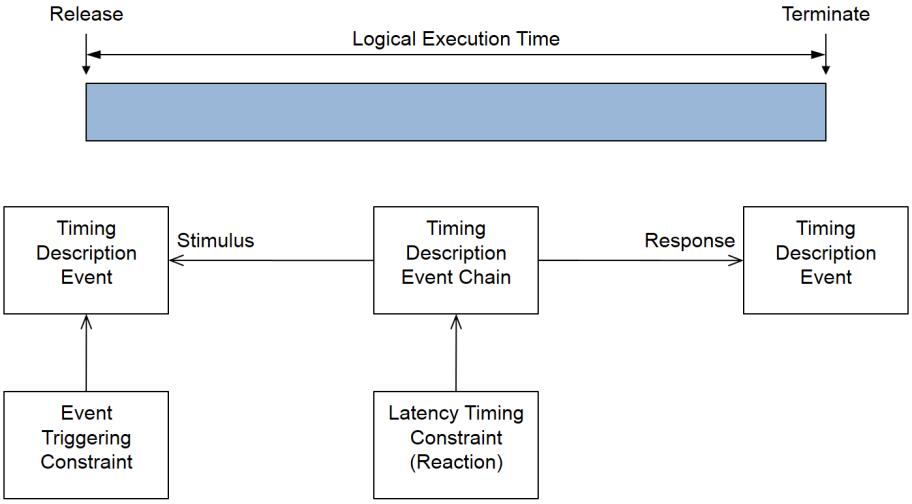


Figure 4.12: Specifying LET Interval Timing

For example, a latency timing constraint is imposed on a timing description event chain representing a LET interval and the attribute maximum specifies the length of the LET interval. Since the value of the attribute minimum has no meaning in the context of a LET interval it is defined that it must be the same as the value of the attribute maximum.

The following listing 4.5 shows an XML fragment which defines a latency timing constraint of 10ms that is imposed on the timing description event chain — LET interval — defined in listing 4.5. In essence, this latency timing constraint specifies the length of the LET interval to be 10ms.

Listing 4.5: Specifying LET Interval Timing

```
<LATENCY-TIMING-CONSTRAINT>
  <SHORT-NAME>ltcLetInterval</SHORT-NAME>
  <CATEGORY>LET_INTERVAL</CATEGORY>
  <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-
      TYPE>
  <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/
      LogicalExecutionTime/TimingExtensions/
      LogicalExecutionTime02/tdecLetInterval</SCOPE-REF>
  <MINIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
  </MAXIMUM>
</LATENCY-TIMING-CONSTRAINT>
```

The recurrence of a LET interval is specified using the periodic event triggering constraint PeriodicEventTriggering. This timing constraint is imposed on the stimu-

lus event — the release point of the LET interval — of the timing description event chain representing a LET interval. The following listing 4.6 shows an XML fragment which defines a periodic event triggering constraint that specifies a recurrence of 10ms; in other words the LET interval is repeated every 10ms.

Listing 4.6: Specifying Recurrence of a LET Interval

```
<PERIODIC-EVENT-TRIGGERING>
  <SHORT-NAME>petcLetIntervalRecurrence</SHORT-NAME>
  <EVENT-REF DEST="TD-EVENT-COMPLEX">/LogicalExecutionTime/
     TimingExtensions/LogicalExecutionTime02/
     LetIntervalRelease</EVENT-REF>
  <MINIMUM-INTER-ARRIVAL-TIME>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
  </MINIMUM-INTER-ARRIVAL-TIME>
  <JITTER>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
  </JITTER>
  <PERIOD>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
  </PERIOD>
</PERIODIC-EVENT-TRIGGERING>
```

4.6.2 Relation between LET Intervals

This section describes the most prominent cases for describing LET intervals and possible relationships among them. These cases are frequently occurring in developing systems. The section describes various ways to use the AUTOSAR elements `PeriodicEventTriggering` and `OffsetTimingConstraint` to specify the length of LET intervals and offsets between LET intervals. Offsets between LET intervals result in overlapping LET intervals or LET intervals in subsequent order with and without gaps between each other.

In Figure 4.13 a periodic LET interval is specified. The LET interval is periodically repeated and the value of the LET interval is the same as the period.

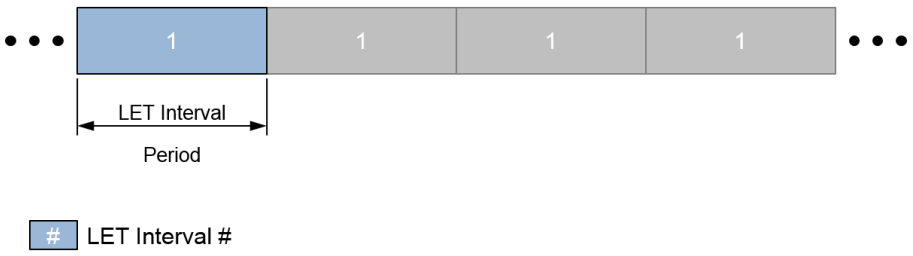


Figure 4.13: Specifying Periodic LET Interval

The corresponding XML fragments that describe this pattern are shown in listing 4.4, 4.5 and 4.6. The first listing presents an XML fragment that specifies the LET interval; the second listing presents an XML fragment that specifies the length of the LET interval; and the third listing presents an XML fragment that specifies the recurrence of the LET interval.

In Figure 4.14 a periodic LET interval is specified, but the value of the period is larger than the LET interval. This results in a gap between the *terminate* point of the LET interval and the next *release* point of the LET interval. The corresponding XML fragment is shown in listing 4.7. The first part of the XML fragments specifies the length of the LET interval which is 6ms by using the latency timing constraint `LatencyTimingConstraint`. The second part specifies the recurrence of the LET interval which is 10ms by using the periodic event triggering constraint `PeriodicEventTriggering`.

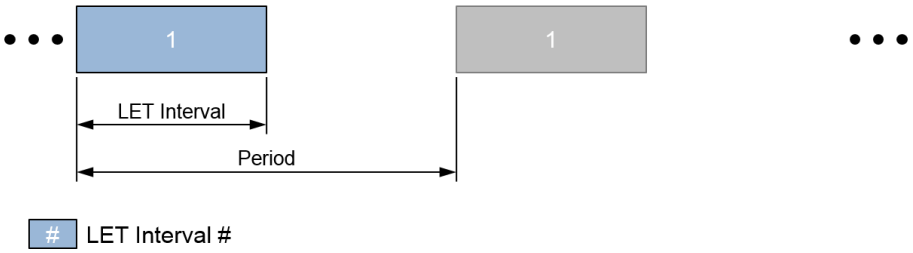


Figure 4.14: Specifying Periodic LET Interval with Gap

Listing 4.7: Periodic LET Interval with Gap

```
<LATENCY-TIMING-CONSTRAINT>
  <SHORT-NAME>ltcLetInterval_1</SHORT-NAME>
  <CATEGORY>LET_INTERVAL</CATEGORY>
  <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
  <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/
     SpecifyingPeriodicLetIntervalWithGap/
     sysTiming_SpecifyingPeriodicLetIntervalWithGap/
     tdecLetInterval_1</SCOPE-REF>
  <MINIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>6</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>6</CSE-CODE-FACTOR>
  </MAXIMUM>
</LATENCY-TIMING-CONSTRAINT>
<PERIODIC-EVENT-TRIGGERING>
  <SHORT-NAME>petcLetInterval_1</SHORT-NAME>
  <MINIMUM-INTER-ARRIVAL-TIME>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
  </MINIMUM-INTER-ARRIVAL-TIME>
  <JITTER>
    <CSE-CODE>3</CSE-CODE>
```

```
      <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
    </JITTER>
    <PERIOD>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
    </PERIOD>
  </PERIODIC-EVENT-TRIGGERING>
```

In Figure 4.15 three LET intervals are specified. The length of the first LET interval 1 is 2.5ms, the length of the second LET interval 2 is 5ms, and the length of the third LET interval 3 is 10ms. The very first release of each LET interval takes place at the same point in time and then each of the LET intervals is repeated periodically. This case is modeled using the Offset Timing Constraint `OffsetTimingConstraint` and the offset between the *release* points of those LET intervals is 0ms, namely no offset.

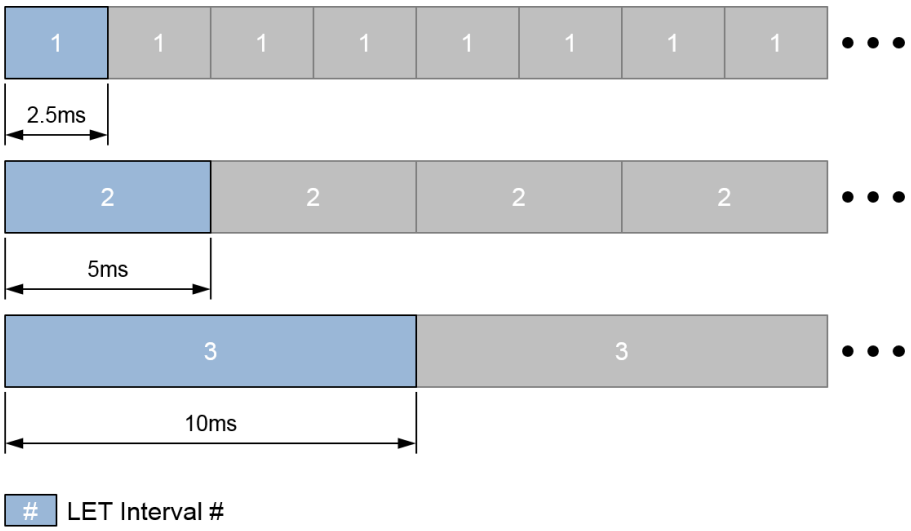


Figure 4.15: Specifying Relationship between Periodic LET Intervals

The following listing 4.8 shows an XML fragment which specifies two offset timing constraints between the *release* point of LET interval 1 and the two *release* points of the LET interval 2 and LET interval 3. In both cases the offset is zero (0ms) specifying that all LET intervals are released at the same point in time.

Listing 4.8: Periodic LET Intervals with no Offset

```
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_2</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/
      SpecifyingPeriodicLetIntervalsWithNoOffset/
      sysTiming_SpecifyingPeriodicLetIntervalsWithNoOffset/
      LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/
      SpecifyingPeriodicLetIntervalsWithNoOffset/
      sysTiming_SpecifyingPeriodicLetIntervalsWithNoOffset/
      LetIntervalRelease_2</TARGET-REF>
  <MINIMUM>
```

```
        <CSE-CODE>2</CSE-CODE>
        <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
      </MINIMUM>
      <MAXIMUM>
        <CSE-CODE>2</CSE-CODE>
        <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
      </MAXIMUM>
    </OFFSET-TIMING-CONSTRAINT>
    <OFFSET-TIMING-CONSTRAINT>
      <SHORT-NAME>otc_LetInterval_3</SHORT-NAME>
      <SOURCE-REF DEST="TD-EVENT-COMPLEX">/
          SpecifyingPeriodicLetIntervalsWithNoOffset/
          sysTiming_SpecifyingPeriodicLetIntervalsWithNoOffset/
          LetIntervalRelease_1</SOURCE-REF>
      <TARGET-REF DEST="TD-EVENT-COMPLEX">/
          SpecifyingPeriodicLetIntervalsWithNoOffset/
          sysTiming_SpecifyingPeriodicLetIntervalsWithNoOffset/
          LetIntervalRelease_3</TARGET-REF>
      <MINIMUM>
        <CSE-CODE>2</CSE-CODE>
        <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
      </MINIMUM>
      <MAXIMUM>
        <CSE-CODE>2</CSE-CODE>
        <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
      </MAXIMUM>
    </OFFSET-TIMING-CONSTRAINT>
```

In Figure 4.16 three LET intervals are specified. The length of the first LET interval 1 is 2.5ms, the length of the second LET interval 2 is 5ms, and the length of the third LET interval 3 is 10ms. The very first release of each LET interval takes place at different points in time and then each LET interval is repeated periodically . This case is modeled using the Offset Timing Constraint `OffsetTimingConstraint`. The offset between the *release* points of the first LET interval 1 and second LET interval 2 is 2.5ms; and the offset between the *release* points of the first LET interval 1 and third LET interval 3 is 7.5ms.

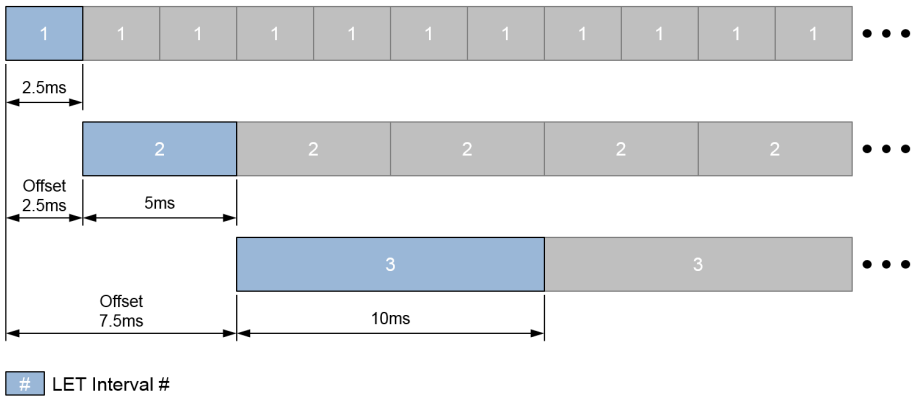


Figure 4.16: Specifying Relationship between Periodic LET Intervals With Offsets

The following listing 4.9 shows an XML fragment which specifies two offset timing constraints between the *release* point of LET interval 1 and the two *release* points of the LET interval 2 and LET interval 3. In the first case the offset between the *release* points of LET interval 1 and LET interval 2 is 2.5ms; and in the second case the offset between the *release* points of LET interval 1 and LET interval 3 is 7.5ms.

Listing 4.9: Periodic LET Intervals with Offset

```
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_2</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/
      SpecifyingPeriodicLetIntervalsWithOffset/
      sysTiming_SpecifyingPeriodicLetIntervalsWithOffset/
      LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/
      SpecifyingPeriodicLetIntervalsWithOffset/
      sysTiming_SpecifyingPeriodicLetIntervalsWithOffset/
      LetIntervalRelease_2</TARGET-REF>
  <MINIMUM>
    <CSE-CODE>2</CSE-CODE>
    <CSE-CODE-FACTOR>25</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>2</CSE-CODE>
    <CSE-CODE-FACTOR>25</CSE-CODE-FACTOR>
  </MAXIMUM>
</OFFSET-TIMING-CONSTRAINT>
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_3</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/
      SpecifyingPeriodicLetIntervalsWithOffset/
      sysTiming_SpecifyingPeriodicLetIntervalsWithOffset/
      LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/
      SpecifyingPeriodicLetIntervalsWithOffset/
      sysTiming_SpecifyingPeriodicLetIntervalsWithOffset/
      LetIntervalRelease_3</TARGET-REF>
  <MINIMUM>
    <CSE-CODE>2</CSE-CODE>
    <CSE-CODE-FACTOR>75</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>2</CSE-CODE>
    <CSE-CODE-FACTOR>75</CSE-CODE-FACTOR>
  </MAXIMUM>
</OFFSET-TIMING-CONSTRAINT>
```

In Figure 4.17 four LET intervals are specified. The first LET interval 1 terminates before the second LET interval 2 is released; the second LET interval 2 terminates before the third LET interval 3 is released; and the third LET interval 3 terminates before the fourth LET interval is released. After the last, the fourth LET interval 4 terminates, the same sequence of LET intervals is repeated. This case is modeled using the Offset Timing Constraint `OffsetTimingConstraint` to specify the offsets between the *release* point of the first LET interval 1 and the *release* points of each LET

interval 2 through 4. The length of each LET interval is listed in Table 4.7 along with the corresponding offsets between the *release* points of the LET intervals.

The periodic event triggering constraint `PeriodicEventTriggering` is used to specify that the period of the sequence is 32ms.

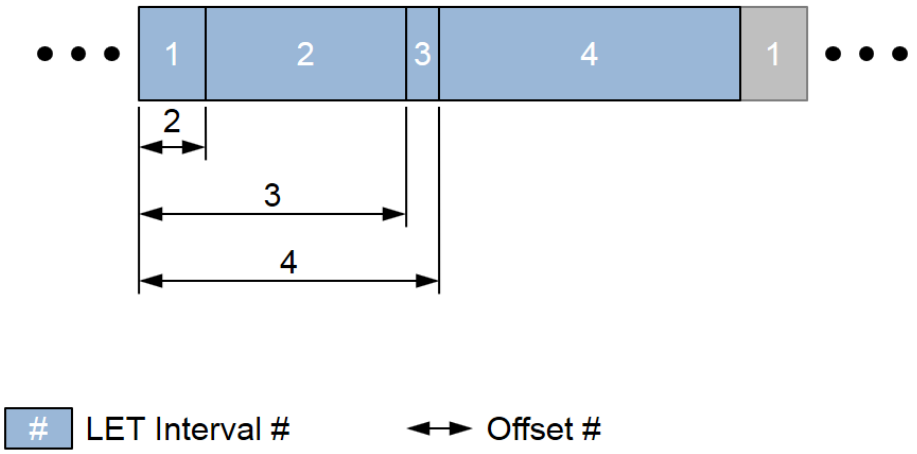


Figure 4.17: Specifying Arbitrary LET Intervals

Table 4.7 below summarizes the length of the LET intervals and the offsets between the release point of the LET interval 1 and the release points of the LET intervals 2 through 4 as shown in Figure 4.17.

LET Interval	*Length*	*Offset*
1	2ms	No offset
2	6ms	2ms
3	1ms	8ms
4	9ms	9ms

Table 4.7: Specifying Arbitrary LET Intervals

The following listing 4.10 shows an XML fragment which specifies the three offsets between the *release* point of the first LET interval 1 and the *release* points of the three LET intervals 2 through 4.

Listing 4.10: Arbitrary LET Intervals

```
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_2</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/
     SpecifyingArbitraryLetIntervals/
     sysTiming_SpecifyingArbitraryLetIntervals/
     LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/
     SpecifyingArbitraryLetIntervals/
     sysTiming_SpecifyingArbitraryLetIntervals/
     LetIntervalRelease_2</TARGET-REF>
  <MINIMUM>
```

```
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>2</CSE-CODE-FACTOR>
    </MINIMUM>
    <MAXIMUM>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>2</CSE-CODE-FACTOR>
    </MAXIMUM>
  </OFFSET-TIMING-CONSTRAINT>
  <OFFSET-TIMING-CONSTRAINT>
    <SHORT-NAME>otc_LetInterval_3</SHORT-NAME>
    <SOURCE-REF DEST="TD-EVENT-COMPLEX">/
        SpecifyingArbitraryLetIntervals/
        sysTiming_SpecifyingArbitraryLetIntervals/
        LetIntervalRelease_1</SOURCE-REF>
    <TARGET-REF DEST="TD-EVENT-COMPLEX">/
        SpecifyingArbitraryLetIntervals/
        sysTiming_SpecifyingArbitraryLetIntervals/
        LetIntervalRelease_3</TARGET-REF>
    <MINIMUM>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>8</CSE-CODE-FACTOR>
    </MINIMUM>
    <MAXIMUM>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>8</CSE-CODE-FACTOR>
    </MAXIMUM>
  </OFFSET-TIMING-CONSTRAINT>
  <OFFSET-TIMING-CONSTRAINT>
    <SHORT-NAME>otc_LetInterval_4</SHORT-NAME>
    <SOURCE-REF DEST="TD-EVENT-COMPLEX">/
        SpecifyingArbitraryLetIntervals/
        sysTiming_SpecifyingArbitraryLetIntervals/
        LetIntervalRelease_1</SOURCE-REF>
    <TARGET-REF DEST="TD-EVENT-COMPLEX">/
        SpecifyingArbitraryLetIntervals/
        sysTiming_SpecifyingArbitraryLetIntervals/
        LetIntervalRelease_4</TARGET-REF>
    <MINIMUM>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>9</CSE-CODE-FACTOR>
    </MINIMUM>
    <MAXIMUM>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>9</CSE-CODE-FACTOR>
    </MAXIMUM>
  </OFFSET-TIMING-CONSTRAINT>
```

In Figure 4.18 a similar case as described in Figure 4.17 is shown but the offset between some LET intervals and the length of some LET intervals are different which result in some gaps between LET intervals. The length of each LET interval and the offsets between the LET intervals of this case are listed in Table 4.8 along with the corresponding offsets between the *release* points of the LET intervals.

Like in the previous example the periodic event triggering constraint `PeriodicEventTriggering` is used to specify that the period of the sequence is 32ms.

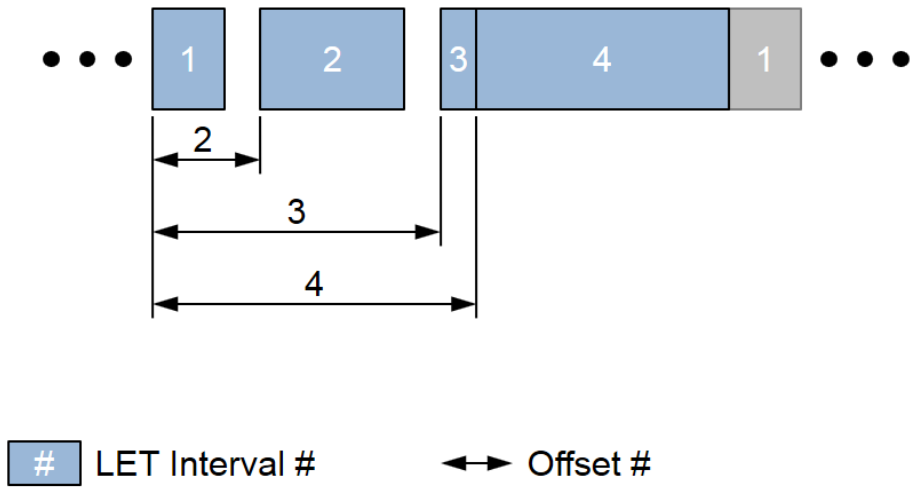


Figure 4.18: Specifying Arbitrary LET Intervals with Gaps

Table 4.8 below summarizes the length of the LET intervals and the offsets between the release point of the LET interval 1 and the release points of the LET intervals 2 through 4 as shown in Figure 4.18.

LET Interval	*Length*	*Offset*
1	2ms	No offset
2	4ms	3ms
3	1ms	8ms
4	7ms	9ms

Table 4.8: Specifying Arbitrary LET Intervals with Gaps

The XML fragment of this example is the same as shown in listing 4.10 except the values of the offset timing constraints, and the latency timing constraints, which specify the length of the four LET intervals.

Considering the development of systems consisting of several electronic control units, each of them most likely equipped with multiple processors, as well as processors containing several independent execution units, also know as *cores*, there is the necessity to be able specify LET intervals supporting the parallel execution of executable entities.

In Figure 4.19 four LET intervals are specified which overlap. The LET interval 3 and LET interval 4 are overlapping each other and overlapping LET interval 1 and LET interval 2, too.

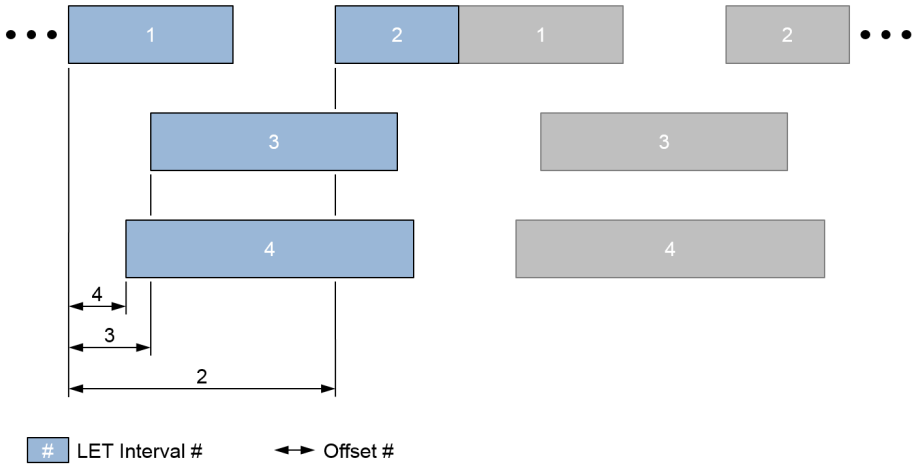


Figure 4.19: Specifying Arbitrary LET Intervals with Overlap

Table 4.9 below summarizes the length of the LET intervals and the offsets between the release point of the LET interval 1 and the release points of the LET intervals 2 through 4 as shown in Figure 4.19.

LET Interval	*Length*	*Offset*
1	8ms	No offset
2	6ms	13ms
3	12ms	4ms
4	14ms	3ms

Table 4.9: Specifying Arbitrary LET Intervals with Overlap

The periodic event triggering constraint `PeriodicEventTriggering` is used to specify that the period of the sequence is 19ms.

The following listing 4.11 shows an XML fragment which specifies three offset timing constraints between the *release* point of LET interval 1 and the three *release* points of the LET interval 2 through LET interval 4. The values of the offsets are selected such that LET interval 3 and LET interval 4 are overlapping the LET interval 1 and LET interval 2. The offset between the *release* points of LET interval 1 and LET interval 2 is 13ms; the offset between the *release* point of LET interval 1 and LET interval 3 is 4ms; and the offset between the *release* point of LET interval 1 and LET interval 4 is 3ms.

Listing 4.11: Arbitrary LET Intervals with Overlap

```
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_2</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/
      SpecifyingArbitraryLetIntervalsWithOverlap/
      sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/
      LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/
      SpecifyingArbitraryLetIntervalsWithOverlap/
      sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/
      LetIntervalRelease_2</TARGET-REF>
```

```
    <MINIMUM>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>13</CSE-CODE-FACTOR>
    </MINIMUM>
    <MAXIMUM>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>13</CSE-CODE-FACTOR>
    </MAXIMUM>
  </OFFSET-TIMING-CONSTRAINT>
  <OFFSET-TIMING-CONSTRAINT>
    <SHORT-NAME>otc_LetInterval_3</SHORT-NAME>
    <SOURCE-REF DEST="TD-EVENT-COMPLEX">/
        SpecifyingArbitraryLetIntervalsWithOverlap/
        sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/
        LetIntervalRelease_1</SOURCE-REF>
    <TARGET-REF DEST="TD-EVENT-COMPLEX">/
        SpecifyingArbitraryLetIntervalsWithOverlap/
        sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/
        LetIntervalRelease_3</TARGET-REF>
    <MINIMUM>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>4</CSE-CODE-FACTOR>
    </MINIMUM>
    <MAXIMUM>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>4</CSE-CODE-FACTOR>
    </MAXIMUM>
  </OFFSET-TIMING-CONSTRAINT>
  <OFFSET-TIMING-CONSTRAINT>
    <SHORT-NAME>otc_LetInterval_4</SHORT-NAME>
    <SOURCE-REF DEST="TD-EVENT-COMPLEX">/
        SpecifyingArbitraryLetIntervalsWithOverlap/
        sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/
        LetIntervalRelease_1</SOURCE-REF>
    <TARGET-REF DEST="TD-EVENT-COMPLEX">/
        SpecifyingArbitraryLetIntervalsWithOverlap/
        sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/
        LetIntervalRelease_4</TARGET-REF>
    <MINIMUM>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>3</CSE-CODE-FACTOR>
    </MINIMUM>
    <MAXIMUM>
      <CSE-CODE>3</CSE-CODE>
      <CSE-CODE-FACTOR>3</CSE-CODE-FACTOR>
    </MAXIMUM>
  </OFFSET-TIMING-CONSTRAINT>
```

4.6.3 Specifying Executable Entities Cluster

During the development of distributed real-time systems there is frequently the case to be capable of grouping executable entities for various purposes, like scheduling,

mapping a number of executable entities to LET intervals, or specifying other constraint that are imposed on a group of executable entities[1].

The capability of the execution order constraint ExecutionOrderConstraint to group executable entities — references to the RTE or BSW events of those executable entities — is used to specify clusters of executable entities. In particular, an execution order constraint executable entity reference group EOCExecutableEntityRefGroup (EOCEERG) is used to references all RTE and/or BSW events of the executable entities that are part of an *executable entities cluster*. And in turn an execution order constraint executable entity reference group can reference other groups of executable entities to form a hierarchy of executable entities clusters.

4.6.4 Mapping Executable Entities Cluster to LET Interval

The term *Executable Entities Cluster* is a synonym for EOCExecutableEntityRefGroup — also known as *group of executable entities* — and is explained in subsection 4.6.3. An executable entities cluster is mapped to a LET interval using the attribute letInterval.

As an example, assume there are eight atomic software components called *Swc01* through *Swc08* and each of these software components has an internal behavior called *ib_Swc0#* where the number sign indicates one of the software components 1 through 8. Each software component has one runnable entity called *re_re01_Swc0#* and a Timing Event called *te_re01_Swc0#* is specified and references the corresponding runnable entity in the software component. The runnable entities have data read and data write access points which read data from the ports of the particular software component and write data to the provided ports of the particular software component. The data dependency graph among the runnable entities is shown in Figure 4.20.

For this example three executable entities clusters are defined: The first executable entities cluster consists of the runnable entities *re_re01_Swc01*, *re_re01_Swc02* and *re_re01_Swc04*; the second one consists of the runnable entities *re_re01_Swc03* and *re_re01_Swc07*; and the third executable entities cluster consists of the runnable entities *re_re01_Swc05*, *re_re01_Swc06* and *re_re01_Swc08*. Three LET intervals are defined in the example and are called *LET interval 1*, *LET interval 2* and *LET interval 3*. The first executable entities cluster is mapped to the first LET interval using the association called letInterval between the element EOCExecutableEntityRefGroup and the element TimingDescriptionEventChain.

[1]The term ”executable entity” is used to refer to both, AUTOSAR *Runnable Entity* and *Basic Software Module Entity*. Indeed, the AUTOSAR element ExecutableEntity is an abstract class and RunnableEntity and BswModuleEntity are specializations of this class.

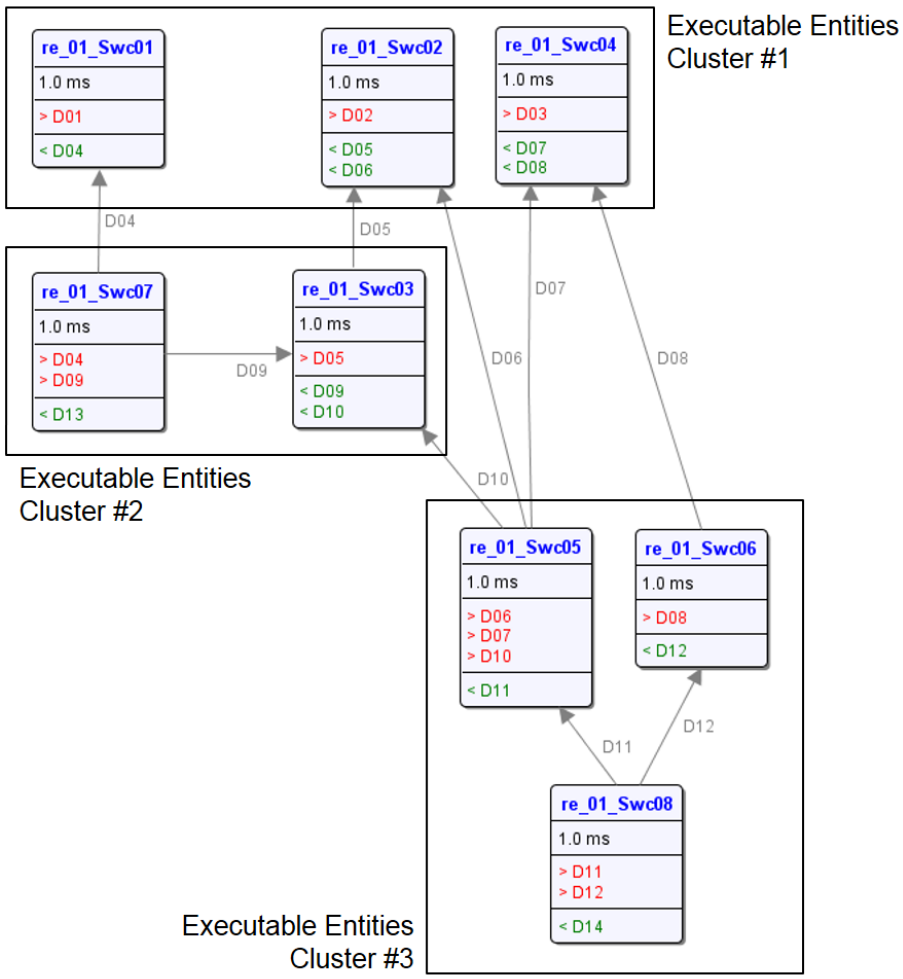


Figure 4.20: Data Dependencies among Executable Entities

The following listing 4.12 shows an XML fragment which specifies that an executable entities cluster and that it is mapped to the LET interval 1.

Listing 4.12: Mapping Executable Entities Cluster to LET Intervals

```
<EOC-EXECUTABLE-ENTITY-REF-GROUP>
  <SHORT-NAME>eocEerg_ExecutableEntitiesCluster_01</SHORT-
    NAME>
  <NESTED-ELEMENT-REFS>
    <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/
      MappingEecToLetInterval/
      sysTiming_MappingEecToLetInterval/
      eoc_DefiningAndMappingEecs/eocERef_te_re01_Swc01_1ms
      </NESTED-ELEMENT-REF>
    <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/
      MappingEecToLetInterval/
      sysTiming_MappingEecToLetInterval/
      eoc_DefiningAndMappingEecs/eocERef_te_re01_Swc02_1ms
      </NESTED-ELEMENT-REF>
    <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/
      MappingEecToLetInterval/
      sysTiming_MappingEecToLetInterval/
```

```
        eoc_DefiningAndMappingEecs/eocERef_te_re01_Swc04_1ms
        </NESTED-ELEMENT-REF>
    </NESTED-ELEMENT-REFS>
    <SUCCESSOR-REFS>
      <SUCCESSOR-REF DEST="EOC-EXECUTABLE-ENTITY-REF-GROUP">/
        MappingEecToLetInterval/
        sysTiming_MappingEecToLetInterval/
        eoc_DefiningAndMappingEecs/
        eocEerg_ExecutableEntitiesCluster_02</SUCCESSOR-REF>
      <SUCCESSOR-REF DEST="EOC-EXECUTABLE-ENTITY-REF-GROUP">/
        MappingEecToLetInterval/
        sysTiming_MappingEecToLetInterval/
        eoc_DefiningAndMappingEecs/
        eocEerg_ExecutableEntitiesCluster_03</SUCCESSOR-REF>
    </SUCCESSOR-REFS>
    <LET-INTERVAL-REFS>
      <LET-INTERVAL-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN"
        >/MappingEecToLetInterval/
        sysTiming_MappingEecToLetInterval/tdecLetInterval_1<
        /LET-INTERVAL-REF>
    </LET-INTERVAL-REFS>
  </EOC-EXECUTABLE-ENTITY-REF-GROUP>
```

An executable entities cluster is mapped to a LET interval by letting the EOCExecutableEntityRefGroup reference the timing description event chain that plays the role of a LET interval.

[TPS_TIMEX_00055] Purpose of the attribute letInterval ⌈ The association letInterval is used to map executable entities contained in a group of executable entities to a LET interval. ⌋*(RS_TIMEX_00022)*

[constr_4554] Usage of optional directed association/reference letInterval ⌈ This optional directed association/reference shall only be used in the context of Logical Execution Time when an EOCExecutableEntityRefGroup is used to specify clusters of executable entities — executable entities cluster. ⌋*()*

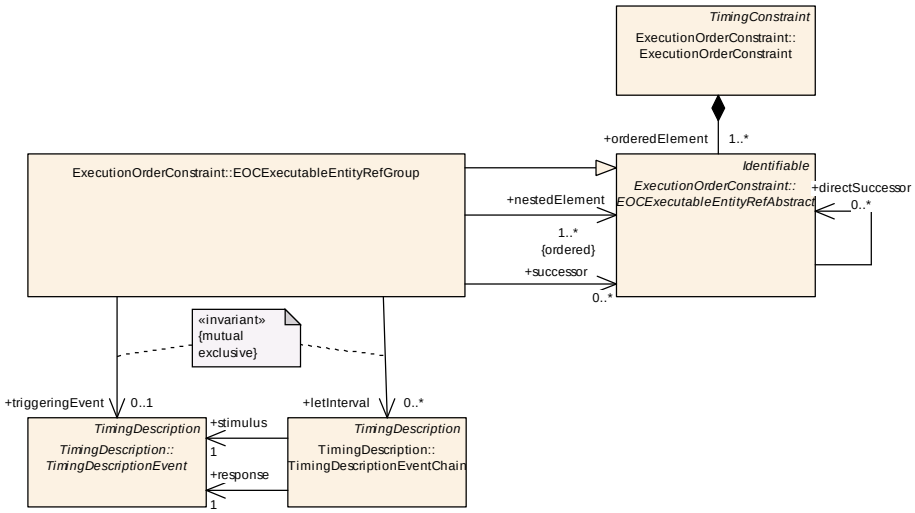


Figure 4.21: Mapping Executable Entities Cluster to LET Interval by referencing the Timing Description Event Chain by an Execution Order Constraint Executable Entity Reference Group

4.6.5 Ignoring Execution Order

In the context of logical execution time it is necessary to specify the execution order of executable entities at a later point in time than the point in time the executable entities are mapped to LET intervals. Since the purpose of the execution order constraint is to specify an execution order among a number of given executable entities, it is required to indicate that the executable entities referenced within an execution order constraint are not ordered yet or the order is not relevant. Indeed, later in the development of a system the execution order becomes relevant.

For this purpose the *optional* attribute `ignoreOrderAllowed` is added to the `ExecutionOrderConstraint` to indicate that the given order shall/could be ignored. The type of this attribute is boolean: The value FALSE indicates to take the specified order into account; and the value TRUE indicates to ignore the given order. Please note that the execution order is either specified by the successor relationships or the fact that the list `nestedElement` is an *ordered* list of elements.

[constr_4553] Usage of optional attribute `ignoreOrderAllowed` ⌈ This optional attribute shall only be used in the context of Logical Execution Time when an `EOCExecutableEntityRefGroup` is used to specify clusters of executable entities — executable entities cluster. ⌋*()*

4.6.6 Categories specified for LET

The attribute `category` of a `TimingDescriptionEventChain` and `TimingDescriptionEvent` indicates the use of these elements in the context of Logical Execution Time.

The value of the attribute `category` of a `TimingDescriptionEventChain` indicates that the timing description event chain plays the role of a LET interval. The value `LET_INTERVAL` specifies that the timing description event chain plays the role of a LET interval. For more details refer to section 6 and in particular constraints [constr_4560] and [constr_4557].

The value of the attribute `category` of a `TimingDescriptionEvent` indicates if the timing description event represents the release point of a LET interval or the terminate point of a LET interval. The value `LET_RELEASE` indicates the point in time the LET interval is released respectively started. And the value `LET_TERMINATE` indicates the point in time the LET interval is terminated respectively stopped. For more details refer to section 5 and in particular constraints [constr_4555], [constr_4556] and [constr_4559].

5 Timing Description Events

[TPS_TIMEX_00001] Purpose of `TimingDescriptionEvent` ⌈ The element `TimingDescriptionEvent` and its specializations are used to describe the occurrences of an event which are observed at a specific location in a system during runtime respectively the operation of the system. ⌋*(RS_TIMEX_00001)*

For example, this can be the start of a `RunnableEntity` or storing a frame in the hardware buffer of a communication controller.

An overview of the different event types is given in Figure 5.1. These are described in more detail in the following.

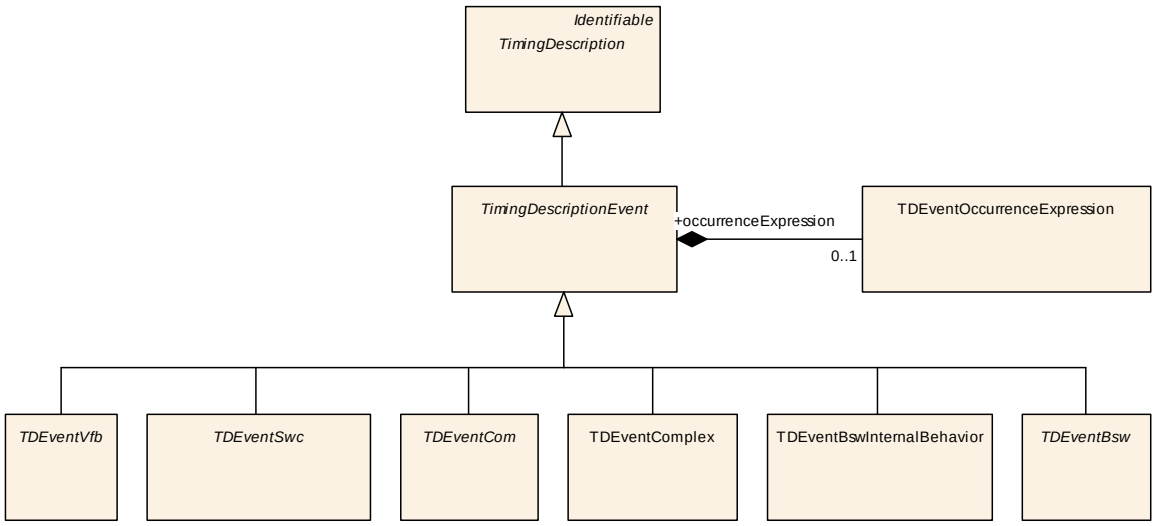


Figure 5.1: Overview of the different types of timing events

Class	***TimingDescriptionEvent*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription			
Note	A timing event is the abstract representation of a specific system behavior – that can be observed at runtime – in the AUTOSAR specification. Timing events are used to define the scope for timing constraints. Depending on the specific scope, the view on the system, and the level of abstraction different types of events are defined. In order to avoid confusion with existing event descriptions in the AUTOSAR templates the timing specific event types use the prefix TD.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*			
Subclasses	*TDEventBsw*, TDEventBswInternalBehavior, *TDEventCom*, TDEventComplex, *TDEventSwc*, *TDEventVfb*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
occurrence Expression	TDEventOccurrence Expression	0..1	aggr	The occurrence expression for this event.

Table 5.1: TimingDescriptionEvent

Table 5.2 lists all allowed values of the attribute `category` of the `TimingDescriptionEvent` and describes the corresponding semantics.

Category	*Description*
empty	No specific semantics is imposed on the `TimingDescriptionEvent`. It indicates the standard behavior.
`STANDARD`	No specific semantics is imposed on the `TimingDescriptionEvent`. It indicates the standard behavior.
`LET_RELEASE`	The `TimingDescriptionEvent` marks the *release point* of a LET interval.
`LET_TERMINATE`	The `TimingDescriptionEvent` marks the *terminate point* of a LET interval.

Table 5.2: Categories of `TimingDescriptionEvent`

The value of the attribute `category` of a `TimingDescriptionEvent` indicates if the timing description event represents the release point of a LET interval or the terminate point of a LET interval. The value `LET_RELEASE` indicates the point in time the LET interval is released respectively started. And the value `LET_TERMINATE` indicates the point in time the LET interval is terminated respectively stopped.

[TPS_TIMEX_00056] Attribute `category` used in context of `TimingDescriptionEvent` and LET ⌈ The attribute `category` is used to indicate that the timing description event represents the release point of a LET interval or the terminate point of a LET interval. ⌋*(RS_TIMEX_00022)*

[constr_4559] `category` of `TimingDescriptionEvent` shall not be extended ⌈ In contrast to the general rule that `category` can be extended by user-specific values it is not allowed to extend the meaning of the attribute `category` of meta-class `TimingDescriptionEvent`. ⌋*()*

[constr_4555] Usage of the category value `LET_RELEASE` in `TimingDescriptionEvent` ⌈ The value `LET_RELEASE` of the attribute `category` of a `TimingDescriptionEvent` shall be set if and only if the timing description event plays the role

of a stimulus event and the corresponding timing description event chain, referencing this timing description event, plays the role of a LET interval. ⌋()

[constr_4556] Usage of the category value `LET_TERMINATE` in `TimingDescriptionEvent` ⌈ The value `LET_TERMINATE` of the attribute `category` of a `TimingDescriptionEvent` shall be set if and only if the timing description event plays the role of a response event and the corresponding timing description event chain, referencing this timing description event, plays the role of a LET interval. ⌋()

Also note that information regarding the occurrence of a `TimingDescriptionEvent` is described separately in section 7.1.

5.1 Timing Events Related to the VFB

[TPS_TIMEX_00016] Purpose of `TDEventVfb` ⌈ The element `TDEventVfb` and its specializations are used to describe the occurrences of an event which are observed at a specific location in the VFB view. ⌋(*RS_TIMEX_00001*)

Events related to the VFB can be used during the specification of:

- `VfbTiming` 3.2
- `SwcTiming` 3.3
- `SystemTiming` 3.4
- `EcuTiming` 3.6

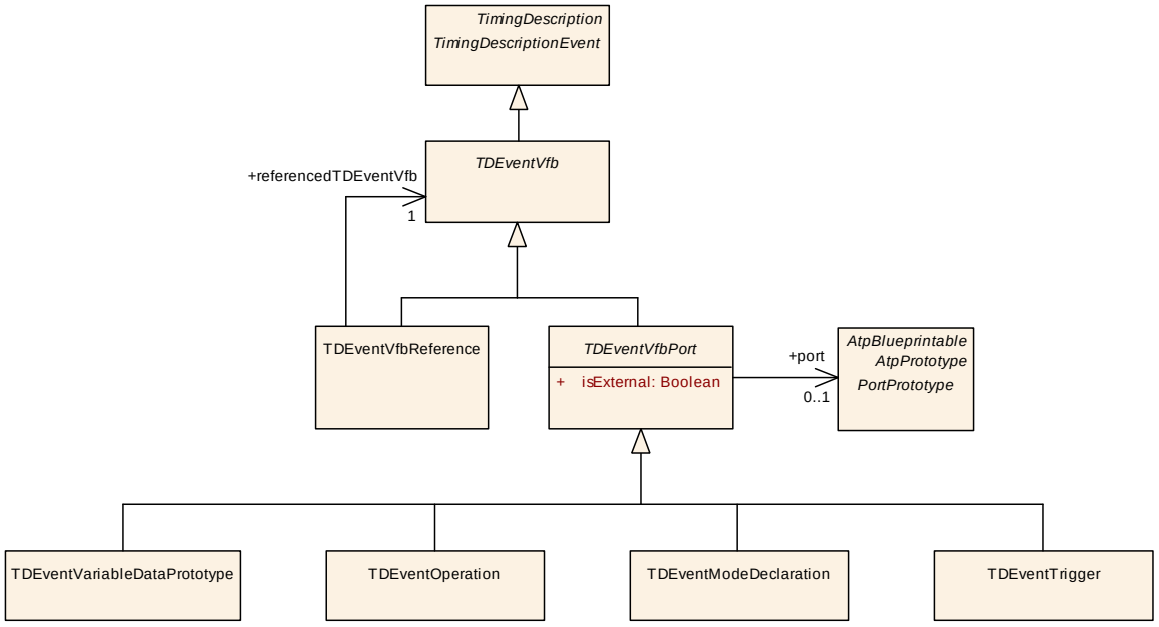


Figure 5.2: VFB events

Class	***TDEventVfb*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	This is the abstract parent class to describe timing events at Virtual Function Bus (VFB) level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	*TDEventVfbPort*, TDEventVfbReference			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
component	SwComponent Prototype	0..1	iref	The context for the scope of this timing event.

Table 5.3: TDEventVfb

[TPS_TIMEX_00042] Purpose of `TDEventVfbPort` ⌈ The element `TDEventVfbPort` and its specializations are used to describe the occurrences of an event which are observed at a specific location in the VFB view. ⌋*(RS_TIMEX_00001, RS_TIMEX_00019)*

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

Table 5.4: TDEventVfbPort

In order to support the description of timing events for hardware I/O already at VFB-level (e.g. in order to refer to the point in time where data is generated by a physical sensor) without having the need to specify the concrete sensor hardware, it is necessary to specify the attribute `isExternal`.

If for a timing event of type `TDEventVfbPort` the attribute is set to ”TRUE”, then the timing event refers to the point in time where the data is generated/processed by the corresponding hardware I/O.

If the attribute is set to ”FALSE”, then the timing event refers to the point in time where the data enters or leaves the respective port of the component at VFB-level.

[TPS_TIMEX_00043] Purpose of `TDEventVfbReference` ⌈ The element `TDEventVfbReference` is used to reference timing description events already specified in other timing views. In other words, it enables one to re-use existing timing models. ⌋ *(RS_TIMEX_00001, RS_TIMEX_00019)*

Class	**TDEventVfbReference**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	This is used to reference timing description events related to the Virtual Function Bus (VFB) view which are specified in other timing views.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TimingDescription*, *Timing DescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
referenced TDEventVfb	TDEventVfb	1	ref	The referenced timing description event.

Table 5.5: TDEventVfbReference

[TPS_TIMEX_00017] `TDEventVariableDataPrototype` specifies events observable at sender/receiver ports ⌈ The element `TDEventVariableDataPrototype` is used to specify events, namely the receipt and sending of veriable data prototypes, observable at required and provided sender/receiver ports. ⌋ *(RS_TIMEX_00001)*

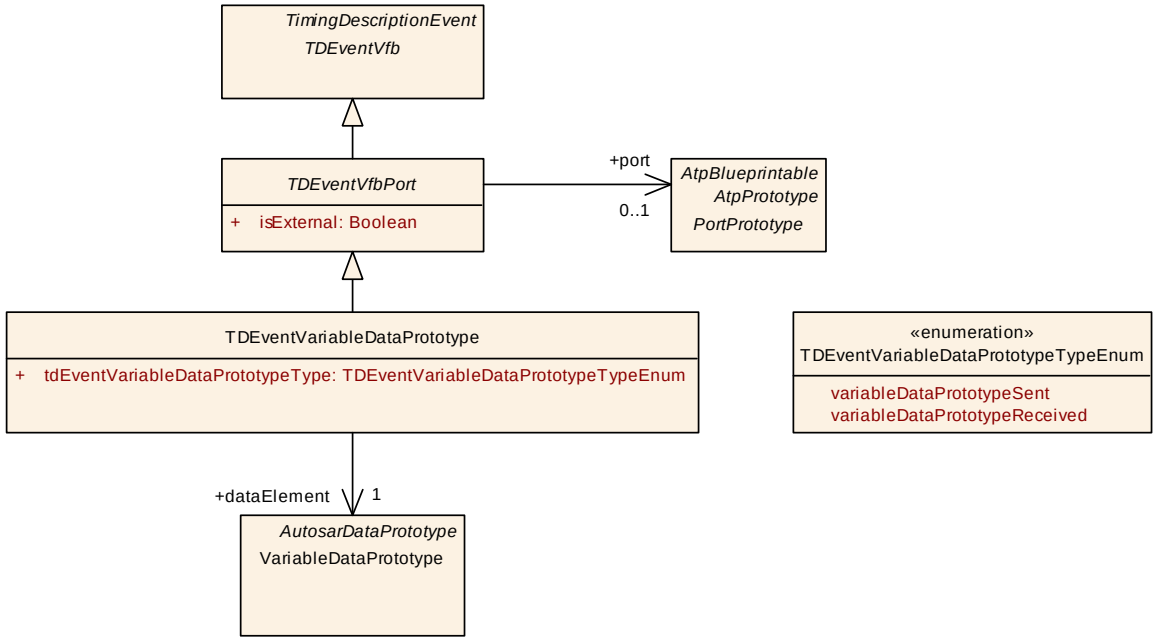


Figure 5.3: Variable Data Prototpye

Class	**TDEventVariableDataPrototype**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb::VariableDataPrototype			
Note	This is used to describe timing events related to sender-receiver communication at VFB level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TDEventVfbPort*, *Timing Description*, *TimingDescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

▽

△

Class	**TDEventVariableDataPrototype**			
dataElement	VariableDataPrototype	1	ref	The referenced VariableDataPrototype
tdEventVariable DataPrototype Type	TDEventVariableData PrototypeTypeEnum	1	attr	The specific type of this timing event.

Table 5.6: TDEventVariableDataPrototype

Enumeration	**TDEventVariableDataPrototypeTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb::VariableDataPrototype
Note	This is used to describe the specific event type of a TDEventVariableDataPrototype
Literal	***Description***
variableData PrototypeReceived	A point in time where the referenced variable data prototype has been successfully transmitted and is available in the related communication buffer (of the RTE) for the receiving SWC. **Tags:** atp.EnumerationValue=0
variableData PrototypeSent	A point in time where the referenced variable data prototype has been successfully sent out by the sending SWC, so that it is available in the related communication buffer (of the RTE) for transmission. **Tags:** atp.EnumerationValue=1

Table 5.7: TDEventVariableDataPrototypeTypeEnum

[TPS_TIMEX_00018] `TDEventOperation` specifies events obeservable at client/server ports. ⌈ The element `TDEventOperation` is used to specify events, namely the invokation of operations and their completion, observable at required and provided client/server ports. ⌋*(RS_TIMEX_00001)*

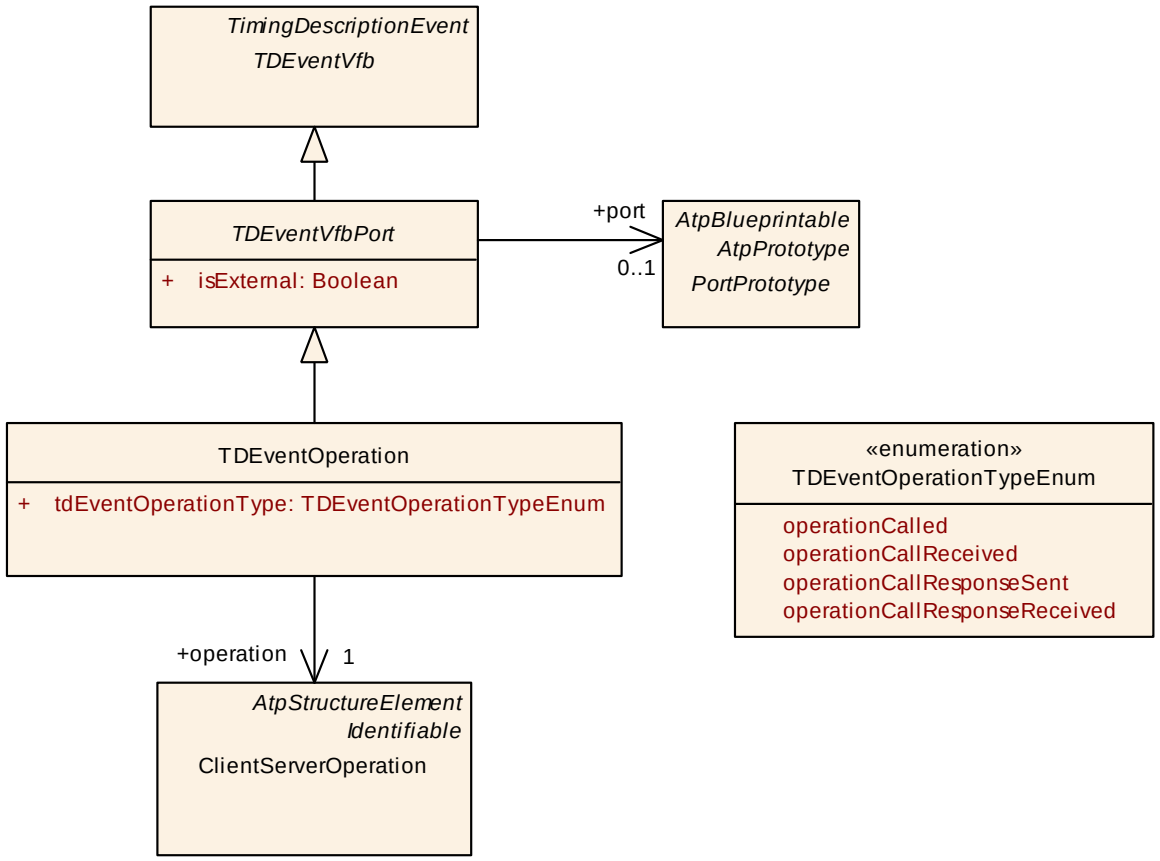


Figure 5.4: Operation

Class	**TDEventOperation**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb::Operation			
Note	This is used to describe timing events related to client-server communication at VFB level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TDEventVfbPort*, *Timing Description*, *TimingDescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
operation	ClientServerOperation	1	ref	The referenced operation.
tdEvent OperationType	TDEventOperationType Enum	1	attr	The specific type of this timing event.

Table 5.8: TDEventOperation

Enumeration	**TDEventOperationTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb::Operation
Note	This is used to describe the specific event type of a TDEventOperation.
Literal	***Description***
operationCall Received	A point in time where the call of the referenced operation is received by the server SWC. **Tags:** atp.EnumerationValue=1

△

Enumeration	**TDEventOperationTypeEnum**
operationCall ResponseReceived	A point in time where the client SWC has received the response of the referenced operation call. **Tags:** atp.EnumerationValue=2
operationCall ResponseSent	A point in time where the server SWC has terminated with the execution of the referenced operation, and has sent out a response. **Tags:** atp.EnumerationValue=3
operationCalled	A point in time where the referenced operation is called by the client SWC. **Tags:** atp.EnumerationValue=0

Table 5.9: TDEventOperationTypeEnum

[TPS_TIMEX_00019] `TDEventModeDeclaration` specifies events obeservable at mode ports. ⌈ The element `TDEventModeDeclaration` is used to specify events, namely initiation and propagation of mode changes, observable at required and provided mode ports. ⌋*(RS_TIMEX_00001)*

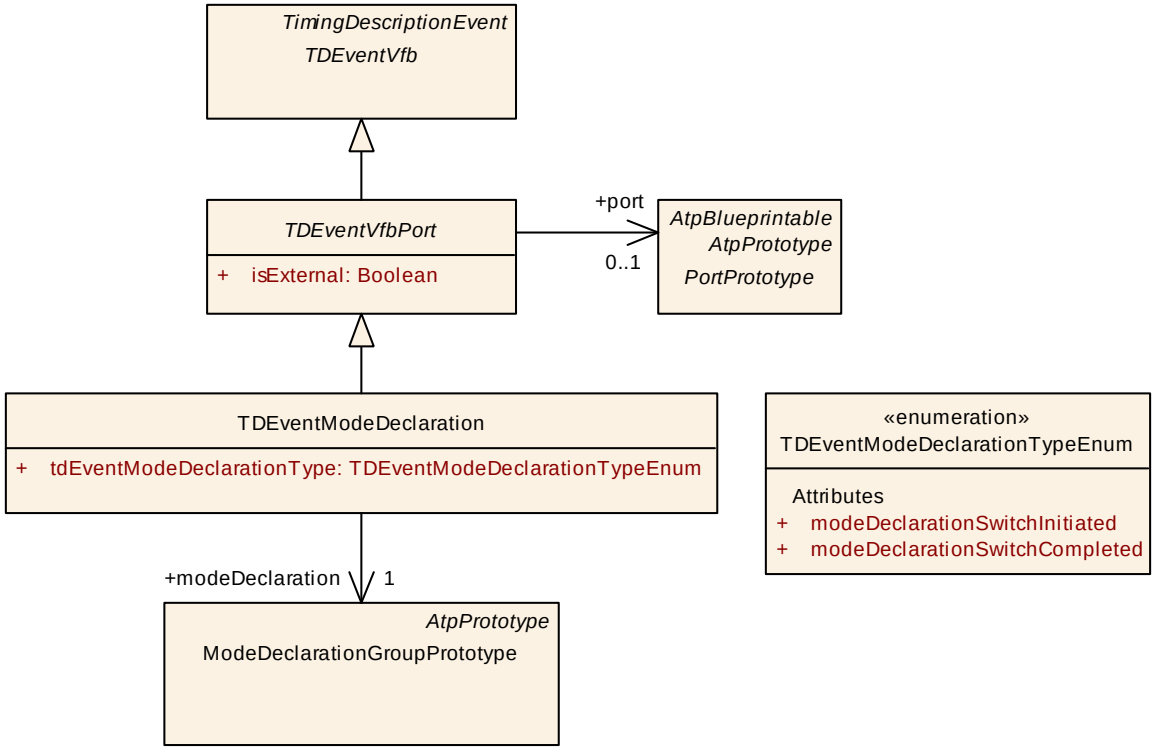


Figure 5.5: Mode Declaration

Class	**TDEventModeDeclaration**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb::ModeDeclaration
Note	This is used to describe timing events related to mode switch communication at VFB level.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TDEventVfbPort*, *Timing Description*, *TimingDescriptionEvent*

▽

△

Class	**TDEventModeDeclaration**			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
entryMode Declaration	ModeDeclaration	0..1	ref	Optional parameter which refines the scope of the TDEventModeDeclaration. If the parameter is set, the event occurs only if the mode declaration group prototype instance shall enter into the referenced ModeDeclaration.
exitMode Declaration	ModeDeclaration	0..1	ref	Optional parameter which refines the scope of the TDEventModeDeclaration. If the parameter is set, the event occurs only if the mode declaration group prototype instance shall exit from the referenced ModeDeclaration.
mode Declaration	ModeDeclarationGroup Prototype	1	ref	The referenced mode declaration group prototype.
tdEventMode DeclarationType	TDEventMode DeclarationTypeEnum	1	attr	The specific type of this timing event.

Table 5.10: TDEventModeDeclaration

Enumeration	**TDEventModeDeclarationTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb::ModeDeclaration
Note	This is used to describe the specific event type of a TDEventModeDeclaration
Literal	***Description***
modeDeclaration SwitchCompleted	A point in time where the switch to the associated ModeDeclarationGroupPrototype has been completed. **Tags:** atp.EnumerationValue=0
modeDeclaration SwitchInitiated	A point in time where the switch to the associated ModeDeclarationGroupPrototype has been initiated. **Tags:** atp.EnumerationValue=1

Table 5.11: TDEventModeDeclarationTypeEnum

[TPS_TIMEX_00039] `TDEventTrigger` specifies events observable at trigger ports ⌈ The element `TDEventTrigger` is used to specify events, namely the activation and release of triggers, observable at required and provided trigger ports. ⌋ *(RS_TIMEX_00001, RS_TIMEX_00018)*

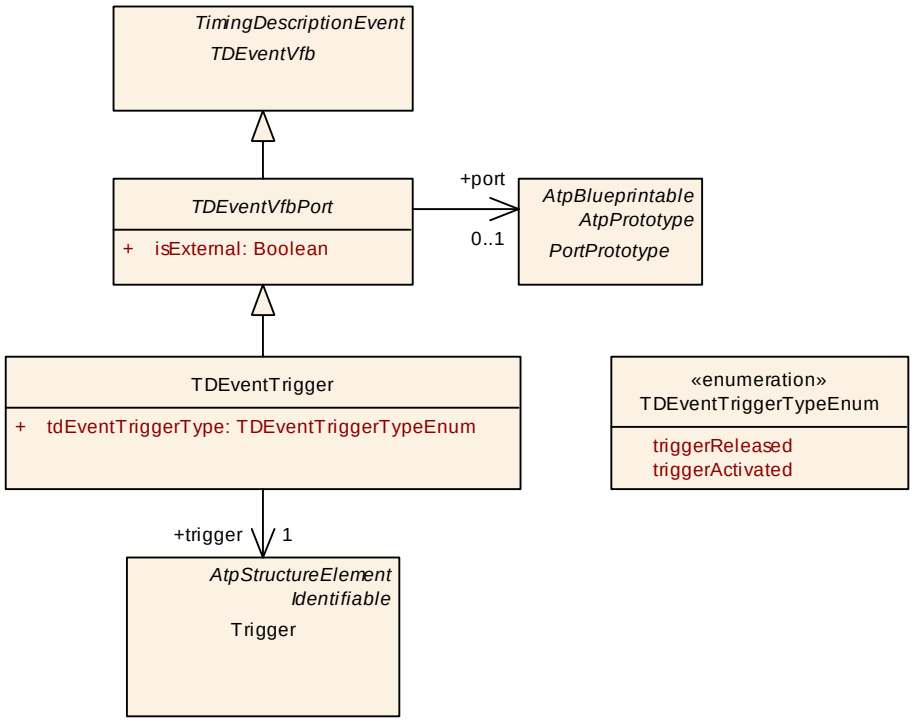


Figure 5.6: Trigger

Class	**TDEventTrigger**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb::Trigger			
Note	This is used to describe timing events related to triggers at VFB level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TDEventVfbPort*, *Timing Description*, *TimingDescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
tdEventTrigger Type	TDEventTriggerType Enum	1	attr	The specific type of this timing event.
trigger	Trigger	1	ref	The trigger which is provided (released) or required (activate) in the given context.

Table 5.12: TDEventTrigger

Enumeration	**TDEventTriggerTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb::Trigger
Note	This is used to describe the specific event type of a TDEventTrigger.
Literal	***Description***
triggerActivated	A point in time where the referenced trigger has been successfully released and is activating runnable entities of the receiving SW-C. **Tags:** atp.EnumerationValue=0
triggerReleased	A point in time where the referenced trigger has been successfully released by the emitting SW-C. **Tags:** atp.EnumerationValue=1

Table 5.13: TDEventTriggerTypeEnum

5.2 Timing Events Related to SwcInternalBehavior

[TPS_TIMEX_00044] Purpose of `TDEventSwc` ⌈ The element `TDEventSwc` is used to specify events, namely the activation, start, termination of runnable entities, as well as variable accesses, which are observable in the Software Component view. ⌋ *(RS_TIMEX_00001, RS_TIMEX_00019, RS_TIMEX_00020)*

Class	***TDEventSwc*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventSwcInternalBehavior			
Note	This is the abstract parent class to describe timing events at Software Component (SW-C) level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	TDEventSwcInternalBehavior, TDEventSwcInternalBehaviorReference			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
component	SwComponent Prototype	0..1	iref	The context for the scope of this timing event.

Table 5.14: TDEventSwc

[TPS_TIMEX_00020] `TDEventSwcInternalBehavior` specifies observable events of runnable entities ⌈ The element `TDEventSwcInternalBehavior` is used to specify events, namely the activation, start, termination of runnable entities, as well as variable accesses, which are observable in the Software Component view. ⌋ *(RS_TIMEX_00001, RS_TIMEX_00019, RS_TIMEX_00020)*

Events related to `SwcInternalBehavior` can be used during the specification of:

- `SwcTiming` 3.3
- `SystemTiming` 3.4
- `EcuTiming` 3.6

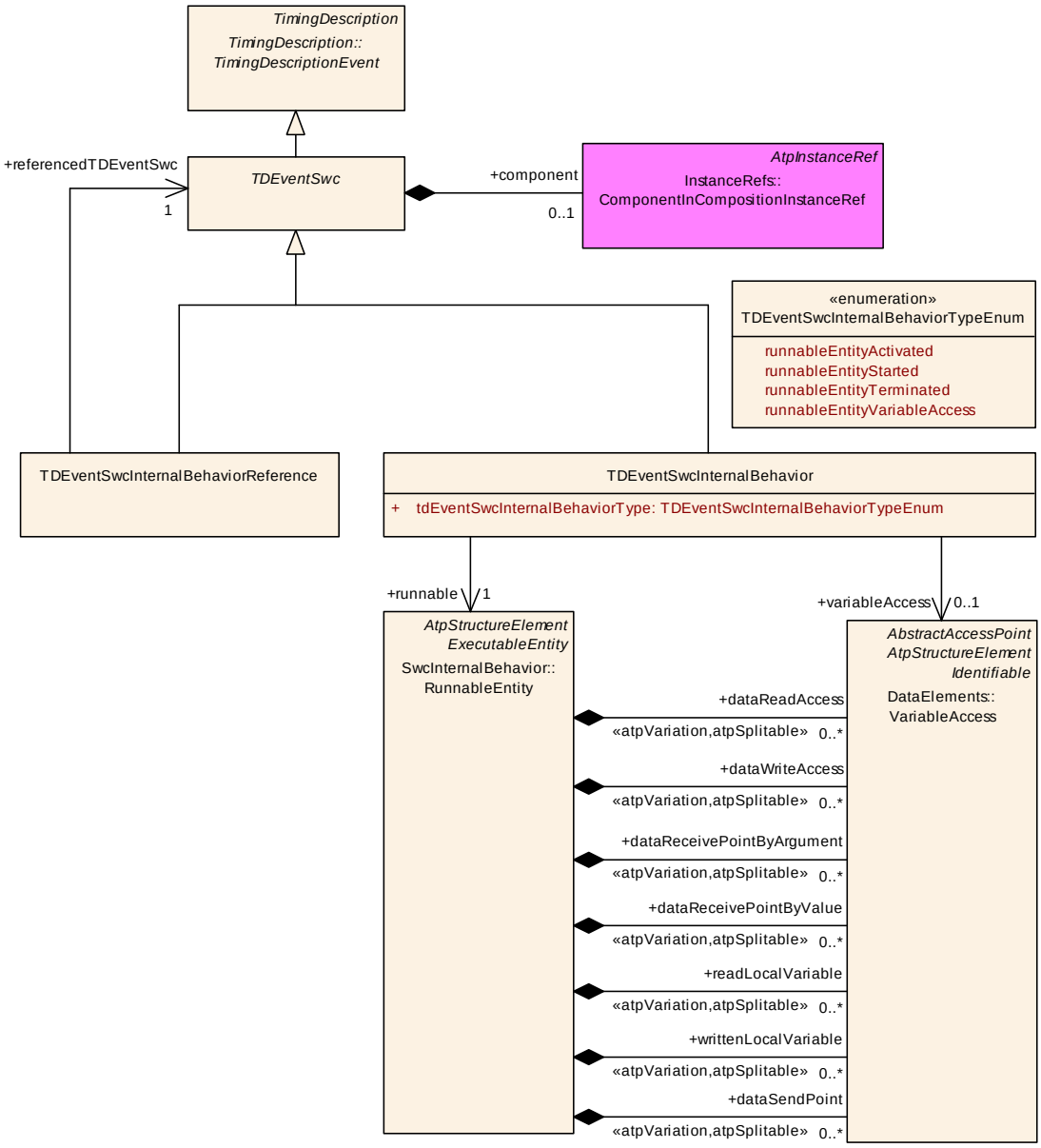


Figure 5.7: Event of type ”SwcInternalBehavior”

Class	**TDEventSwcInternalBehavior**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventSwcInternalBehavior			
Note	This is used to describe timing events related to the SwcInternalBehavior of an AtomicSwComponent Type.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventSwc*, *TimingDescription*, *Timing DescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
runnable	RunnableEntity	1	ref	The scope of this timing event.
tdEventSwc Internal BehaviorType	TDEventSwcInternal BehaviorTypeEnum	1	attr	The specific type of this timing event.

△

Class	**TDEventSwcInternalBehavior**			
variableAccess	VariableAccess	0..1	ref	The scope of this timing event.

Table 5.15: TDEventSwcInternalBehavior

Enumeration	**TDEventSwcInternalBehaviorTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventSwcInternalBehavior
Note	This is used to describe the specific event type of a TDEventSwcInternalBehavior.
Literal	*Description*
runnableEntity Activated	A point in time where the associated RunnableEntity has been activated, which means that it has entered the state "to be started". **Tags:** atp.EnumerationValue=0
runnableEntity Started	A point in time where the associated RunnableEntity has entered the state "started" after its activation. **Tags:** atp.EnumerationValue=1
runnableEntity Terminated	A point in time where the associated RunnableEntity has terminated and entered the state "suspended". **Tags:** atp.EnumerationValue=2
runnableEntity VariableAccess	A point in time where the associated variable is accessed. **Tags:** atp.EnumerationValue=3

Table 5.16: TDEventSwcInternalBehaviorTypeEnum

[constr_4510] Specifying references to `RunnableEntity` and `VariableAccess` ⌈ A `RunnableEntity` and `VariableAccess` shall be referenced at the same time if and only if the value of `tdEventSwcInternalBehaviorType` is "runnableEntity-VariableAccess". These two references are not mutual exclusive. ⌋*()*

[constr_4511] Validity of referencing `RunnableEntity` ⌈ A `RunnableEntity` shall be referenced if and only if the value of `tdEventSwcInternalBehaviorType` is "runnableEntityActivated", "runnableEntityStarted", "runnableEntityTerminated", or "runnableEntityVariableAccess". ⌋*()*

[constr_4512] Validity of referencing `VariableAccess` ⌈ A `VariableAccess` shall be referenced if and only if the value of `tdEventSwcInternalBehaviorType` is "runnableEntityVariableAccess". ⌋*()*

[TPS_TIMEX_00045] Purpose of `TDEventSwcInternalBehaviorReference` ⌈ The element `TDEventSwcInternalBehaviorReference` is used to reference timing description events already specified in other timing views. In other words, it enables one to re-use existing timing models. ⌋*(RS_TIMEX_00001, RS_TIMEX_00019)*

Class	**TDEventSwcInternalBehaviorReference**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventSwcInternalBehavior			
Note	This is used to reference timing description events related to the Software Component (SW-C) view which are specified in other timing views.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventSwc*, *TimingDescription*, *TimingDescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
referenced TDEventSwc	TDEventSwc	1	ref	The referenced timing description event.

Table 5.17: TDEventSwcInternalBehaviorReference

5.3 Timing Events Related to Bus Communication

[TPS_TIMEX_00021] Purpose of `TDEventCom` ⌈ The element `TDEventCom` and its specializations are used to describe the occurrences of an event which are observed at a specific location in the System view, in particular any event related to communications. ⌋*(RS_TIMEX_00001)*

Events related to communication can be used during the specification of:

- `SystemTiming` 3.3
- `EcuTiming` 3.6

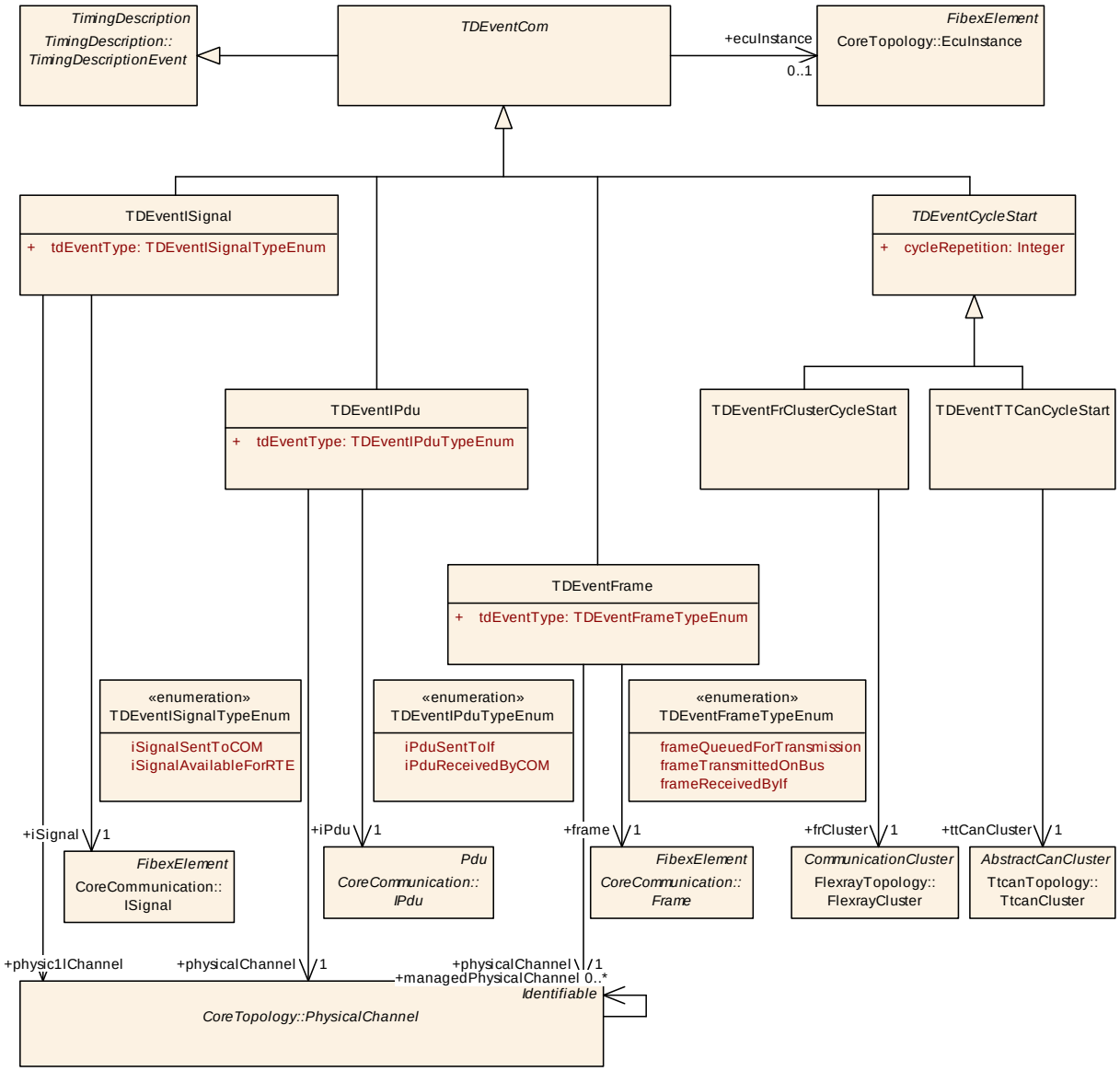


Figure 5.8: Events regarding communication

Class	***TDEventCom*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is the abstract parent class to describe timing events related to communication including the physical layer.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	*TDEventCycleStart*, TDEventFrame, TDEventFrameEthernet, TDEventIPdu, TDEventISignal			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
ecuInstance	EcuInstance	0..1	ref	The ECU context for a particular timing event. The link is optional, because the EcuInstance can not be defined for events of type TDEventCycleStart.

Table 5.18: TDEventCom

[TPS_TIMEX_00022] `TDEventISignal` specifies events related to the exchange of I-Signals ⌈ The element `TDEventISignal` is used to specify events, namely the exchange of I-Signals, observable between the RTE and the AUTOSAR Com. ⌋ *(RS_TIMEX_00001)*

Class	**TDEventISignal**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is used to describe timing events related to the exchange of I-Signals between COM and RTE.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TimingDescription*, *Timing DescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
iSignal	ISignal	1	ref	The scope of this timing event.
physical Channel	PhysicalChannel	1	ref	The PhysicalChannel on which the ISignal is transmitted.
tdEventType	TDEventISignalType Enum	1	attr	The specific type of this timing event.

Table 5.19: TDEventISignal

Enumeration	**TDEventISignalTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom
Note	This is used to describe the specific event type of a TDEventISignal.
Literal	***Description***
iSignalAvailableFor RTE	A point in time, where the COM module makes the contained signal / signal group available for the RTE and the corresponding Rx Indication callout is generated (if configured). **Tags:** atp.EnumerationValue=0
iSignalSentToCOM	A point in time, where a transmission request call is issued by the RTE on a named COM signal / signal group and the new value is stored to the carrier COM I-PDU buffer. **Tags:** atp.EnumerationValue=1

Table 5.20: TDEventISignalTypeEnum

[TPS_TIMEX_00023] `TDEventIPdu` specifies events related to the exchange of I-PDUs ⌈ The element `TDEventIPdu` is used to specify events, namely the exchange of I-PDUs, observable between the bus specific BSW modules (CANbus, FlexRay, LIN) and the AUTOSAR Com. ⌋*(RS_TIMEX_00001)*

Class	**TDEventIPdu**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is used to describe timing events related to the exchange of I-PDUs between the bus specific (Flex Ray / CAN / LIN) Interface BSW module and COM.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TimingDescription*, *Timing DescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
iPdu	IPdu	1	ref	The scope of this timing event.

▽

△

Class	**TDEventIPdu**			
physical Channel	PhysicalChannel	1	ref	The PhysicalChannel on which the IPdu is transmitted.
tdEventType	TDEventIPduTypeEnum	1	attr	The specific type of this timing event.

Table 5.21: TDEventIPdu

Enumeration	**TDEventIPduTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom
Note	This is used to describe the specific event type of a TDEventIPdu.
Literal	*Description*
iPduReceivedBy COM	A point in time where the received frame is processed by the corresponding (FlexRay / CAN / LIN) Interface BSW module, routed through the PDUR and the contained PDUs are pushed to the COM module. **Tags:** atp.EnumerationValue=0
iPduSentToIf	A point in time where the carrier COM I-PDU is routed through the PDUR and is pushed to the bus specific (FlexRay / CAN / LIN) Interface BSW module. **Tags:** atp.EnumerationValue=1

Table 5.22: TDEventIPduTypeEnum

[TPS_TIMEX_00024] `TDEventFrame` specifies events related to the exchange of network frames ⌈ The element `TDEventFrame` is used to specify events, namely the exchange of Frames, observable between the communication controller and the bus specific BSW modules (CANbus, FlexRay, LIN) and observable at the physical layer. ⌋ *(RS_TIMEX_00001)*

Class	**TDEventFrame**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is used to describe timing events related to the exchange of frames between the communication controller and the bus specific (FlexRay / CAN / LIN) Interface BSW module.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TimingDescription*, *TimingDescriptionEvent*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
frame	Frame	1	ref	The scope of this timing event.
physical Channel	PhysicalChannel	1	ref	The PhysicalChannel on which the Frame is transmitted.
tdEventType	TDEventFrameType Enum	1	attr	The specific type of this timing event.

Table 5.23: TDEventFrame

Enumeration	**TDEventFrameTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom
Note	This is used to describe the specific event type of a TDEventFrame.
Literal	***Description***
frameQueuedFor Transmission	A point in time where the frame containing the named signal / I-PDU is queued for transmission within the related Communication Driver. **Tags:** atp.EnumerationValue=0
frameReceivedByIf	A point in time where the frame is pushed from the subscriber's communication controller to the corresponding (FlexRay / CAN / LIN) Interface BSW module. **Tags:** atp.EnumerationValue=1
frameTransmitted OnBus	A point in time where the transmission of the frame completes successfully, and the subscriber's communication controller receives the frame from the bus. **Tags:** atp.EnumerationValue=2

Table 5.24: TDEventFrameTypeEnum

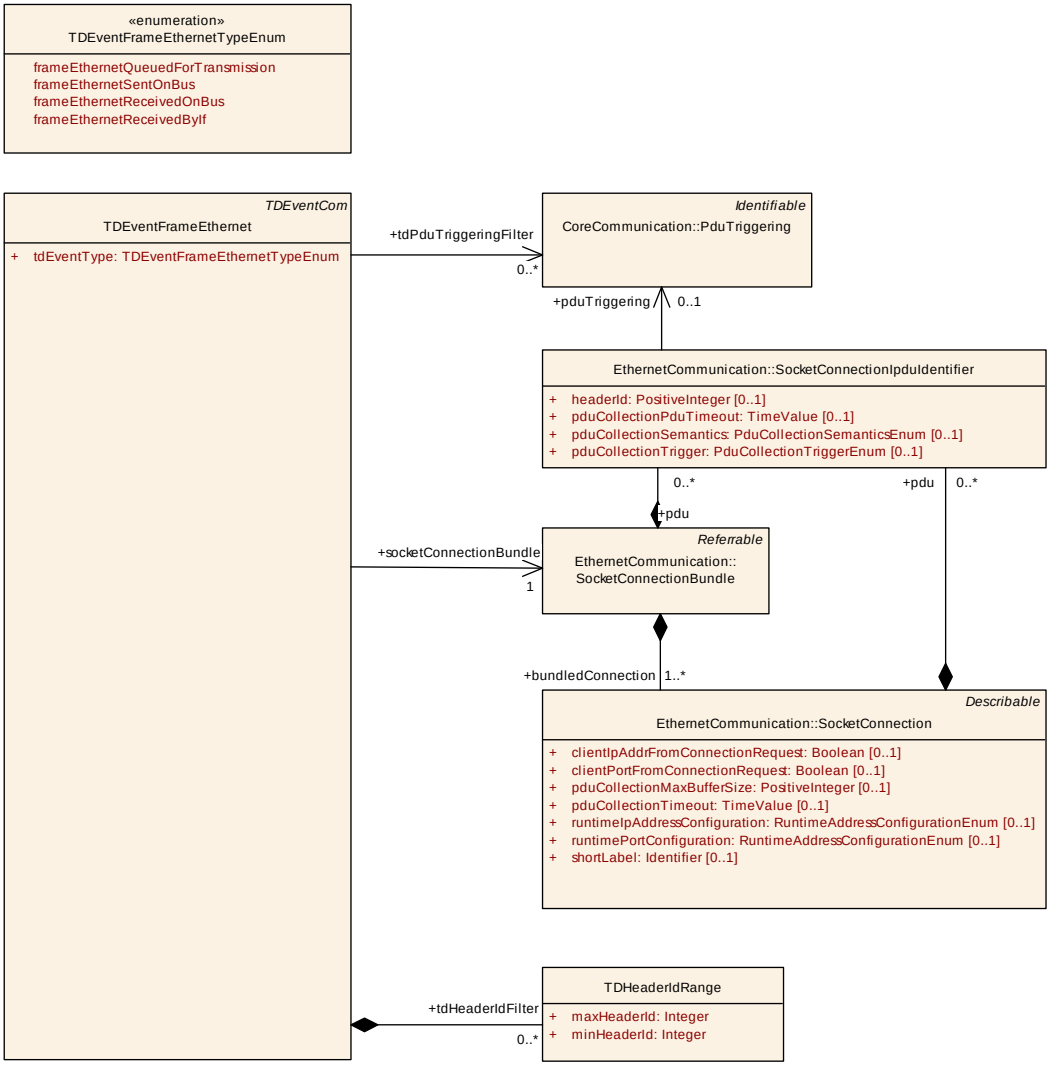


Figure 5.9: Events regarding Ethernet communication

[TPS_TIMEX_00052] `TDEventFrameEthernet` specifies events related to the exchange of Ethernet frames ⌈ The element `TDEventFrameEthernet` is used to specify events, namely the exchange of Ethernet frames, observable between the Ethernet communication controller and the Ethernet specific BSW modules, as well as observable at the physical layer. ⌋*(RS_TIMEX_00001)*

Class	**TDEventFrameEthernet**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is used to describe timing description events related to the exchange of Ethernet frames between an Ethernet communication controller and the BSW Ethernet interface and driver module.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TimingDescription*, *TimingDescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
socket Connection Bundle	SocketConnection Bundle	1	ref	Specifies the SocketConnectionBundle by the means of which the PDUs are transmitted or received within an Ethernet Frame.
tdEventType	TDEventFrameEthernet TypeEnum	1	attr	This is used to describe the specific event type of a TDEventFrameEthernet.
tdHeaderIdFilter	TDHeaderIdRange	*	aggr	Specifies the header identifier or a range of header identifiers that if contained in the Ethernet frame let the TDEventFrameEthernet occur.
tdPduTriggering Filter	PduTriggering	*	ref	Specifies the PDU that if contained in the Ethernet frame let the TDEventFrameEthernet occur.

Table 5.25: TDEventFrameEthernet

Enumeration	**TDEventFrameEthernetTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom
Note	This is used to describe the specific event type of a TDEventFrameEthernet.
Literal	***Description***
frameEthernet QueuedFor Transmission	A point in time where the Ethernet frame containing the specified PDUs is queued for transmission within the corresponding Ethernet Communication Driver. **Tags:** atp.EnumerationValue=0
frameEthernet ReceivedByIf	A point in time where the frame is pushed from the corresponding Ethernet communication controller to the BSW Ethernet communication interface. **Tags:** atp.EnumerationValue=1
frameEthernet ReceivedOnBus	A point in time where the receipt of the Ethernet frame/packet completes successfully on the recipient's Ethernet communication controller. In other words, the Ethernet frame/packet has entered the recipient's Ethernet communication controller which means the last bit of the Ethernet frame/packet has been received. **Tags:** atp.EnumerationValue=2
frameEthernetSent OnBus	A point in time where the transmission of the Ethernet frame/packet completes successfully on the physical Ethernet communication network. In other words, the Ethernet frame/packet has left the sender's Ethernet communication controller, which means that the last bit of the Ethernet frame/packet has been sent. **Tags:** atp.EnumerationValue=3

Table 5.26: TDEventFrameEthernetTypeEnum

Class	**TDHeaderIdRange**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	Specifies a range of PDU header identifiers. This range is specified by a minimum and maximum header identifier; and the maximum header identifier shall be greater than or equal the minimum header identifier.			
Base	*ARObject*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
maxHeaderId	Integer	1	attr	Specifies the maximum PDU header identifier, in other words the upper bound of a range of PDU header identifiers.
minHeaderId	Integer	1	attr	Specifies the minimum PDU header identifier, in other words the lower bound of a range of PDU header identifiers.

Table 5.27: TDHeaderIdRange

Class	***TDEventCycleStart*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is the abstract parent class to describe timing events related to a point in time where a communication cycle starts. Via the attribute "cycleRepetition", a filtered view to the cycle start can be defined.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	TDEventFrClusterCycleStart, TDEventTTCanCycleStart			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
cycleRepetition	Integer	1	attr	The start of every <cycleRepetition> cycle is targeted by this event.

Table 5.28: TDEventCycleStart

[TPS_TIMEX_00025] `TDEventFrClusterCycleStart` specifies the event related to the start of a FlexRay communication cycle ⌈ The element `TDEventFrClusterCycleStart` is used to specify events, namely the start of a communication cycle, observable at the physical layer of the FlexRay bus. ⌋*(RS_TIMEX_00001)*

Class	**TDEventFrClusterCycleStart**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is used to describe the timing event related to a point in time where a communication cycle starts on a FlexRay cluster.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TDEventCycleStart*, *TimingDescription*, *TimingDescriptionEvent*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
frCluster	FlexrayCluster	1	ref	The scope of this timing event.

Table 5.29: TDEventFrClusterCycleStart

[TPS_TIMEX_00026] `TDEventTTCanCycleStart` specifies the event related to the start of a TTCAN communication cycle ⌈ The element `TDEventTTCanCycleStart` is used to specify events, namely the start of a communication cycle, observable at the physical layer of the TTCAN bus. ⌋*(RS_TIMEX_00001)*

Class	**TDEventTTCanCycleStart**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is used to describe the timing event related to a point in time where a communication cycle starts on a TTCAN cluster.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TDEventCycleStart*, *Timing Description*, *TimingDescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
ttCanCluster	TtcanCluster	1	ref	The scope of this timing event.

Table 5.30: TDEventTTCanCycleStart

5.4 Timing Events Related to the BSW

[TPS_TIMEX_00028] `TDEventBswInternalBehavior` specifies observable events of BSW module entities ⌈ The element `TDEventBswInternalBehavior` is used to specify events, namely the activation, start and termination of BSW module entities, which are observable in the Basic Software Module view. ⌋ *(RS_TIMEX_00001)*

Events related to the BSW can be used during the specification of:

- `BswModuleTiming` 3.5.1
- `EcuTiming` 3.6

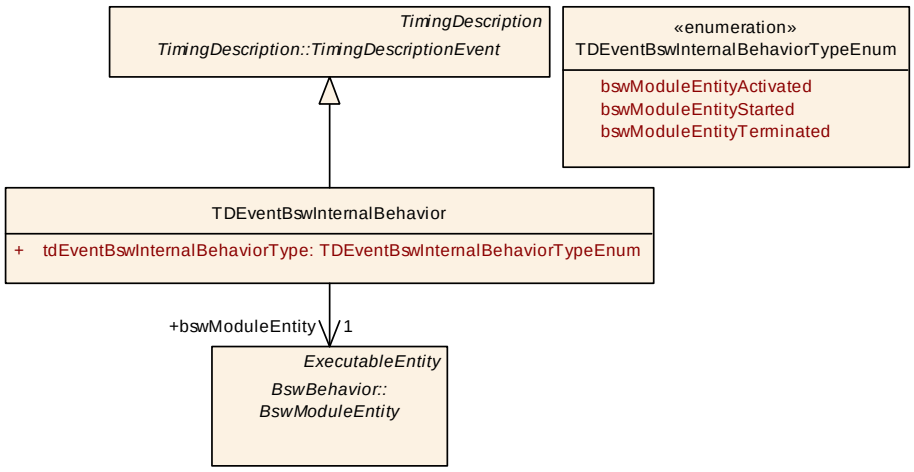


Figure 5.10: Events related to the internal structure of a BSW module

Class	**TDEventBswInternalBehavior**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventBswInternalBehavior			
Note	This is used to describe timing events related to the BswInternalBehavior of a BSW module.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
bswModule Entity	BswModuleEntity	1	ref	The scope of this timing event.
tdEventBsw Internal BehaviorType	TDEventBswInternal BehaviorTypeEnum	1	attr	The specific type of this timing event.

Table 5.31: TDEventBswInternalBehavior

Please note: For every `TDEventBswInternalBehavior` its scope is defined by the *bswModuleEntity* reference. It points to the BSW module entity for which the event can be observed. This scope definition assumes that every BSW module exists only once on each ECU. Otherwise the scope would not be precise enough because every module instance would bring the same BSW module entities.

Enumeration	**TDEventBswInternalBehaviorTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventBswInternalBehavior
Note	This is used to describe the specific event type of a TDEventBswInternalBehavior.
Literal	***Description***
bswModuleEntity Activated	A point in time where the associated BswModuleEntity has been activated, which means that it has entered the state "to be started". **Tags:** atp.EnumerationValue=0
bswModuleEntity Started	A point in time where the associated BswModuleEntity has entered the state "started" after its activation. **Tags:** atp.EnumerationValue=1
bswModuleEntity Terminated	A point in time where the associated BswModuleEntity has terminated and entered the state "suspended" **Tags:** atp.EnumerationValue=2

Table 5.32: TDEventBswInternalBehaviorTypeEnum

[TPS_TIMEX_00029] Purpose of `TDEventBsw` ⌈ The element `TDEventBsw` is used to specify events which are observable in the Basic Software Module view, which means that the occurrences of such events are observable between the Basic Software Modules. ⌋*(RS_TIMEX_00001)*

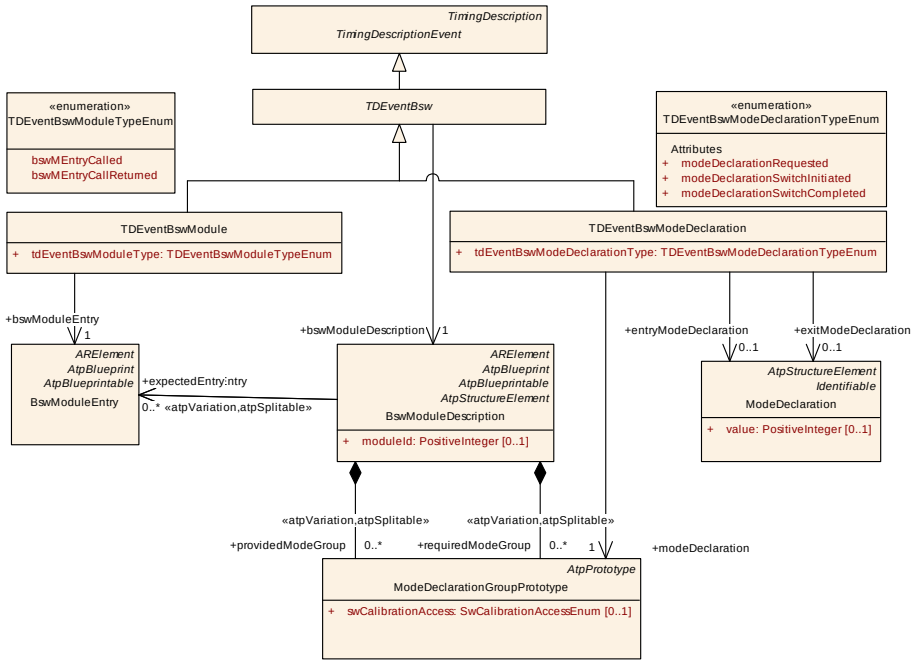


Figure 5.11: Events dealing with inter BSW module relations and mode communications on BSW level

[TPS_TIMEX_00030] `TDEventBswModule` specifies observable events when basic software entries are called ⌈ The element `TDEventBswModule` is used to specify events, namely the calling of and return from called basic software module entries, observable when such entries are called within the Basic Software. ⌋ *(RS_TIMEX_00001)*

Class	**TDEventBswModule**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventBsw			
Note	This is used to describe timing events related to the interaction between BSW modules.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventBsw*, *TimingDescription*, *TimingDescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
bswModuleEntry	BswModuleEntry	1	ref	The scope of this timing event.
tdEventBswModuleType	TDEventBswModuleTypeEnum	1	attr	The specific type of this timing event.

Table 5.33: TDEventBswModule

Enumeration	**TDEventBswModuleTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventBsw
Note	This is used to describe the specific event type of a TDEventBswModule.
Literal	***Description***
bswMEntryCallReturned	A point in time where the call of the associated BswModuleEntry has returned. **Tags:** atp.EnumerationValue=1

▽

△

Enumeration	**TDEventBswModuleTypeEnum**
bswMEntryCalled	A point in time where the associated BswModuleEntry has been called. **Tags:** atp.EnumerationValue=0

Table 5.34: TDEventBswModuleTypeEnum

[TPS_TIMEX_00031] `TDEventBswModeDeclaration` **specifies observable events in case of BSW mode communication** ⌈ The element `TDEventBswModeDeclaration` is used to specify events that are observable when mode changes are initiated and propagated in the Basic Software. ⌋*(RS_TIMEX_00001)*

Class	**TDEventBswModeDeclaration**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventBsw			
Note	This is used to describe timing events related to the mode communication on BSW level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventBsw*, *TimingDescription*, *Timing DescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
entryMode Declaration	ModeDeclaration	0..1	ref	Optional parameter which refines the scope of the TDEventBswModeDeclaration. If the parameter is set, the event occurs only if the mode declaration group prototype instance shall enter into the referenced ModeDeclaration.
exitMode Declaration	ModeDeclaration	0..1	ref	Optional parameter which refines the scope of the TDEventBswModeDeclaration. If the parameter is set, the event occurs only if the mode declaration group prototype instance shall exit from the referenced ModeDeclaration.
mode Declaration	ModeDeclarationGroup Prototype	1	ref	The scope of this timing event.
tdEventBsw Mode DeclarationType	TDEventBswMode DeclarationTypeEnum	1	attr	The specific type of this timing event.

Table 5.35: TDEventBswModeDeclaration

Enumeration	**TDEventBswModeDeclarationTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventBsw
Note	This is used to describe the specific event type of a TDEventBswModeDeclaration.
Literal	***Description***
modeDeclaration Requested	A point in time where the associated ModeDeclarationGroupPrototype has been requested. **Tags:** atp.EnumerationValue=0
modeDeclaration SwitchCompleted	A point in time where the switch to the associated ModeDeclarationGroupPrototype has been completed. **Tags:** atp.EnumerationValue=1
modeDeclaration SwitchInitiated	A point in time where the switch to the associated ModeDeclarationGroupPrototype has been initiated by the BswM. **Tags:** atp.EnumerationValue=2

Table 5.36: TDEventBswModeDeclarationTypeEnum

5.5 Complex Timing Event

[TPS_TIMEX_00027] Purpose of `TDEventComplex` ⌈ The element `TDEventComplex` is used to specify relationships between occurrences of events. ⌋ *(RS_TIMEX_00001)*

Complex timing events can be used during the specification of:

- `VfbTiming` 3.2
- `SwcTiming` 3.3
- `SystemTiming` 3.4
- `BswModuleTiming` 3.5.1
- `EcuTiming` 3.6

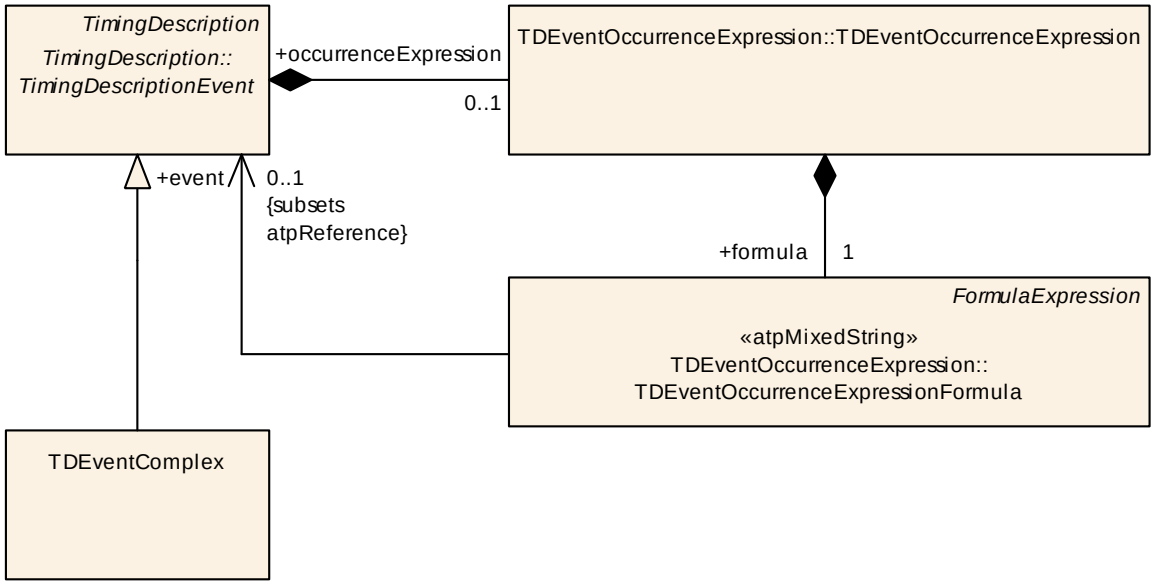


Figure 5.12: Complex timing event

Class	**TDEventComplex**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventComplex			
Note	This is used to describe complex timing events. The context of a complex timing event either is described informally, e.g. using the documentation block, or is described formally by the associated TDEventOccurrenceExpression.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 5.37: TDEventComplex

A complex timing event is a special observable event. In comparison to the ”atomic” events described above a complex event does not contain information about the context it references, like `VariableDataPrototype` in `TDEventVariableDataPro-`

totype. Instead, a complex event uses the occurrence expression to specify the context with regards to occurrences of TimingDescriptionEvents as describe in the following section.

5.6 Occurrence Expression Language for Timing Events

The TimingDescriptionEvents mentioned in the previous sections allow to specify observable events with a well-defined context. However, sometimes the context information of the events is not sufficient, because additional conditions, like a value filter or additional stimuli, influence the occurrence. Thus, the occurrence expression provides means to overcome the limitations of atomic events.

The occurrence expression provides the ability to refine the context specification of a timing event for the following cases:

Content Filter filters occurrences of an atomic event based on the *value* of exchanged data or operation arguments.

Complex Event combines any number of atomic and complex event to specify a new timing event.

5.6.1 Specifying an Occurrence Expression

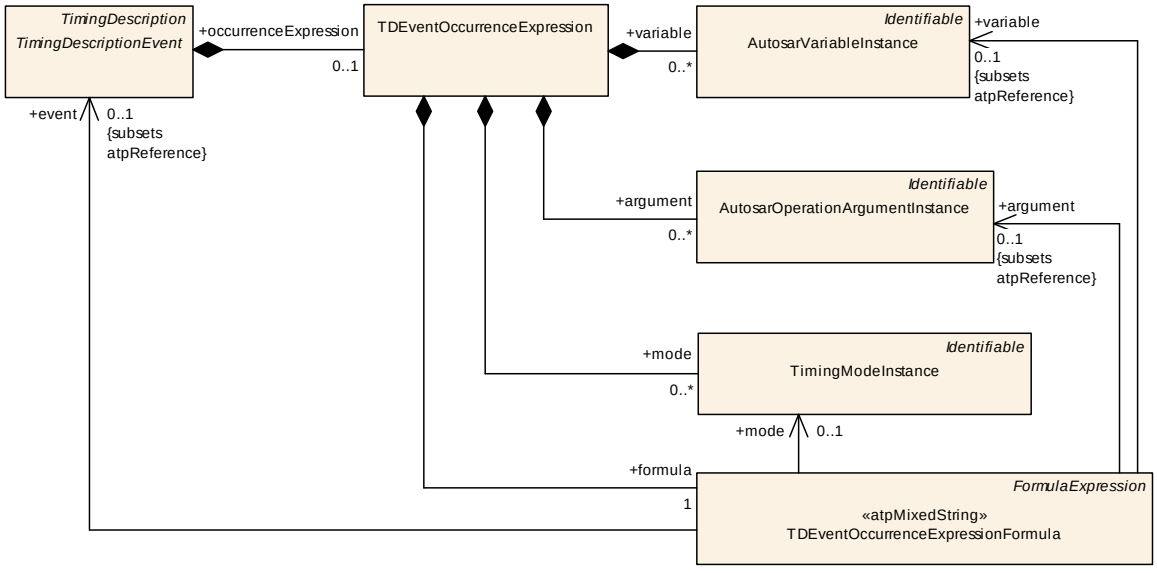


Figure 5.13: The occurrence expression

As shown in Figure 5.13, each TimingDescriptionEvent aggregates a TDEventOccurrenceExpression as optional parameter. A TDEventOccurrenceExpression is a container for all information required to formulate the expression. The expression itself is defined via TDEventOccurrenceExpressionFormula which is

derived from FormulaExpression (see Generic Structure Template [5]). The TDEventOccurrenceExpressionFormula uses the capabilities of the FormulaExpression and adds the following functions to the expression language:

- The function *TIMEX_value*, which requires as operand either a reference to an AutosarVariableInstance or a reference to an AutosarOperationArgumentInstance whose value shall be evaluated. The return type of this function is Numerical (see constraint [constr_4551]).
- The function *TIMEX_occurs*, which requires as operand a reference to the TimingDescriptionEvent whose occurrence shall be evaluated. The return type of this function is Boolean. It returns TRUE if the referenced timing event occurs at the point in time the expression is evaluated.
- The function *TIMEX_hasOccurred*, which requires as operand a reference to the TimingDescriptionEvent whose occurrence shall be evaluated. The return type of this function is Boolean. It returns TRUE if the referenced timing event has occurred *at least once* before or at the same point in time the expression is evaluated.
- The function *TIMEX_timeSinceLastOccurrence*, which requires as operand a reference to the TimingDescriptionEvent whose occurrence shall be evaluated. The return type of this function is Float and the unit is seconds. It returns the time difference between the point in time of the last occurrence of the referenced event and the point in time the expression is evaluated.
- The function *TIMEX_angleSinceLastOccurrence*, which requires as operand a reference to the TimingDescriptionEvent whose occurrence shall be evaluated. The return type of this function is Float and the unit is degree. It returns the angle of the crank shaft between the point in time of the last occurrence of the referenced event and the point in time the expression is evaluated.
- The function *TIMEX_modeActive* queries the TimingModeInstance specified as argument. The return type of this function is Boolean. It returns TRUE if the specified mode declaration is *active* at the point in time the expression is evaluated, otherwise it returns FALSE.

All operands required by the functions are references to model elements. Thus, TDEventOccurrenceExpressionFormula requires references to the respective elements of type TimingDescriptionEvent, AutosarVariableInstance and AutosarOperationArgumentInstance. Due to the atpMixedString nature of the TDEventOccurrenceExpressionFormula several references can be used within the occurrence expression.

[constr_4500] Restricted usage of functions ⌈ The functions *TIMEX_occurs*, *TIMEX_hasOccurred*, *TIMEX_timeSinceLastOccurrence*, *TIMEX_angleSinceLastOccurrence*, and *TIMEX_modeActive* can only be used for occurrence expressions, which are applied to events of type TDEventComplex. ⌋ *()*

[constr_4501] Application rule for the occurrence expression in `TDEventComplex` ⌈ The occurrence expression shall be specified such that it describes an *event* rather than a state. As a consequence the occurrence expression must ensure that a complex timing event *could* only occur at the occurrence time of one of the referenced `TimingDescriptionEvent`s. ⌋*()*

[constr_4502] Use references only as function operands ⌈ The references to model elements (e.g. the *timing event* reference targeting `TimingDescriptionEvent`) do have specific semantics. The usage of these references within the expression is *only* allowed as operand of the functions mentioned above. ⌋*()*

[constr_4551] Use only Numericals in `TDEventOccurrenceExpression` ⌈ The target data prototype of the instance references of `variable` and `argument` shall be `Numerical`. ⌋*()*

The example given below shows how to combine the functions mentioned above. The occurrence expression for a complex event called *EC* shall be described. Figure 5.14 sketches the AUTOSAR software component model of this example.

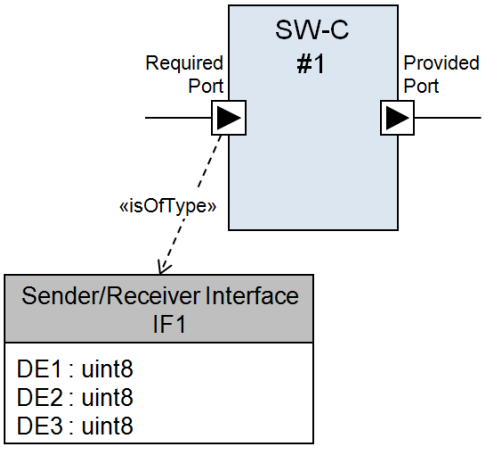


Figure 5.14: The software architecture used by the Occurrence Expression Example

The complex event *EC* occurs when the following conditions are fulfilled:

Condition1 Either atomic timing event *E1* or *E2* must occur. In this example, *E1* and *E2* are atomic timing events which occur when the `VariableDataPrototype` *DE1* and *DE2* are received on `PortPrototype` called *Required Port* of the software component called *SW-C 1*.

Condition2 `VariableDataPrototype` called *DE3* must be greater than 3.

Condition3 The `VariableDataPrototype`s called *DE1* and *DE2* must be received within a time interval of maximum 0.5 milliseconds.

The complex event *EC* would be described by the following occurrence expression:

```
//Condition 1
```

```
( TIMEX_occurs( /example/expression/E1 )
   || TIMEX_occurs( /example/expression/E2 ) )
//Condition 2
&& TIMEX_value( /example/expression/EC/DE3 ) > 3
//Condition 3
&& abs( TIMEX_timeSinceLastOccurrence( /example/expression/E1 ) -
   TIMEX_timeSinceLastOccurrence( /example/expression/E2 ) ) <= 0.0005
```

Due to the first condition the complex event *EC* can only occur when one of the atomic events *E1* or *E2* occurs at the point in time of evaluation. Thus, this expression satisfies the semantics constraint defined in [constr_4501].

The corresponding AUTOSAR XML file fragment for the complex event *EC* has the following appearance:

Listing 5.1: AUTOSAR XML representation of the occurrence expression for the complex event EC

```
<AR-PACKAGE>
  <SHORT-NAME>example</SHORT-NAME>
  <ELEMENTS>
    <VFB-TIMING>
      <SHORT-NAME>expression</SHORT-NAME>
      <TIMING-DESCRIPTIONS>
        <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
          <SHORT-NAME>E1</SHORT-NAME>
          ...
        </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
        <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
          <SHORT-NAME>E2</SHORT-NAME>
          ...
        </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
        <TD-EVENT-COMPLEX>
          <SHORT-NAME>EC</SHORT-NAME>
          <OCCURRENCE-EXPRESSION>
            <VARIABLES>
              <AUTOSAR-VARIABLE-INSTANCE>
                <SHORT-NAME>D3</SHORT-NAME>
                <VARIABLE-INSTANCE-IREF>...</VARIABLE-INSTANCE-IREF>
              </AUTOSAR-VARIABLE-INSTANCE>
            </VARIABLES>
          <FORMULA>
            ((TIMEX_occurs(&lt;EVENT-REF DEST=&quot;TD-EVENT-VARIABLE-DATA-
               PROTOTYPE&quot;>/example/expression/E1&lt;/EVENT-REF>)
            || TIMEX_occurs(&lt;EVENT-REF DEST=&quot;TD-EVENT-VARIABLE-DATA-
               PROTOTYPE&quot;>/example/expression/E2&lt;/EVENT-REF>))
            &amp;&amp;
            TIMEX_value(&lt;VARIABLE-REF DEST=&quot;AUTOSAR-VARIABLE-INSTANCE
               &quot;>/example/expression/EC/DE3 &lt;/VARIABLE-REF>) > 3
            &amp;&amp;
            abs(TIMEX_timeSinceLastOccurrence(&lt;EVENT-REF DEST=&quot;TD-
               EVENT-VARIABLE-DATA-PROTOTYPE&quot;>
              /example/expression/E1&lt;/EVENT-REF>)
            - TIMEX_timeSinceLastOccurrence(&lt;EVENT-REF DEST=&quot;TD-EVENT
               -VARIABLE-DATA-PROTOTYPE&quot;>
                /example/expression/E1&lt;EVENT-REF>))
```

```
        &lt;= 0.0005)
      </FORMULA>
        </OCCURRENCE-EXPRESSION>
      </TD-EVENT-COMPLEX>
    </TIMING-DESCRIPTIONS>
    <COMPONENT-REF DEST="COMPOSITION-SW-COMPONENT-TYPE">/example/
      expression/SWC1</COMPONENT-REF>
  </VFB-TIMING>
 </ELEMENTS>
</AR-PACKAGE>
```

Class	**TDEventOccurrenceExpression**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventOccurrenceExpression			
Note	This is used to specify a filter on the occurrences of TimingDescriptionEvents by means of a TDEvent OccurrenceExpressionFormula. Filter criteria can be variable and argument values, i.e. the timing event only occurs for specific values, as well as the temporal characteristics of the occurrences of arbitrary timing events.			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
argument	AutosarOperation ArgumentInstance	*	aggr	An occurrence expression can reference an arbitrary number of OperationArgumentPrototypes in its expression. This association aggregates instance references to OperationArgumentPrototypes which can be referenced in the expression.
formula	TDEventOccurrence ExpressionFormula	1	aggr	This is the expression formula which is used to describe the occurrence expression.
mode	TimingModeInstance	*	aggr	An occurrence expression can reference an arbitrary number of TimingModeInstances in its expression. This association aggregates instance references to Mode Declaration which can be referenced in the expression.
variable	AutosarVariable Instance	*	aggr	An occurrence expression can reference an arbitrary number of VariableDataPrototpyes in its expression. This association aggregates instance references to Variable DataPrototypes which can be referenced in the expression.

Table 5.38: TDEventOccurrenceExpression

Class	«atpMixedString» **TDEventOccurrenceExpressionFormula**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventOccurrenceExpression			
Note	This is an extension of the FormulaExpression for the AUTOSAR Timing Extensions. A TDEventOccurrenceExpressionFormula provides the means to express the temporal characteristics of timing event occurrences in correlation with specific variable and argument values. The formal definition of the extended functions (ExtUnaryFunctions) is described in detail in the AUTOSAR Timing Extensions.			
Base	*ARObject*, *FormulaExpression*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
argument	AutosarOperation ArgumentInstance	0..1	ref	This is one particular argument value used in the expression formula.

▽

△

Class	«atpMixedString» **TDEventOccurrenceExpressionFormula**			
event	TimingDescriptionEvent	0..1	ref	This is one particular timing description event used in the expression formula.
mode	TimingModeInstance	0..1	ref	This is one particular mode used in the expression formula.
variable	AutosarVariable Instance	0..1	ref	This is one particular variable value used in the expression formula.

Table 5.39: TDEventOccurrenceExpressionFormula

Class	**AutosarVariableInstance**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventOccurrenceExpression::InstanceRefsUsage			
Note	This class represents a reference to a variable instance within AUTOSAR. This way it is possible to reference a variable instance in the occurrence expression formula. The variable instance can target to one of the following variables: • a variable provided via a PortPrototype as whole • an element inside of a composite variable provided via a PortPrototype			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
variableInstance	DataPrototype	1	iref	This is the reference to the instanceRef definition.

Table 5.40: AutosarVariableInstance

Class	**AutosarOperationArgumentInstance**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventOccurrenceExpression::InstanceRefsUsage			
Note	This class represents a reference to an argument instance. This way it is possible to reference an argument instance in the occurrence expression formula. The argument instance can target to one of the following arguments: • a whole argument used in an operation of a PortPrototype with ClientServerInterface • an element inside of a composite argument used in an operation of a PortPrototype with Client ServerInterface			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
operation Argument Instance	DataPrototype	1	iref	This is the reference to the instanceRef definition.

Table 5.41: AutosarOperationArgumentInstance

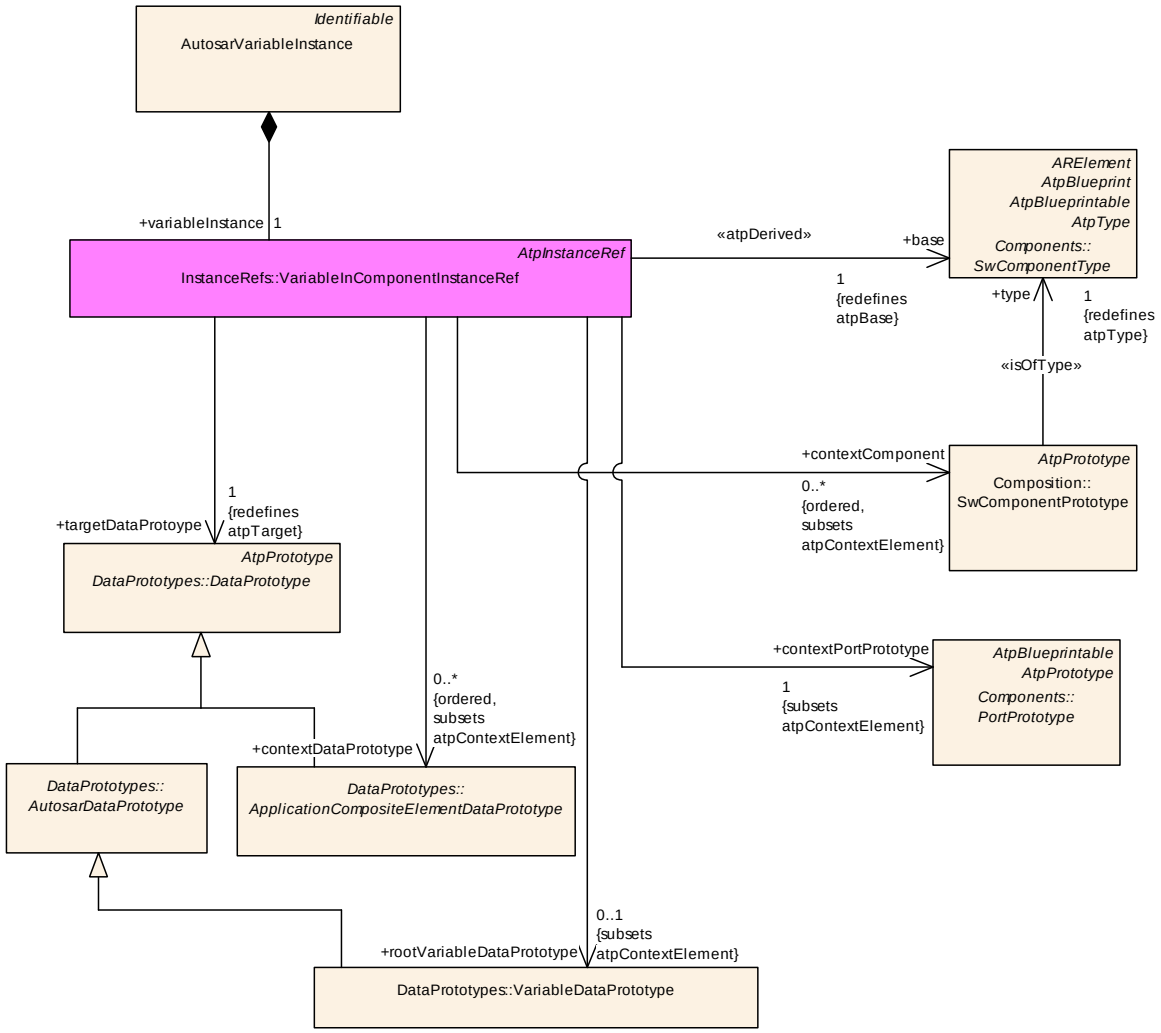


Figure 5.15: The required context information to reference a variable instance within AUTOSAR.

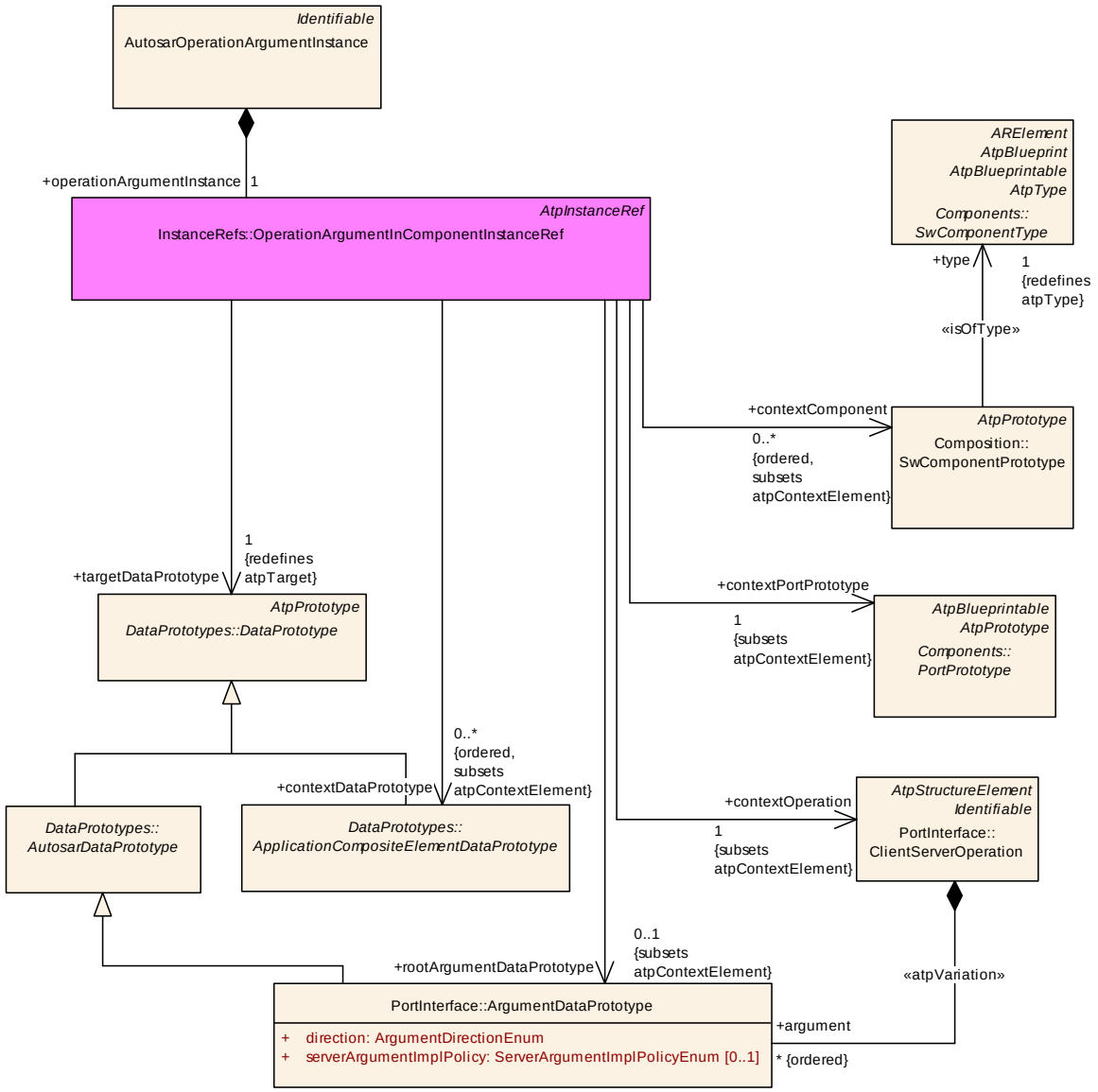


Figure 5.16: The required context information to reference an operation argument instance within AUTOSAR.

5.6.2 Occurrence Expression Language Syntax

The occurrence expression language is based on the syntax of the formula language defined in the Generic Structure Template [5]. It extends the language by additional functions and additional references to model elements. In the following, the implications of the extensions to the syntax are presented based on the grammar definition.

Note: The grammar defined for the formula language is not part of the listing below. It presents only the timing specific extensions of the formula language and the enhanced functions and references.

Listing 5.2: AUTOSAR Occurrence Expression Language

```
ExtUnaryFuncName   :   'TIMEX_value' |
                       'TIMEX_occurs' |
                       'TIMEX_hasOccurred' |
                       'TIMEX_timeSinceLastOccurrence' |
                       'TIMEX_angleSinceLastOccurrence' |
                       'TIMEX_modeActive'
                       ;
```

5.6.3 Interpreting an Occurrence Expression

Based on the specification mechanism described in the previous sections it is possible to use the occurrence expression formula to refine the timing specification to the intended precision. This section describes how such an occurrence expression has to be interpreted. The duty of the interpreter is to determine the occurrences of the TimingDescriptionEvent for which the occurrence expression is defined. This is done in two ways, depending on whether the occurrence expression is used as a content filter or as a complex event.

5.6.3.1 Interpreting a Content Filter

In this case, the occurrence expression is defined for an atomic event. Only the unary timing function *TIMEX_value(<reference to argument or variable>)* is allowed to be used for the content filter. On each occurrence of the atomic event the interpreter checks whether the content filter defined by the expression is fulfilled. This is done by evaluating the function *TIMEX_value* based on its operand type:

AutosarVariableInstance the value of the referenced variable is evaluated at the point in time the atomic event occurs.

AutosarOperationArgumentInstance the value of the referenced argument is evaluated at the point in time the atomic event occurs.

[constr_4552] Restricted usage of AutosarVariableInstance for Content Filter ⌈ If a content filter is defined for an atomic event then references to AutosarVariableInstances are only allowed if the atomic event is of type TDEventVariableDataPrototype. Only if such an atomic event occurs, the value of the variables can be evaluated. Thus, also the scope of the atomic event must be the same as the AutosarVariableInstance, meaning that they must point to the same VariableDataPrototype. ⌋()

[constr_4503] Restricted usage of AutosarOperationArgumentInstance for Content Filter ⌈ If a content filter is defined for an atomic event then references to AutosarOperationArgumentInstances are only allowed if the atomic event is of type TDEventOperation. Only if such an atomic event occurs, the value of the operation arguments can be evaluated. Thus, also the scope of the atomic event must be the same as the AutosarOperationArgumentInstance, meaning that they must

point to the same ClientServerOperation. Finally, references to an AutosarOperationArgumentInstance with argument direction "out" are only allowed, if the atomic event of type TDEventOperation refers either to the point in time when the operation call response has been sent (TD-EVENT-OPERATION-TYPE=OPERATION-CALL-RESPONSE-SENT) or to the point in time when the operation call response has been received (TD-EVENT-OPERATION-TYPE=OPERATION-CALL-RESPONSE-RECEIVED). ⌋()

5.6.3.2 Interpreting a Complex Event

In this case, the occurrence expression is defined for a complex event. All features of the occurrence expression language can be used for this expression type. At a specific point in time *t*, the interpreter evaluates the expression to determine if the complex event has occurred.

Considering the occurrence expression defined for the example given in Section 5.6.1, the interpreter "implements" a function *EC(t)* which returns TRUE, if the complex event EC occurs at time *t*:

```
EC(t) =
( TIMEX_occurs( t, /example/expression/E1 )
   || TIMEX_occurs( t, /example/expression/E2 ) )
&& TIMEX_value( t, /example/expression/EC/DE3 ) > 3
&& abs( TIMEX_timeSinceLastOccurrence( t, /example/expression/E1 ) -
   TIMEX_timeSinceLastOccurrence( t, /example/expression/E2 ) ) <= 0.0005
```

Since the expression satisfies [constr_4501], it must only be evaluated at occurrence times of *E1* or *E2*, because only then the complex event *EC* can occur and the expression can return TRUE.

As shown in the sketched trace in Figure 5.17 the timing description events called *E1* and *E2* occur at different times. On the left hand side of this figure the two events occur within a time interval of 0.0005 seconds. The point in time the given occurrence expression is evaluated is the point in time the event *E2* occurs. The result of the occurrence expression at this point in time, $t_{evaluate}$ respectively t_{E2}, is TRUE. On the right hand side of this figure the two events do not occur within a time interval of 0.0005 seconds. The point in time the given occurrence expression is evaluated is the point in time the event *E1* occurs. The result of the occurrence expression at this point in time, $t_{evaluate}$ respectively t_{E1}, is FALSE.

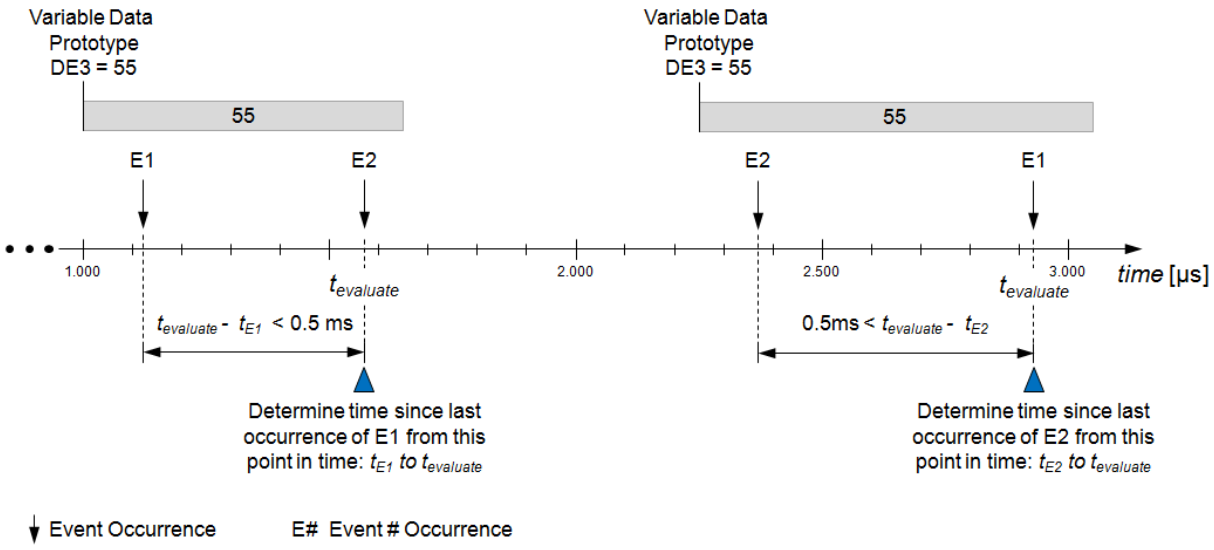


Figure 5.17: Trace showing various occurrences of the timing description events *E1* and *E2*, as well as the value of the variable *DE3*.

Based on the several functions provided by the occurrence expression language, the interpreter requires the following information from the system:

- the value of a referenced `AutosarOperationArgumentInstance` at time *t*.
- the value of a referenced `AutosarVariableInstance` at time *t*.
- the occurrences of a referenced `TimingDescriptionEvent` at time *t* and before.

There are different ways to gather the required information:

- Model analysis and simulation: In a deterministic system environment, occurrences of `TimingDescriptionEvent`s can be determined offline, for example the point in time a frame will be transmitted in the static segment of a FlexRay network.
- Target trace: The required information can be gathered from a running system by recording the points in time a `TimingDescriptionEvent` has occurred. For example, an ECU trace for `TDEventSwcInternalBehavior` may contain a marker for each point in time a `RunnableEntity` is activated, started or terminated.

If the interpreter has the required information as input, the different functions provided by the occurrence expression language can be interpreted as follows:

- TIMEX_value(t, <reference to an `AutosarVariableInstance`>) returns the variable value at time *t*.
- TIMEX_value(t, <reference to an `AutosarOperationArgumentInstance`>) returns the operation argument value at time *t*.

- TIMEX_occurs(t, <reference to a TimingDescriptionEvent>) returns TRUE (or 1) if the referenced event has occurred at time *t*, else it returns FALSE (or 0).
- TIMEX_hasOccurred(t, <reference to a TimingDescriptionEvent>) returns TRUE (or 1) if the referenced event has occurred *at least once* before or at time *t*.
- TIMEX_timeSinceLastOccurrence(t, <reference to a TimingDescriptionEvent>) returns the time difference between *t* and the point in time of the last occurrence of the referenced event. The unit of time is seconds.
- TIMEX_angleSinceLastOccurrence(t, <reference to a TimingDescriptionEvent>) returns the angle difference between *t* and the point in time of the last occurrence of the referenced event. The unit of angle is degree.
- TIMEX_modeActive(t, <reference to a TimingModeInstance>) returns TRUE (or 1) if the referenced mode is active at time *t*, else it returns FALSE (or 0).

6 Timing Description Event Chains

[TPS_TIMEX_00002] Purpose of TimingDescriptionEventChain ⌈ The element TimingDescriptionEventChain is used to specify a causal relationship between timing description events and their occurrences during the runtime of a system. ⌋ *(RS_TIMEX_00001, RS_TIMEX_00004, RS_TIMEX_00005, RS_TIMEX_00009)*

A timing event chain describes a causal order for a set of functionally dependent timing events. Each event chain defines at least the relationship between two differing events, its *stimulus* and *response* [constr_4515]. This means that if the stimulus event occurs then the response event occurs after or in other words the response event follows if and only if the stimulus event occurred before.

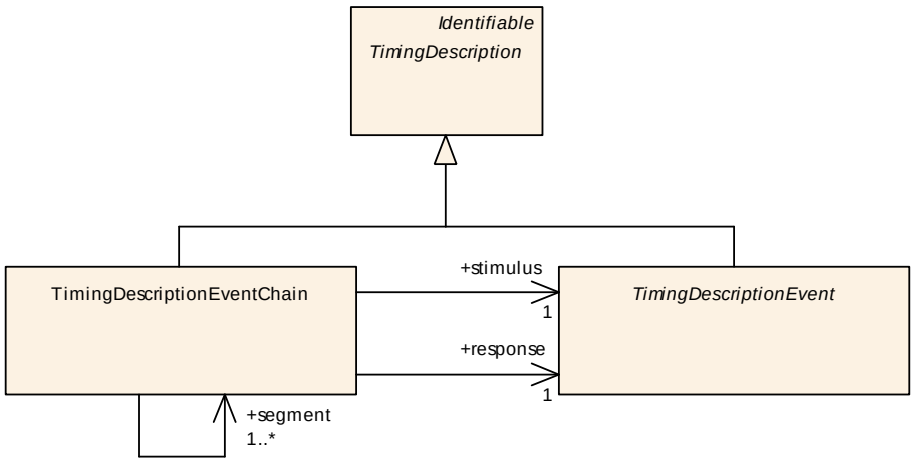


Figure 6.1: TimingDescriptionEventChain

Class	**TimingDescriptionEventChain**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription			
Note	An event chain describes the causal order for a set of functionally dependent timing events. Each event chain has a well defined stimulus and response, which describe its start and end point. Furthermore, it can be hierarchically decomposed into an arbitrary number of sub-chains, so called *event chain segments*.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
response	TimingDescriptionEvent	1	ref	The response event representing the point in time where the event chain is terminated. **Tags:** xml.sequenceOffset=20
segment	TimingDescriptionEvent Chain	1..*	ref	A composed event chain consists of an arbitrary number of sub-chains. **Tags:** xml.sequenceOffset=30
stimulus	TimingDescriptionEvent	1	ref	The stimulus event representing the point in time where the event chain is activated. **Tags:** xml.sequenceOffset=10

Table 6.1: TimingDescriptionEventChain

Thus, by means of an event chain, the correlation between a stimulation of a system and its corresponding response can be explicitly described, and used as a formalized definition of the scope for timing constraints. This is important, because timing constraints refer to a specific part of the overall system's timing and need clear validity semantics. Figure 6.2 shows an event chain "End-to-End Timing" describing the causal dependency between "Sensor" and "Actuator". The sequence of event chain segments shows the details of "End-to-End Timing" according to the AUTOSAR timing views.

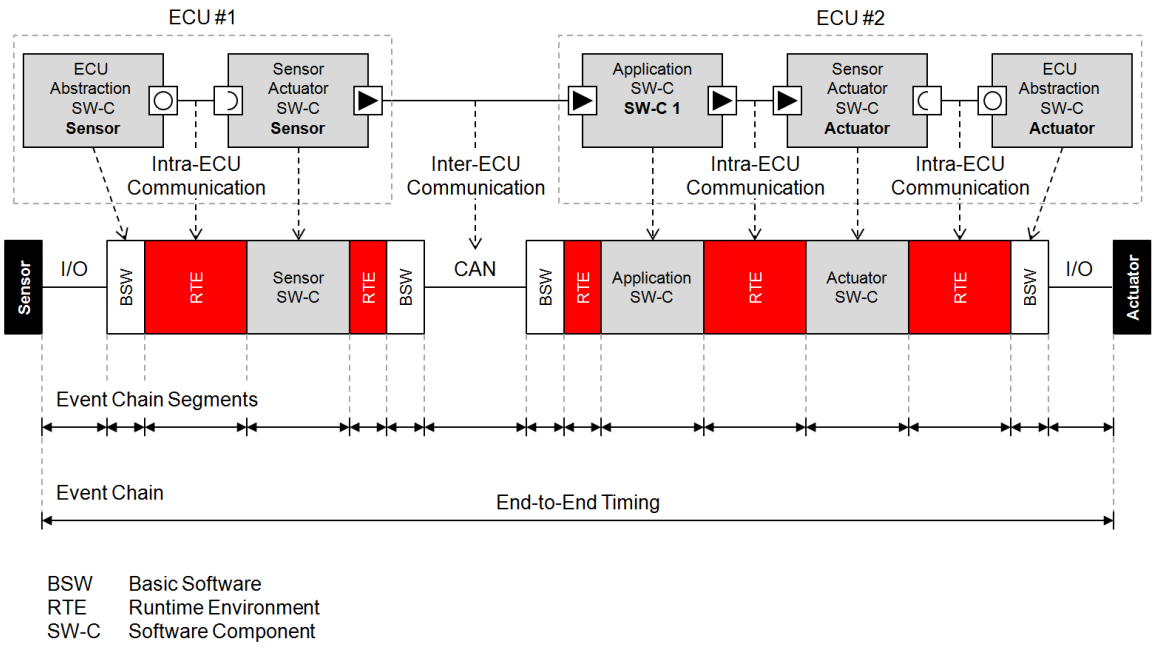


Figure 6.2: Example of an end-to-end event chain between sensor sampling and actuator access

[constr_4515] Specifying stimulus and response in TimingDescriptionEventChain ⌈ The references between TimingDescriptionEventChain and TimingDescriptionEvent playing the role stimulus and response shall not reference the same TimingDescriptionEvent. ⌋*()*

[constr_4516] Specifying event chain segments ⌈ If a TimingDescriptionEventChain consists of further event chain segments then at least one sequence of event chain segments shall exists from the event chain's stimulus to the response. ⌋*()*

[constr_4517] Referencing no further event chain segments ⌈ If a TimingDescriptionEventChain is not subdivided in further event chain segments, then the reference playing the role of segment shall reference this TimingDescriptionEventChain. In other words, an event chain without any event chain segment shall reference itself. ⌋*()*

[constr_4518] Specifying stimulus event and response event of first and last event chain segment ⌈ The stimulus event of the first event chain segment and the response event of the last event chain segment shall reference the stimulus and response of the parent event chain the event chain segments directly belong to. ⌋*()*

Table 6.2 lists all allowed values of the attribute category of the TimingDescriptionEventChain and describes the corresponding semantics.

Category	*Description*
empty	No specific semantics is imposed on the TimingDescriptionEventChain. It indicates the standard behavior.
STANDARD	No specific semantics is imposed on the TimingDescriptionEventChain. It indicates the standard behavior.
LET_INTERVAL	The TimingDescriptionEventChain plays the role of a LET interval. Any *implict* sender-receiver communication of executable entities mapped to the LET interval follow the LET semantics.

Table 6.2: Categories of TimingDescriptionEventChain

[TPS_TIMEX_00057] Attribute category used in context of TimingDescriptionEventChain and LET ⌈ The attribute category is used to indicate that the timing description event chain plays the role of a LET interval. ⌋*(RS_TIMEX_00022)*

The value of the attribute category of a TimingDescriptionEventChain indicates that the timing description event chain plays the role of a LET interval. The value LET_INTERVAL specifies that the timing description event chain plays the role of a LET interval.

[constr_4560] category of TimingDescriptionEventChain shall not be extended ⌈ In contrast to the general rule that category can be extended by user-specific values it is not allowed to extend the meaning of the attribute category of meta-class TimingDescriptionEventChain. ⌋*()*

[constr_4557] Usage of the category value LET_INTERVAL in TimingDescriptionEventChain ⌈ The value LET_INTERVAL of the attribute category of a TimingDescriptionEventChain shall be set if and only if the timing description event chain references 1) a timing description event playing the role stimulus and the value of the category of this referenced TimingDescriptionEvent is set to LET_RELEASE; and 2) a timing description event playing the role response and the value of the category of this referenced TimingDescriptionEvent is set to LET_TERMINATE. ⌋*()*

6.1 Approach

The following subsections describe how to structure event chains for systems. Depending on the pre-conditions two different approaches can be distinguished: top-down (decomposition) and bottom-up (composition).

The decomposition respectively composition of event chains can be performed according to the software component hierarchy, but does not necessarily have to follow this hierarchy. The primary purpose is to increase respectively decrease granularity of the timing descriptions.

Note that event chains are used in all AUTOSAR timing views and any composition and decomposition of event chains can be done across various AUTOSAR timing views.

6.1.1 Decomposition

In a first step the time critical path in the system is identified. This means that a causal relationship between a stimulus event and response event is described by an event chain. For this event chain a timing constraint is specified describing the time budget. The second step is to decompose this event chain into event chain segments which implies that the given time budget gets split — decomposed —, too.

Since event chain segments are event chains as well, these event chain segments can be subject to further decomposition.

Figure 6.3 shows a time critical path between the event ”requesting the brake pedal position” (*Stimulus*) and the event ”making available the determined vehicle speed” (*Response*). This event chain (*EC*) is subject to a timing constraint, namely a LatencyTimingConstraint, and is budgeted accordingly. For example, the time budget for the event chain *EC* is constrained by a maximum latency of 2 ms.
In subsequent steps of the development and with deeper knowledge about the system’s dynamics, this event chain and its time budget can be split across the system’s components. This results in the event chain segments *EC1*, *EC2* and *EC3* and their appropriate time budgets. The sum of these time budgets must not exceed the given time budget of 2 ms.

6.1.2 Composition

In the first step the system is build up based on available software components including timing descriptions. In the second step available event chains are connected with each other. This results in a sequence of event chains where the response event of one event chain plays the role of the stimulus event of the subsequent event chain. In the third step, a high-level event chain is specified based on a sequence of available event chains which play the role of event chain *segments*. For this high-level event chain a time budget must be specified. Finally, the aggregated time budget needs to be assessed if acceptable which means that the aggregated time budget must be equal or less than the time budget of the high-level event chain.

Figure 6.3 shows the connected event chains *EC1*, *EC2* and *EC3*. For each event chain a time budget, using a `LatencyTimingConstraint`, is specified: The time budget of event chain *EC1* is 0.5 ms, of event chain *EC2* is 0.6 ms and of event chain *EC3* is 0.7 ms. The high-level event chain *EC* is a composition of the event chains *EC1*, *EC2* and *EC3*. The stimulus event of the high-level event chain is the event ”requesting the brake pedal position” (*Stimulus*) and the response event of the high-level event chain is the event ”making available the determined vehicle speed” (*Response*). Eventually, a time budget is assigned to the high-level event chain using a `LatencyTimingConstraint`, for example 2 ms. This value is consistent with the aggregated time budget of the event chain segments (0.5 ms + 0.6 ms + 0.7 ms = 1.8 ms).

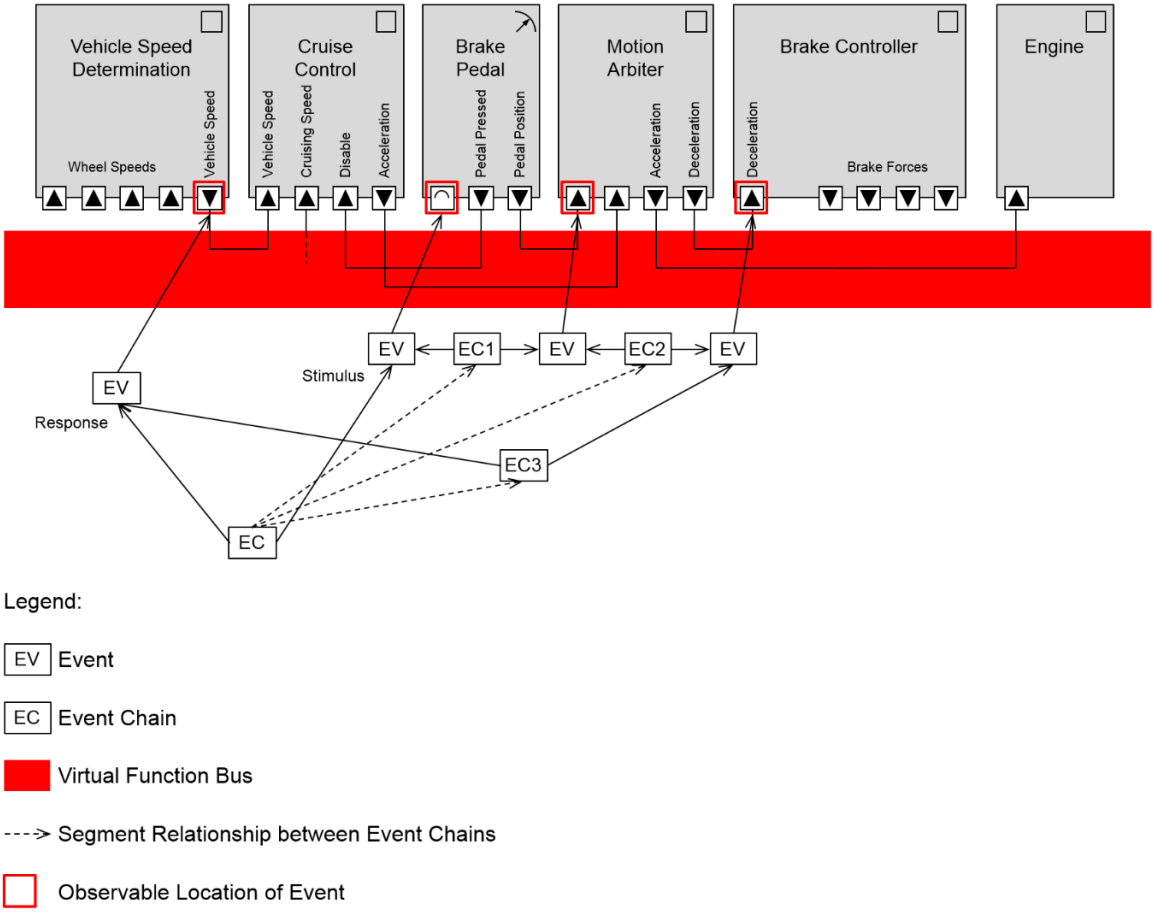


Figure 6.3: Example of a composed and decomposed event chain

6.2 Patterns

A sequence or hierarchy of event chains can form complex structures. However, if one of the aforementioned approaches is correctly followed then there is only a handful of patterns applicable. These patterns are introduced in the following with a simple example.

6.2.1 Sequence

The most frequently used pattern is the sequence of events. Such a sequence describes a succession of causally related events without an alternative path.

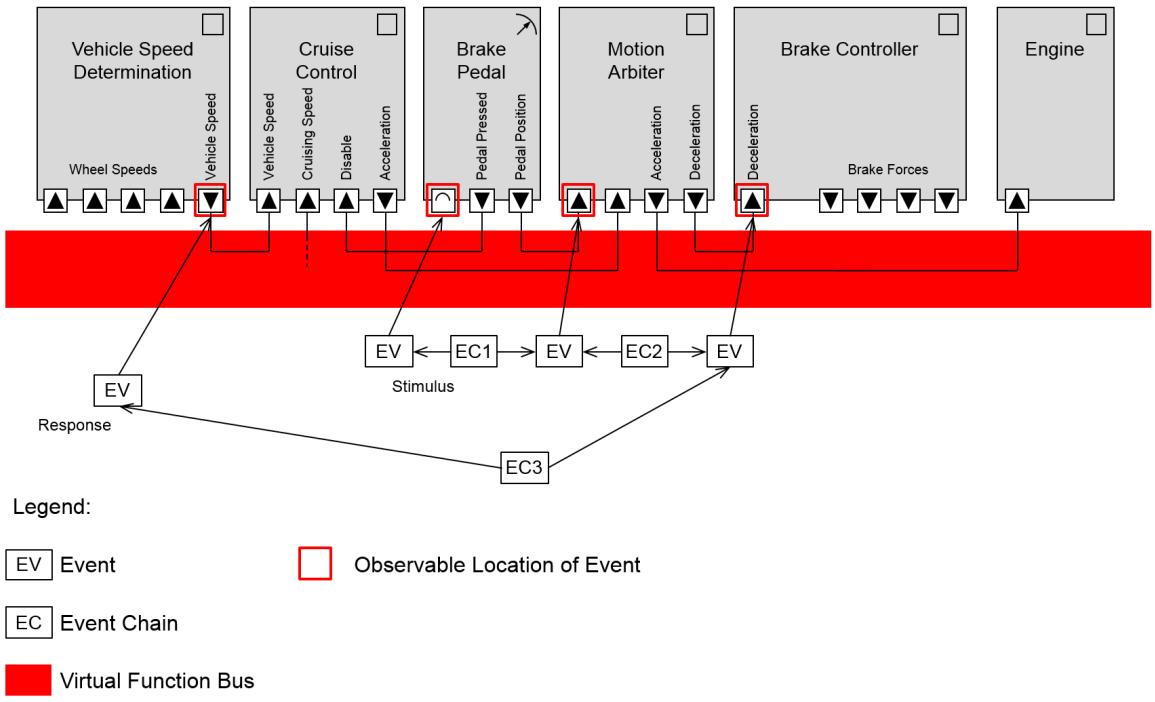


Figure 6.4: Example of the ”Sequence” pattern

An example for this pattern is depicted in Figure 6.4. The event chains *EC1* through *EC3* define a causal relationship of events observed at a port of the software component called *Brake Pedal* and a port of the software component called *Vehicle Speed Determination*.

6.2.2 Fork

The ”Fork” pattern describes the constellation where several event chains have one common stimulus event and different response events.

The pattern is illustrated in Figure 6.5, which shows a path that forks because the SW-C *Brake Controller* calculates the brake force value for each wheel (*EC5* through *EC8*).

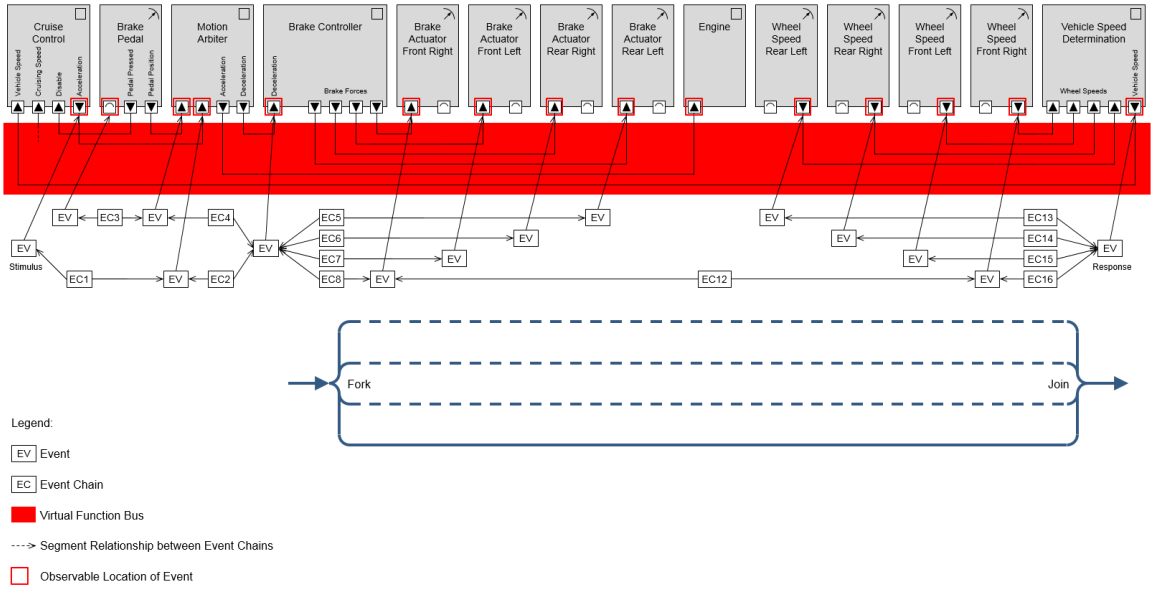


Figure 6.5: Example of the ”Fork” and ”Join” pattern

6.2.3 Join

The ”Join” pattern describes the constellation where several event chains have one common response event and different stimulus events.

The pattern is illustrated in Figure 6.5 which shows a path that joins because the SW-C *Vehicle Speed Determination* aggregates the wheel speed values from individual wheels (*EC13* through *EC16*).

6.2.4 Alternative

The ”Alternative” pattern describes the constellation where more than one path between a stimulus and response event exists. This implies that at least one ”Fork” is followed by at least one ”Join”.

The pattern is illustrated in Figure 6.6 which shows that an event observed at a required port of the SW-C *Motion Arbiter* leads to an occurrence of an event either at the required port called *Deceleration* of the SW-C *Brake Controller*, or at the required port called *Acceleration* of the SW-C *Engine*. These alternative causal relationships are described by the event chains *EC2* and *EC4* in this figure. In either case, the deceleration or acceleration of the vehicle leads to the occurrence of an event at the provided port called *Vehicle Speed* of the SW-C *Vehicle Speed Determination* reporting the vehicle’s speed. These alternative causal relationships are described by the event chains *EC3* and *EC5* which both reference the same response event. To fulfill the overall event chain, only one of the alternative paths must have been occurred.

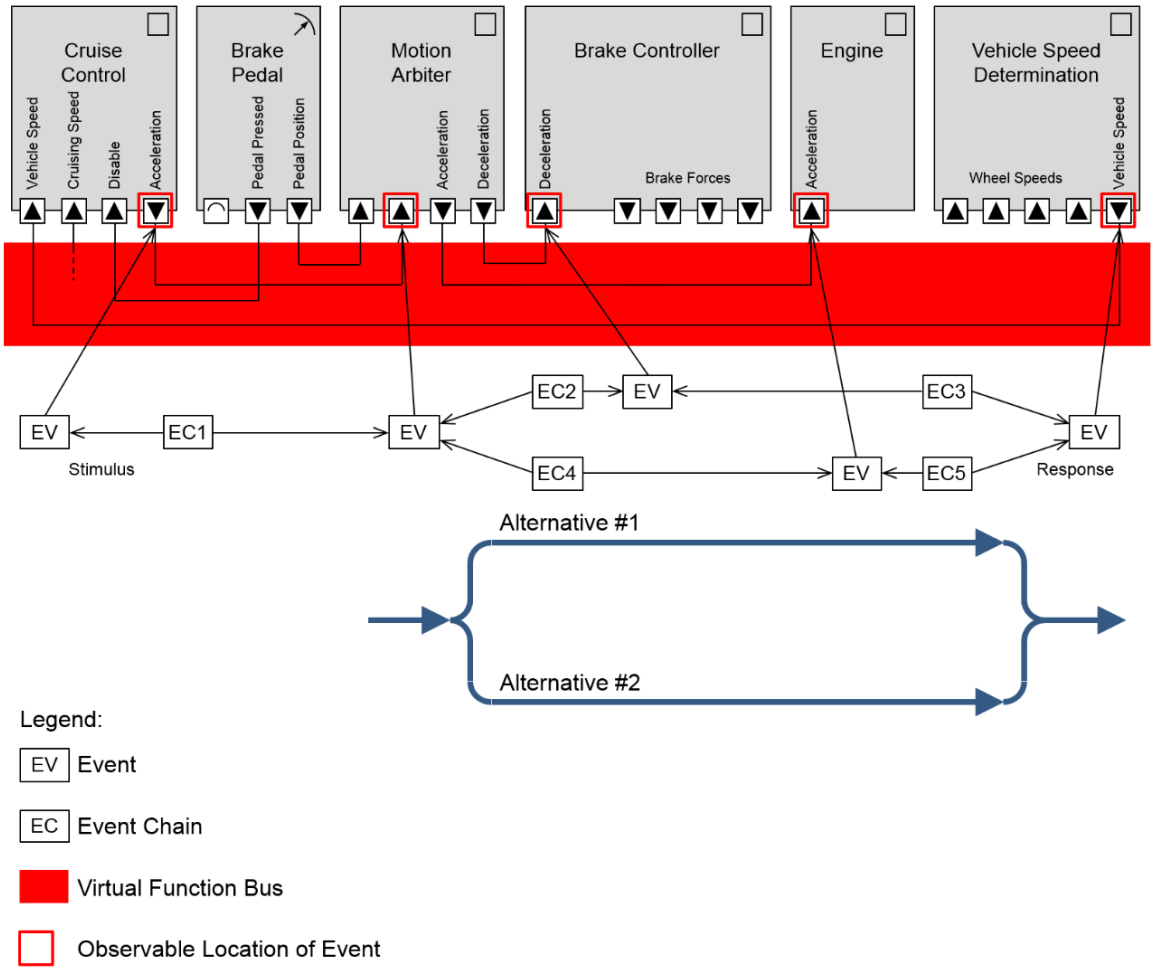


Figure 6.6: Example of the ”Alternative” pattern

6.2.5 Cycle

The ”Cycle” pattern describes the constellation where a path from the response event of an event chain leads to the stimulus of this event chain.

The pattern is illustrated in Figure 6.7 which shows three event chains *EC8*, *EC12* and *EC17* forming a cycle. The stimulus event of event chain *EC8* is the response event of event chain *EC17*; and the response event of event chain *EC12* is the stimulus event of event chain *EC17*. Event chain *EC8* and *EC12* reference the same event in different roles, namely response event from event chain *EC8* perspective and stimulus event from the event chain *EC12* perspective.

Note that an event chain referencing the same event for its stimulus and its response is forbidden according to the constraint [constr_4515]. As a consequence a cycle consists of at least two event chains.

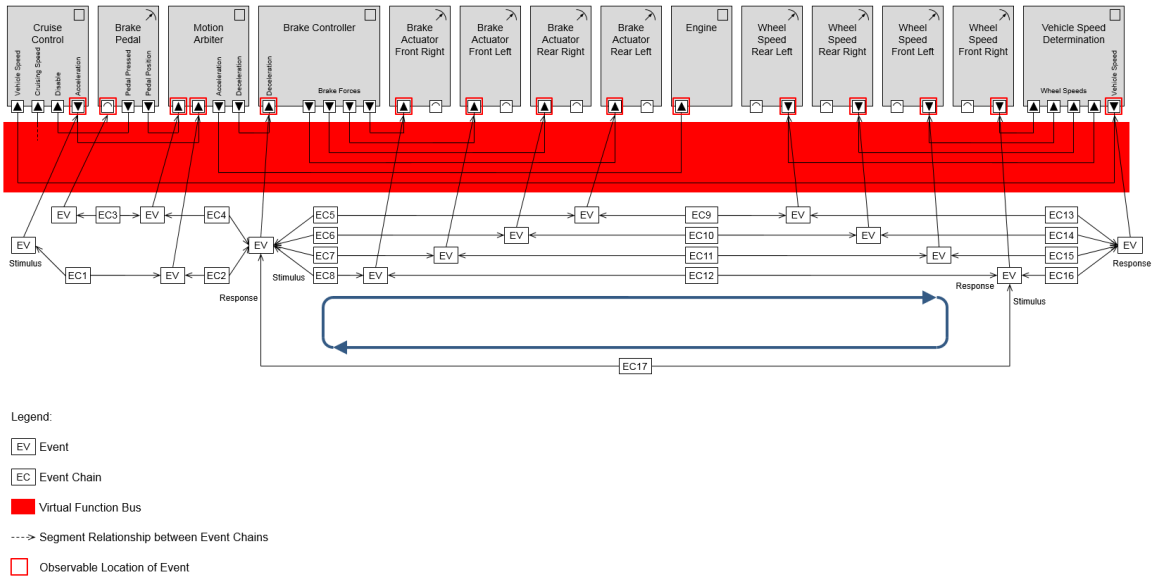


Figure 6.7: Example of the ”Cycle” pattern

7 Timing Constraints

Timing constraints can be applied either on Timing Description Events, on Timing Description Event Chains, or on an ordered list of Executable Entities. Applied to Timing Description Events a Timing Constraint classifies a single event or a group of events with a temporal restriction, for example a period, a latency or a time interval considered as synchronous. Also the direction has to be considered, which means in the semantics of the constraint it matters whether an event source (forward semantics) or an event sink (backward semantics) is considered. Applied to Timing Description Event Chains a condition or property for this event chain is set. As the event chain has a semantic of a directed acyclic graph, the direction is obvious, but it matters whether a single event chain or a group of event chains are constraint. Applied to an ordered list of Executable Entities, a Timing Constraint is a property, which means either the execution order or an execution time.

Mentioned in context of a requirement specification, Timing Constraints can be used as functional requirements and therefore can be tested. For usage in context of a performance specification, Timing Constraints can be used as system properties or timing guarantees. The following table gives an overview over scope and usage of the different types of Timing Constraints described in the following chapters:

Constraint	*Imposed on*	*Use Case*
Event Triggering	`TimingDescriptionEvent`	Specification of an activation Model

Latency Timing	TimingDescriptionEventChain	End-to-End path latency (in reaction or max age semantics)
Age	TimingDescriptionEvent	Restriction
Synchronization Timing	TimingDescriptionEventChain	Restrictions for forks and joins of event chains
Synchronization Timing	TimingDescriptionEvent	Restriction
Offset Timing	TimingDescriptionEvent	Restriction
Execution Order	ExecutableEntity	Restriction
Execution Time	ExecutableEntity	Restriction

Table 7.1: Constraints

7.1 EventTriggeringConstraint

[TPS_TIMEX_00003] **EventTriggeringConstraint specifies occurrence behavior respectively model** ⌈ The element EventTriggeringConstraint is used to specify the particular occurrences of a given timing description event. ⌋ *(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008)*

AUTOSAR offers five basic types of event triggering as depicted in Figure 7.1.

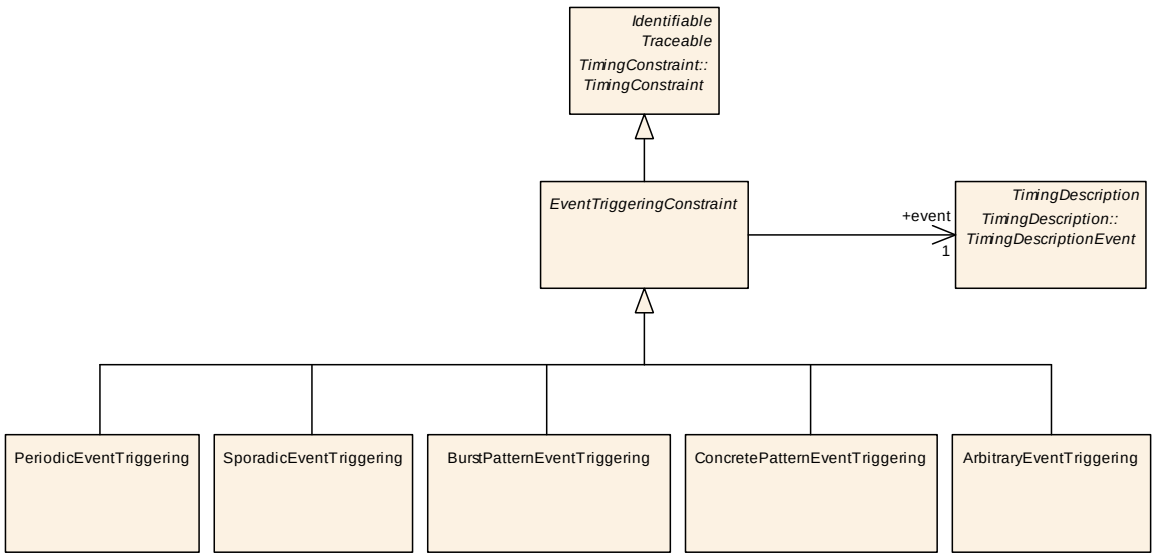


Figure 7.1: The different types of event triggerings

Class	***EventTriggeringConstraint*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	Describes the occurrence behavior of the referenced timing event. The occurrence behavior can only be determined when a mapping from the timing events to the implementation can be obtained. However, such an occurrence behavior can also be described by the modeler as an assumption or as a requirement about the occurrence of the event.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Subclasses	ArbitraryEventTriggering, BurstPatternEventTriggering, ConcretePatternEventTriggering, PeriodicEventTriggering, SporadicEventTriggering			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
event	TimingDescriptionEvent	1	ref	The referenced timing event

Table 7.2: EventTriggeringConstraint

7.1.1 PeriodicEventTriggering

[TPS_TIMEX_00010] `PeriodicEventTriggering` specifies periodic occurrences of events ⌈ The element `PeriodicEventTriggering` is used to specify the characteristics of a timing description event which occurs periodic. ⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008, RS_TIMEX_00015)*

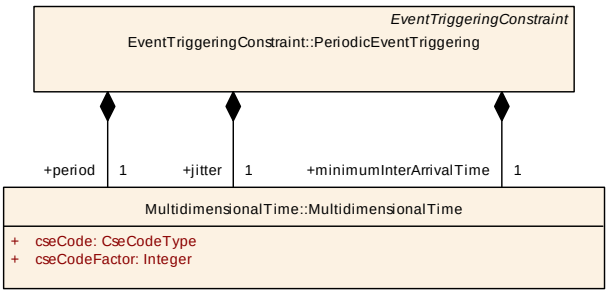


Figure 7.2: PeriodicEventTriggering

Class	**PeriodicEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	The PeriodicEventTriggering describes the behavior of an event with a strict periodic occurrence pattern, given by the period attribute. Additionally, it is possible to soften the strictness of the periodic occurrence behavior by specifying a jitter, so that there can be a deviation from the period up to the size of the jitter.			
Base	*ARObject*, *EventTriggeringConstraint*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
jitter	MultidimensionalTime	1	aggr	The maximum jitter of the periodic event occurrence. **Tags:** xml.sequenceOffset=20
minimumInterArrivalTime	MultidimensionalTime	1	aggr	The minimum time distance between two consecutive occurrences of the associated event. **Tags:** xml.sequenceOffset=10

▽

△

Class	**PeriodicEventTriggering**			
period	MultidimensionalTime	1	aggr	The period of the event occurrence. **Tags:** xml.sequenceOffset=30

Table 7.3: PeriodicEventTriggering

The Periodic Event Triggering is characterized by the following parameters:

- Period
- Jitter
- Minimum Inter-Arrival Time

The listed parameters are required ones and are described in the following.

Period This parameter `period` specifies the periodic distance between subsequent occurrences of the event.

Jitter This parameter `jitter` specifies the maximum deviation from the period.

Minimum Inter-Arrival Time This parameter `minimumInterArrivalTime` specifies the minimum distance between subsequent occurrences of the event. Note, that if the value of the parameter `minimumInterArrivalTime` is less than the value of the parameter `period` minus the value of the parameter `jitter`, then the parameter `minimumInterArrivalTime` has no effect on the properties of the periodic event triggering constraints.

[constr_4543] Maximum value of the parameter `minimumInterArrivalTime` ⌈ The value of the parameter `minimumInterArrivalTime` shall be less than or equal the value of the parameter `period`. ⌋*()*

Let t_n be the point-in-time of the n-th occurrence of the event. A Periodic Event Triggering Constraint is satisfied if, and only if at least one reference point-in-time $t_{reference}$ exists such that for every occurrence of the event at t_n the following holds true: $t_{reference} + (n-1)period \leq t_n \leq t_{reference} + (n-1)period + jitter$ and for all of those event occurrences the minimum distance shall be less than or equal to `minimumInterArrivalTime`.

$$\exists t_{reference} \mid \forall n: \quad t_{reference} + (n-1)period \leq t_n \leq t_{reference} + (n-1)period + jitter$$
$$AND \quad \forall n: \quad t_{n+1} - t_n \leq minimumInterArrivalTime$$

Figure 7.3 illustrates the parameters of the `PeriodicEventTriggering`. The upper part of this figure shows the case that the value of `jitter` is less than the value of the parameter `period`; whereas the lower part of this figure shows the case that the value of `jitter` is greater than or equal the value of the parameter `period`.

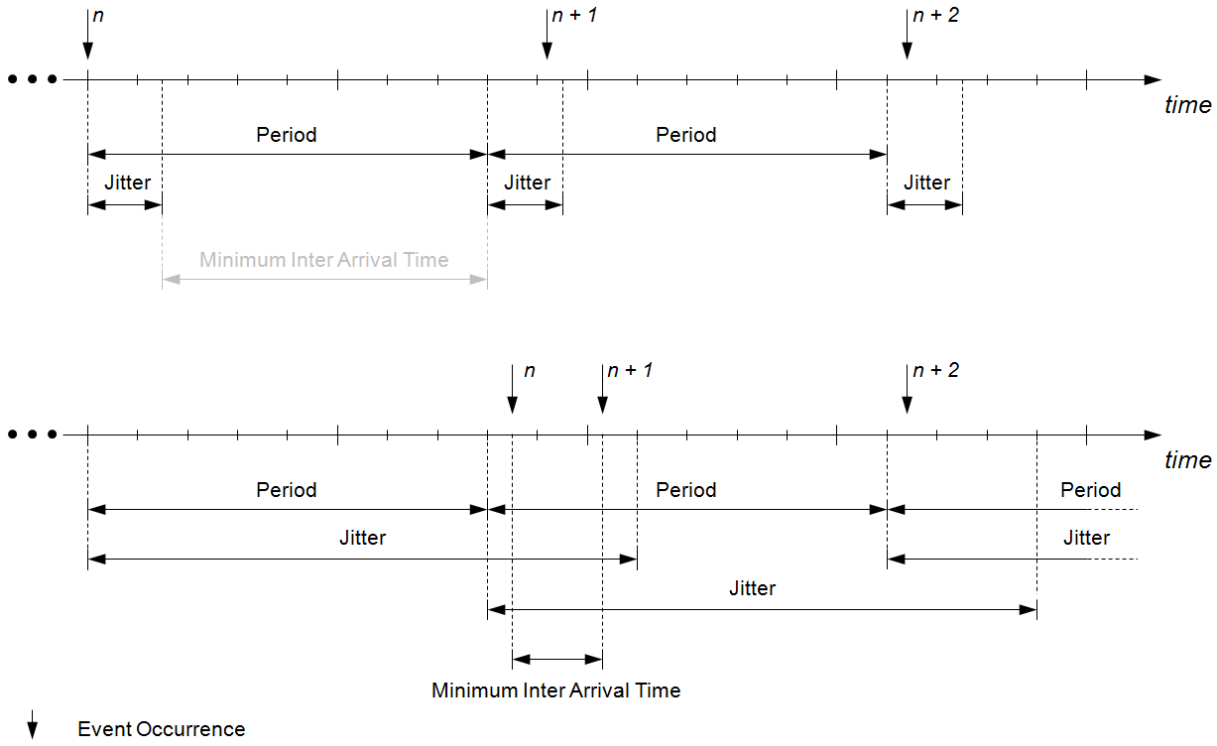


Figure 7.3: Parameters characterizing the Periodic Event Triggering

7.1.1.1 Examples

A Periodic Event Triggering Constraint is specified with the following parameters: `period` is six milliseconds (6ms) and `jitter` is two milliseconds (2ms). In other words, one imposes a timing constraint on an event to occur every six milliseconds and specifies that a deviation of two milliseconds is tolerable. In addition, it is assumed that the `minimumInterArrivalTime` is one millisecond (1ms) and therefore has no impact on the timing of the event's occurrences. This timing constraint is shown in Figure 7.4. The repeating gray-colored rectangles in this figure indicate the time intervals during which the event may occur; in other words it marks the subsequent time intervals the event is expected to occur.

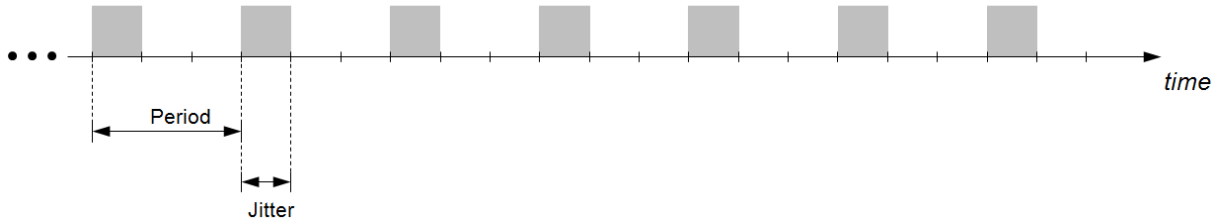


Figure 7.4: Example of a Periodic Event Triggering Constraint

The following figures show various event occurrences recorded during the observation of a system subject to analysis. The time interval for the observation is given by $t_{end-observation} - t_{start-observation}$. In the given example the system is observed for a period of 33.6 milliseconds.

The subsequent event occurrences shown in Figure 7.5 satisfy the given periodic event triggering constraint, because all occurrences of the event observed during the observation time interval happen in their corresponding time interval given by `period` and `jitter`.

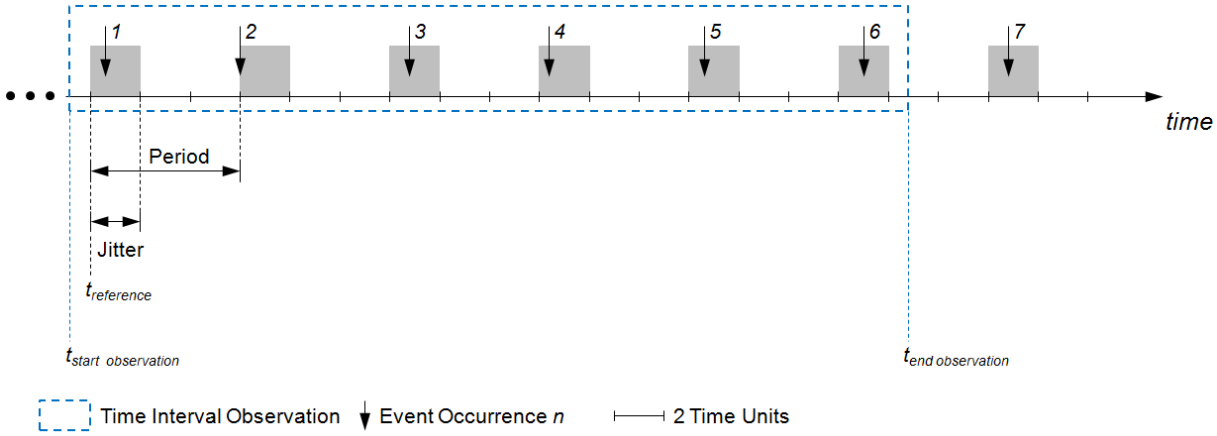


Figure 7.5: Event occurrences satisfying the given Period Event Triggering Constraint shown in the example at the beginning of this subsection.

The subsequent event occurrences shown in Figure 7.6 satisfy the given periodic event triggering constraint, because all occurrences of the event observed during the observation time interval happen in their corresponding time interval given by `period` and `jitter`. In contrast to the example shown in Figure 7.5 the reference point-in-time is another one.

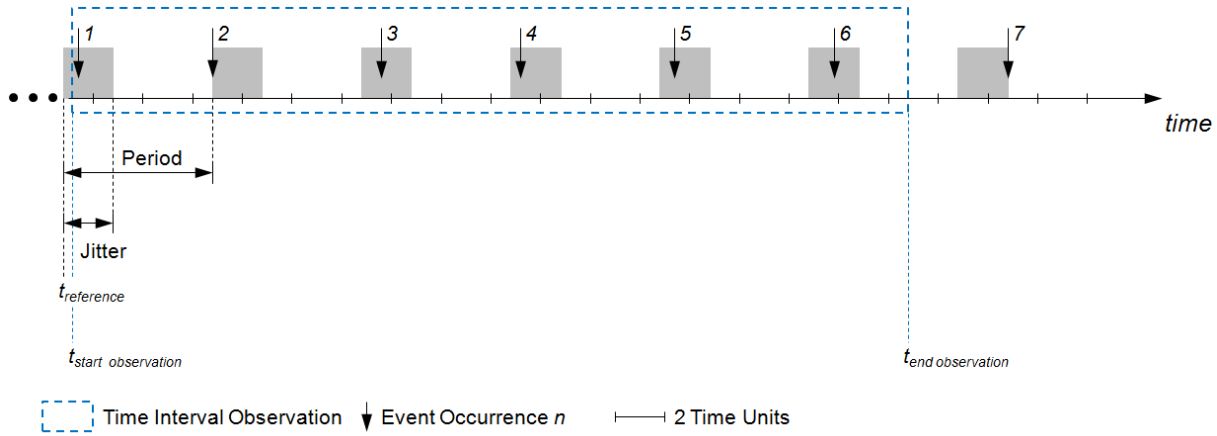


Figure 7.6: Event occurrences satisfying the given Period Event Triggering Constraint shown in the example at the beginning of this subsection, but with another reference point-in-time $t_{reference}$.

The subsequent event occurrences shown in Figure 7.7 violate the given periodic event triggering constraint, because the fifth occurrence of the event does not happen in its corresponding time interval given by `period` and `jitter`. In other words, there does not exist a reference point-in-time that ensures that all occurrences of the event observed during the observation time interval happen in their corresponding time interval given by `period` and `jitter`. And this results in a violation of the parameters `period` and `jitter`.

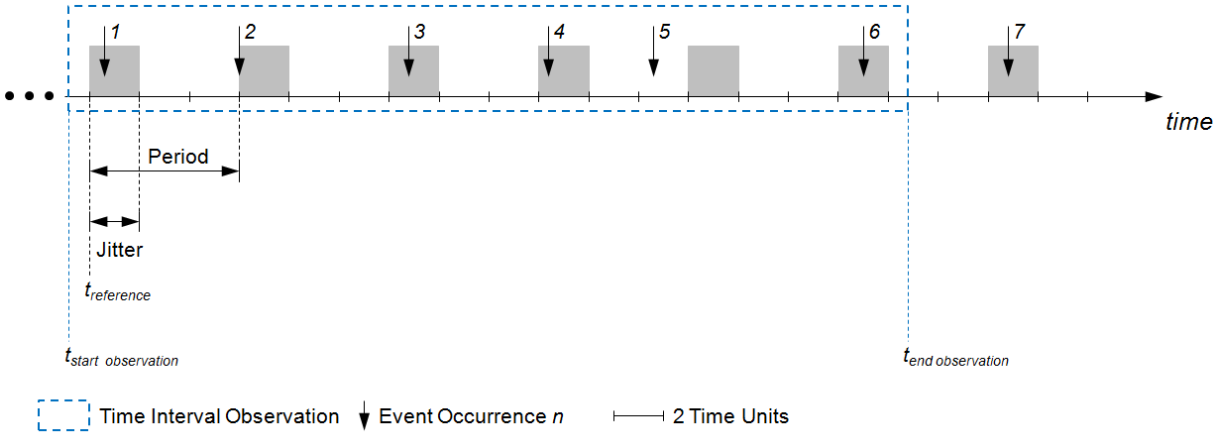


Figure 7.7: Event occurrences violating the given Period Event Triggering Constraint shown in the example at the beginning of this subsection.

The subsequent event occurrences shown in Figure 7.8 violate the given periodic event triggering constraint, because the fourth occurrence of the event does not happen in its corresponding time interval given by `period` and `jitter`. In other words, the fourth occurrence of the event happens in the time interval the fifth occurrence of the event happens and therefore violates the specified `jitter`.

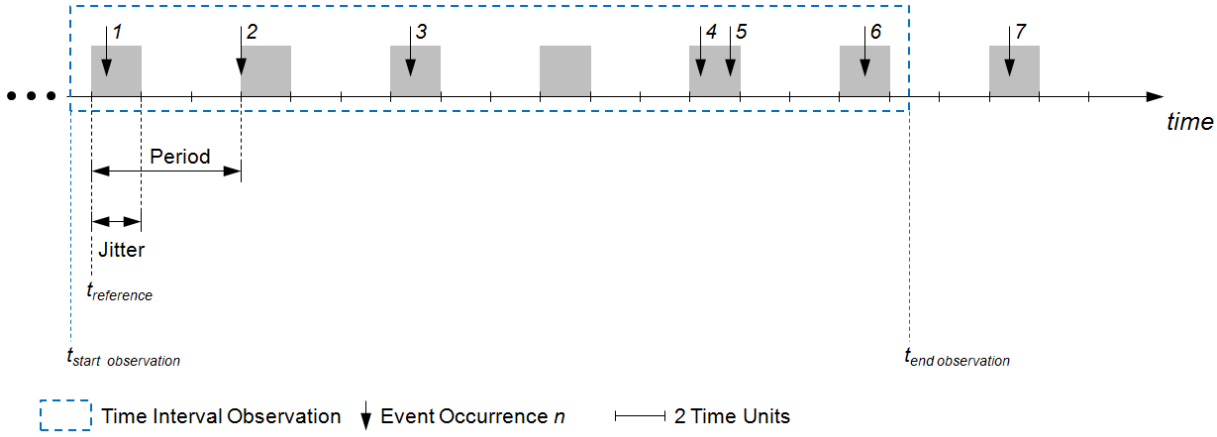


Figure 7.8: Event occurrences satisfying the given Period Event Triggering Constraint shown in the example at the beginning of this subsection.

7.1.2 SporadicEventTriggering

[TPS_TIMEX_00011] `SporadicEventTriggering` specifies sporadic occurrences of events ⌈ The element `SporadicEventTriggering` is used to specify the characteristics of a timing description event which occurs sporadic. ⌋ *(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008)*

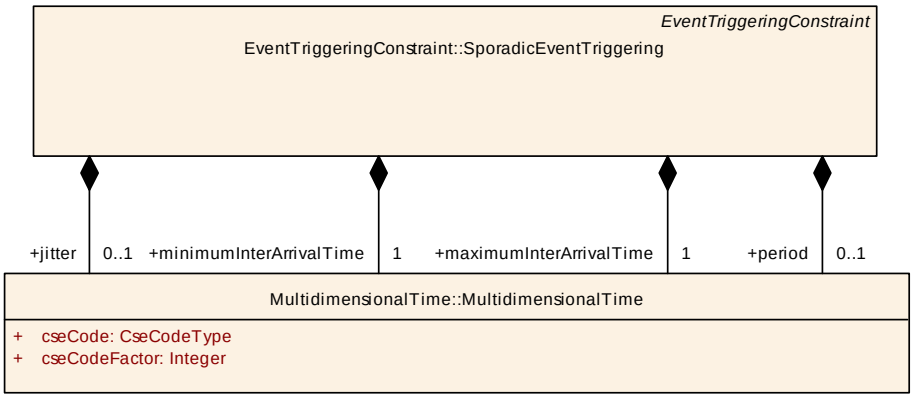


Figure 7.9: SporadicEventTriggering

Class	**SporadicEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	The SporadicEventTriggering describes the behavior of an event which occurs occasionally or singly.			
Base	*ARObject*, *EventTriggeringConstraint*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
jitter	MultidimensionalTime	0..1	aggr	The maximum jitter of the sporadic event occurrence. Jitter=max |nthPeriod - standardPeriod| **Tags:** xml.sequenceOffset=30
maximumInterArrivalTime	MultidimensionalTime	1	aggr	The maximum time distance between two consecutive occurrences of the associated event. **Tags:** xml.sequenceOffset=20
minimumInterArrivalTime	MultidimensionalTime	1	aggr	The minimum time distance between two consecutive occurrences of the associated event. **Tags:** xml.sequenceOffset=10
period	MultidimensionalTime	0..1	aggr	The period of the event occurrence. **Tags:** xml.sequenceOffset=40

Table 7.4: SporadicEventTriggering

This is a generalization of the periodic event triggering described in subsection 7.1.1. The difference is that the event can, but not necessarily must occur. For this reason, there is one additional parameter required for the specification of the `SporadicEventTriggering`, namely the `maximumInterArrivalTime`, which specifies the largest possible time distance between two event occurrences.

The Sporadic Event Triggering is characterized by the following parameters:

- Minimum Inter-Arrival Time

- Maximum Inter-Arrival Time
- Period
- Jitter

The first two parameters are required ones and the last two parameters are optional. These parameters are described in the following and Figure 7.10 illustrates the parameters of the `SporadicEventTriggering`.

Minimum Inter-Arrival Time This parameter `minimumInterArrivalTime` specifies the minimum distance between subsequent occurrences of the event.

Maximum Inter-Arrival Time This parameter `maximumInterArrivalTime` specifies the maximum distance between subsequent occurrences of the event.

Period This optional parameter `period` specifies the periodic distance between subsequent occurrences of the event.

Jitter This optional parameter `jitter` specifies the maximum deviation from the period.

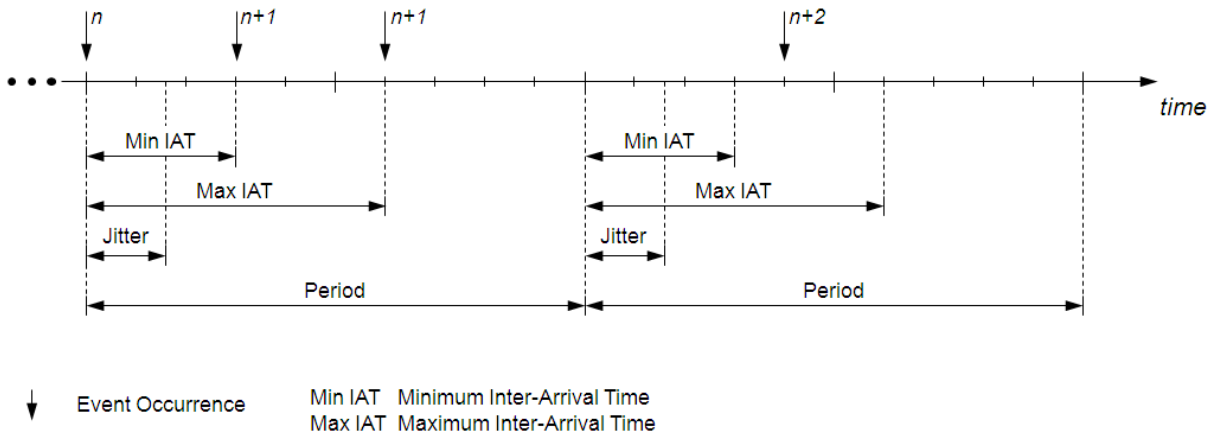


Figure 7.10: Parameters characterizing the Sporadic Event Triggering

7.1.3 ConcretePatternEventTriggering

[TPS_TIMEX_00012] `ConcretePatternEventTriggering` specifies concrete pattern of occurrences of events ⌈ The element `ConcretePatternEventTriggering` is used to specify the characteristics of a timing description event which occurs as a concrete pattern. ⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008)*

This describes events which occur following a known pattern.

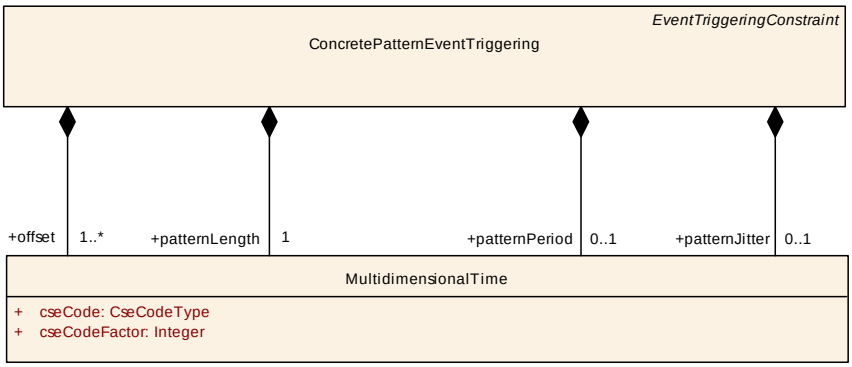


Figure 7.11: ConcretePatternEventTriggering

Class	**ConcretePatternEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	The ConcretePatternEventTriggering describes the behavior of an event, which occurs following a precisely known pattern.			
Base	*ARObject*, *EventTriggeringConstraint*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
offset	MultidimensionalTime	1..*	aggr	The offset for each occurrence of the event in the specified time interval. **Tags:** xml.name=TIME-VALUE xml.roleElement=true xml.sequenceOffset=10 xml.typeElement=false
patternJitter	MultidimensionalTime	0..1	aggr	The optional parameter "Pattern Jitter" specifies the deviation of the time interval's starting point from the beginning of the given period. This parameter is only applicable in conjunction with the parameter "Pattern Period".
patternLength	MultidimensionalTime	1	aggr	The length of the observed time interval. **Tags:** xml.sequenceOffset=20
patternPeriod	MultidimensionalTime	0..1	aggr	The optional parameter "Pattern Period" specifies the time distance between the beginnings of subsequent repetitions of the given concrete pattern.

Table 7.5: ConcretePatternEventTriggering

The Concrete Pattern Event Triggering is characterized by the following parameters:

- Pattern Length
- Offset
- Pattern Period
- Pattern Jitter

The first two parameters are required ones, whereas the two last parameters are optional. The parameters are described in the following and are illustrated in Figure 7.12 and Figure 7.13.

Pattern Length This parameter `patternLength` specifies the time interval the pattern occurs in.

Offset This parameter `offset` specifies a list of point-in-times in the time interval given by the parameter `patternLength` at which the event occurs.

Pattern Period This optional parameter `patternPeriod` specifies the time distance between the beginnings of subsequent repetitions of the given burst pattern.

Pattern Jitter This optional parameter `patternJitter` specifies the maximum deviation of the time interval's starting point from the beginning of the given period. This parameter is only applicable in conjunction with the parameter `patternPeriod`.

The constraints listed below apply to the `ConcretePatternEventTriggering` and shall be considered when using this event triggering constraint.

[constr_4519] Specifying `patternLength` ⌈ The `patternLength` shall be specified such that the following holds: $0 \leq max(\texttt{offset}) \leq \texttt{patternLength}$. ⌋*()*

[constr_4544] Specifying `patternLength`, `patternJitter` and `patternPeriod` ⌈ The pattern length, pattern jitter and pattern period shall be specified such that the following holds: $\texttt{patternLength} + \texttt{patternJitter} < \texttt{patternPeriod}$. ⌋ *()*

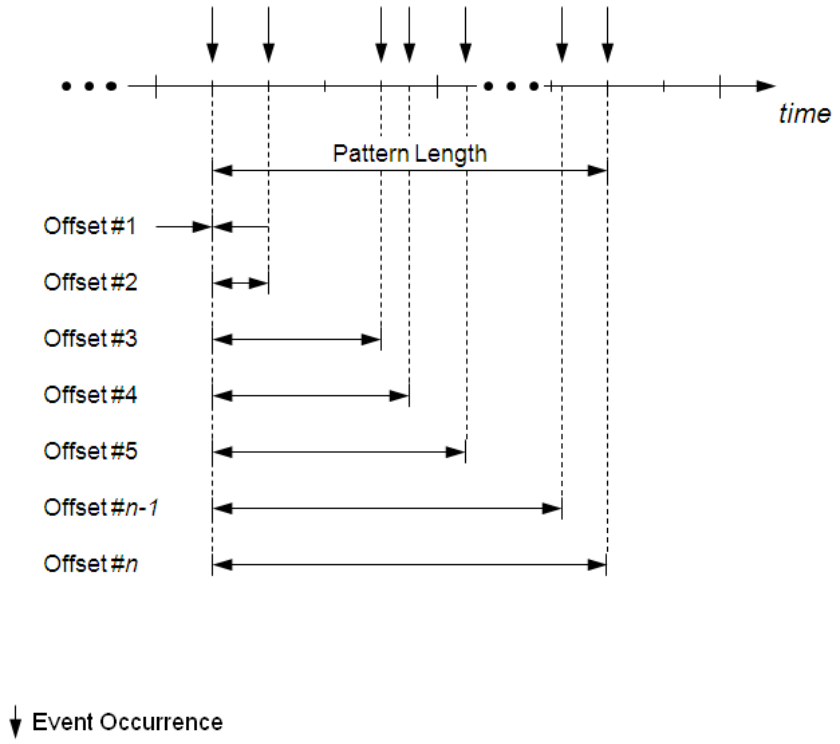


Figure 7.12: Parameters characterizing the Concrete Pattern Event Triggering

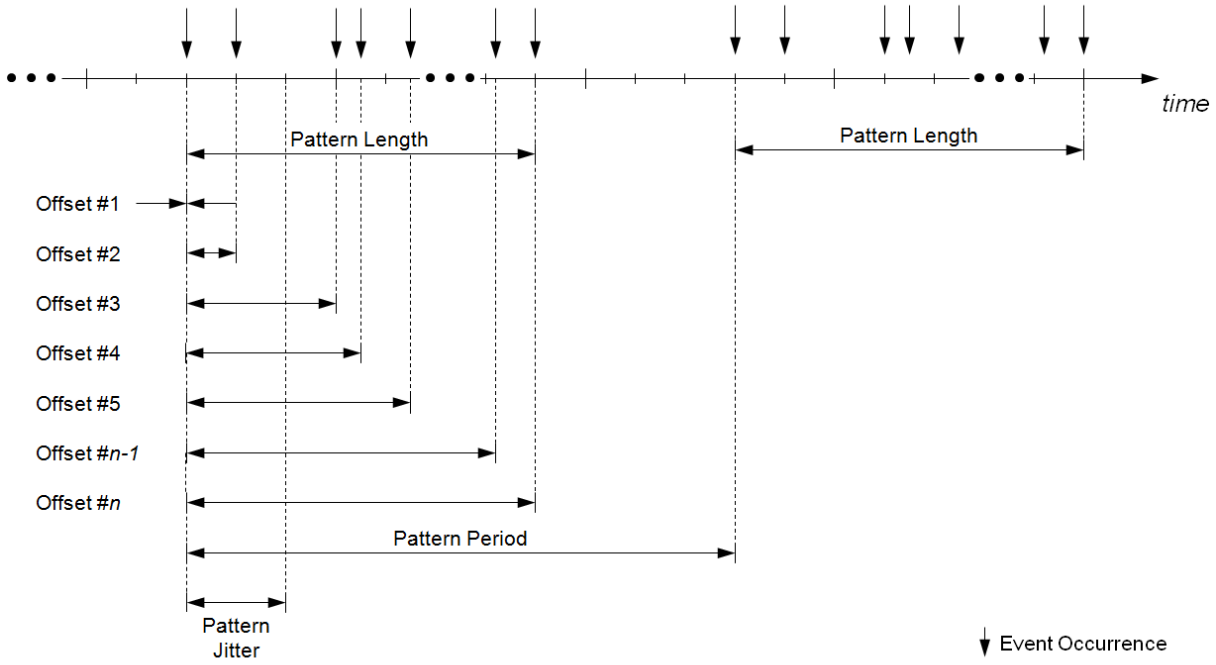


Figure 7.13: Parameters characterizing the Concrete Pattern Event Triggering when periodically being repeated

7.1.4 BurstPatternEventTriggering

[TPS_TIMEX_00013] `BurstPatternEventTriggering` specifies burst of occurrences of events ⌈ The element `BurstPatternEventTriggering` is used to specify the characteristics of a timing description event which occurs as a burst. ⌋ *(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008)*

The purpose of the `BurstPatternEventTriggering` is to describe a burst of occurrences of one and the same event. The Burst Pattern Event Triggering is characterized by the following parameters:

- Pattern Length
- Minimum Inter Arrival Time
- Maximum Number of Occurrences
- Minimum Number of Occurrences
- Pattern Period
- Pattern Jitter

The first three parameters are required ones, whereas the last three parameters are optional.

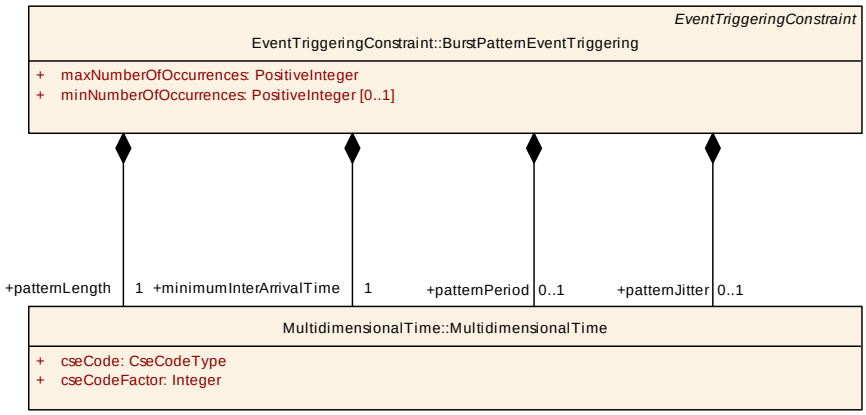


Figure 7.14: BurstPatternEventTriggering

Class	**BurstPatternEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	A BurstPatternEventTriggering describes the maximum number of occurrences of the same event in a given time interval. This is typically used to model a worst case activation scenario.			
Base	*ARObject*, *EventTriggeringConstraint*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Timing Constraint*, *Traceable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
maxNumberOf Occurrences	PositiveInteger	1	attr	The maximum number of event occurrences within the given time interval.
minNumberOf Occurrences	PositiveInteger	0..1	attr	The minimum number of event occurrences within the given time interval. **Tags:** xml.sequenceOffset=10
minimumInter ArrivalTime	MultidimensionalTime	1	aggr	The parameter "Minimum Inter-Arrival Time" specifies the minimum distance between subsequent occurrences of the event within the given time interval.
patternJitter	MultidimensionalTime	0..1	aggr	The optional parameter "Pattern Jitter" specifies the deviation of the time interval's starting point from the beginning of the given period. This parameter is only applicable in conjunction with the parameter "Pattern Period".
patternLength	MultidimensionalTime	1	aggr	The parameter "Pattern Length" specifies the duration of the time interval within which the event repeatedly occurs. The event occurs at arbitrary points in time within the given time interval.
patternPeriod	MultidimensionalTime	0..1	aggr	The optional parameter "Pattern Period" specifies the time distance between the beginnings of subsequent repetitions of the given burst pattern.

Table 7.6: BurstPatternEventTriggering

The parameters are described in the following and are illustrated in Figure 7.15 and Figure 7.16.

Pattern Length This parameter `patternLength` specifies the duration of the time interval within which the event repeatedly occurs. The event occurs at arbitrary points in time within the given time interval.

Minimum Inter-Arrival Time This parameter `minimumInterArrivalTime` specifies the minimum distance between subsequent occurrences of the event within the given time interval.

Maximum Number of Occurrences This parameter `maxNumberOfOccurrences` specifies the maximum number of times the event can occur within the time interval. In other words, the event may never occur or any number of times between one (1) and the specified maximum number of occurrences. If the parameter `minNumberOfOccurrences` is specified then the event occurs at least the number of times specified by `minNumberOfOccurrences` and at maximum by `maxNumberOfOccurrences`.

Minimum Number of Occurrences This optional parameter `minNumberOfOccurrences` specifies the minimum number of times the event occurs within the given time interval. In other words, this parameter specifies the least number of times the event occurs in the given time interval. The value zero (0) for this parameter is permitted.

Pattern Period This optional parameter `patternPeriod` specifies the time distance between the beginnings of subsequent repetitions of the given burst pattern.

Pattern Jitter This optional parameter `patternJitter` specifies the maximum deviation of the time interval's starting point from the beginning of the given period. This parameter is only applicable in conjunction with the parameter `patternPeriod`.

The constraints listed below apply to the `BurstPatternEventTriggering` and shall be considered when using this event triggering constraint.

[constr_4505] Specifying minimum and maximum number of occurrences ⌈ The minimum and maximum number of occurrences shall be specified such that the following holds: $0 \leq$ `minNumberOfOccurrences` $\leq$ `maxNumberOfOccurrences`. ⌋ *()*

[constr_4506] Specifying minimum inter-arrival time and pattern length ⌈ The minimum inter-arrival time and pattern length shall be specified such that the following holds: $0 <$ `minimumInterArrivalTime` $\leq$ `patternLength`. ⌋*()*

[constr_4507] Specifying pattern length, pattern jitter and patter period ⌈ The pattern length, pattern jitter and pattern period shall be specified such that the following holds: `patternLength` $+$ `patternJitter` $<$ `patternPeriod`. ⌋*()*

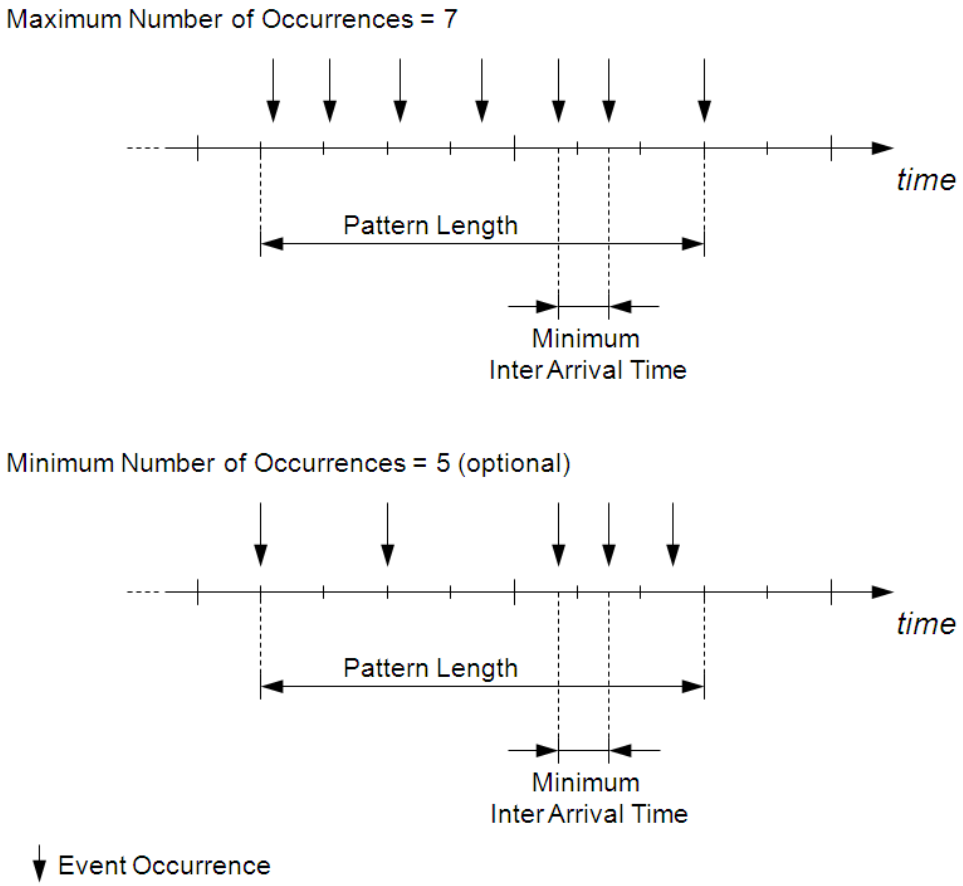


Figure 7.15: Parameters characterizing the Burst Pattern Event Triggering

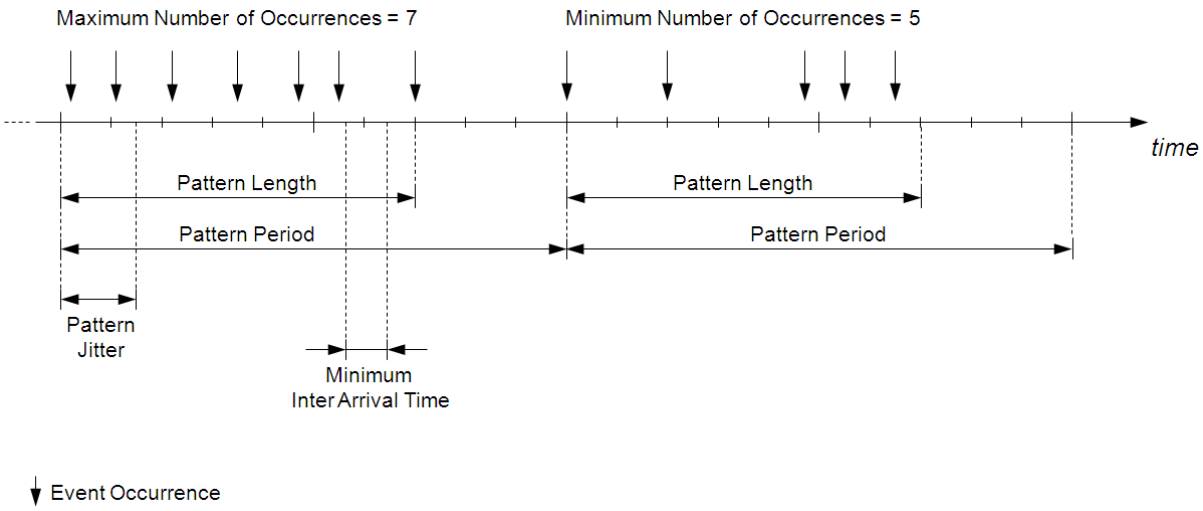


Figure 7.16: Parameters characterizing the Burst Pattern Event Triggering when periodically being repeated

7.1.5 ArbitraryEventTriggering

[TPS_TIMEX_00014] `ArbitraryEventTriggering` specifies arbitrary occurrences of an event ⌈ The element `ArbitraryEventTriggering` is used to specify the characteristics of a timing description event which occurs arbitrary. ⌋ *(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008)*

This describes the occasional occurrence of a timing event.

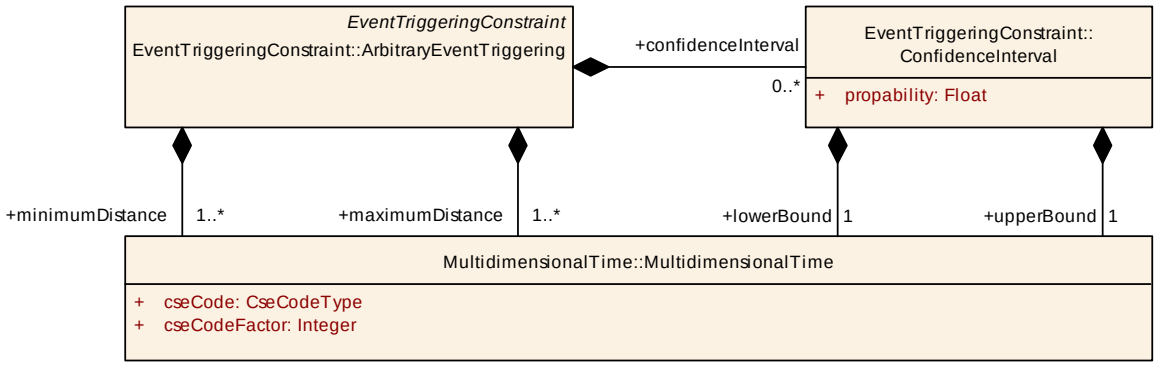


Figure 7.17: ArbitraryEventTriggering

Class	**ArbitraryEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	The ArbitraryEventTriggering describes that an event occurs occasionally, singly, irregularly or randomly. The primary purpose of this event triggering is to abstract event occurrences captured by data acquisition tools (background debugger, trace analyzer, etc.) during system runtime.			
Base	*ARObject*, *EventTriggeringConstraint*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
confidence Interval	ConfidenceInterval	*	aggr	List of confidence intervals. **Tags:** xml.sequenceOffset=30
maximum Distance	MultidimensionalTime	1..*	aggr	The nth array element describes the maximum distance that can be observed for a sample of n+1 event occurrences. This is an array with an identical number of elements as for the minimumDistance. **Tags:** xml.name=TIME-VALUE xml.roleElement=true xml.sequenceOffset=20 xml.typeElement=false
minimum Distance	MultidimensionalTime	1..*	aggr	The nth array element describes the minimum distance that can be observed for a sample of n+1 event occurrences. This is an array with an identical number of elements as for the maximumDistance. ▽

▽

△

Class	**ArbitraryEventTriggering**			
				△ **Tags:** xml.name=TIME-VALUE xml.roleElement=true xml.sequenceOffset=10 xml.typeElement=false

Table 7.7: ArbitraryEventTriggering

Class	**ConfidenceInterval**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	Additionally to the list of measured distances of event occurrences, a confidence interval can be specified for the expected distance of two consecutive event occurrences with a given probability.			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
lowerBound	MultidimensionalTime	1	aggr	The lower bound of the expected distance of two consecutive event occurrences.
propability	Float	1	attr	The probability for the measured lower and upper bound of the confidence interval.
upperBound	MultidimensionalTime	1	aggr	The upper bound of the expected distance of two consecutive event occurrences.

Table 7.8: ConfidenceInterval

In contrast to the `ConcretePatternEventTriggering`, this event triggering is not as strict to the occurrence of an event, but generally describes event occurrences.

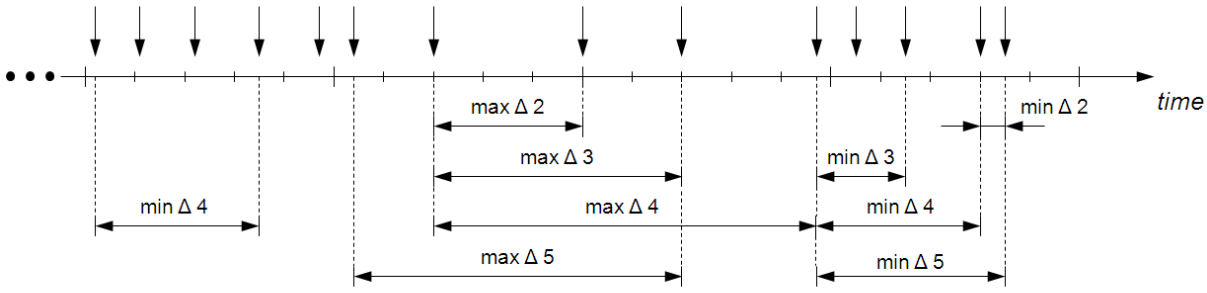
The Arbitrary Event Triggering is characterized by the following parameters:

- Minimum Distance
- Maximum Distance

These parameters are required ones and are described in the following. Figure 7.18 illustrates the parameters of the `ArbitraryEventTriggering`.

Minimum Distance The parameter `minimumDistance` specifies the minimum distance between n subsequent event occurrences, and $n = 2, 3, 4, ...$

Maximum Distance The parameter `maximumDistance` specifies the maximum distance between n subsequent event occurrences, and $n = 2, 3, 4, ...$



min Δn Least minimum inter-arrival time between *n* subsequent occurrences of the event E and n = {2, 3, 4, ...}

max Δn Major maximum inter-arrival time between *n* subsequent occurrences of the event E and n = {2, 3, 4, ...}

Event Occurrence

Figure 7.18: Parasmeters characterizing the Arbitrary Event Triggering

7.2 LatencyTimingConstraint

[TPS_TIMEX_00004] `LatencyTimingConstraint` specifies latency constraints
⌈ The element `LatencyTimingConstraint`[1] is used to specify the amount of time that elapses between the occurrence of any two timing description events. ⌋
(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00012, RS_TIMEX_00015)

For example, this can be the time it takes for a packet of data on a bus network to get from one designated point to another, or the time it takes for a task to be executed on a processor.

In the timing specification a `LatencyTimingConstraint` is associated with one `TimingDescriptionEventChain`, and specifies the minimum and/or maximum time duration between the occurrence of the stimulus and the occurrence of the corresponding response of that chain. However, in multi-rate networks, data can get lost or get duplicated because of potential different producer and consumer periods. Data loss occurs, if the consumer's period is greater than the producer's period (undersampling). Accordingly, data duplication occurs, if the consumer's period is smaller than the producer's period (oversampling). This is depicted in figure 7.19.

[1] A synonym for delay

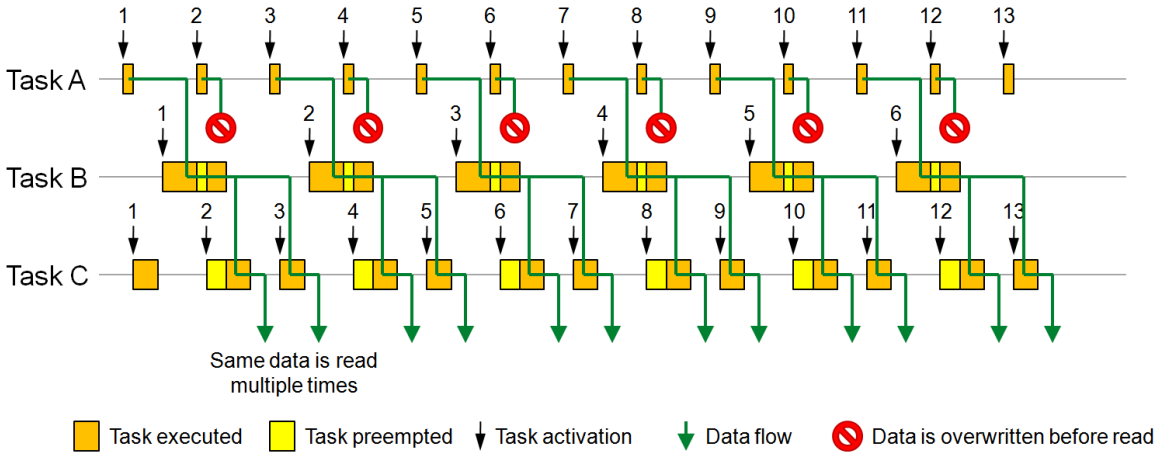


Figure 7.19: Loss and duplication of data due to under- and oversampling.

Considering under- and oversampling, two end-to-end latency semantics are of interest for automotive systems and can thus be expressed with the AUTOSAR timing extensions. These are the *age* of a certain response and the *reaction* to a certain stimulus.

The *data age timing constraint* is mainly important in control engineering, but may appear in all domains. Here the focus is from the response perspective rather than from the stimulus perspective. In other words, the assumption is that last is best, i.e., it is accepted/tolerated that a value is overwritten along the path from stimulus to response. When for example an actuator value is periodically updated, it is of importance that the corresponding input values are not too old. In this case the constrained time of importance is the delay from the latest stimulus to a given response.

The *reaction time constraint* is utilized when the first reaction to a stimulus is of importance. This is usually the case in body electronics, but may also be the case in other domains. One example is the time it takes from a button is pressed to the light is switched on. Another example, from the chassis domain, is the time from the brake pedal is pressed until the brakes are activated. In both cases the constrained time of importance is the delay from a given stimulus to the first corresponding response.

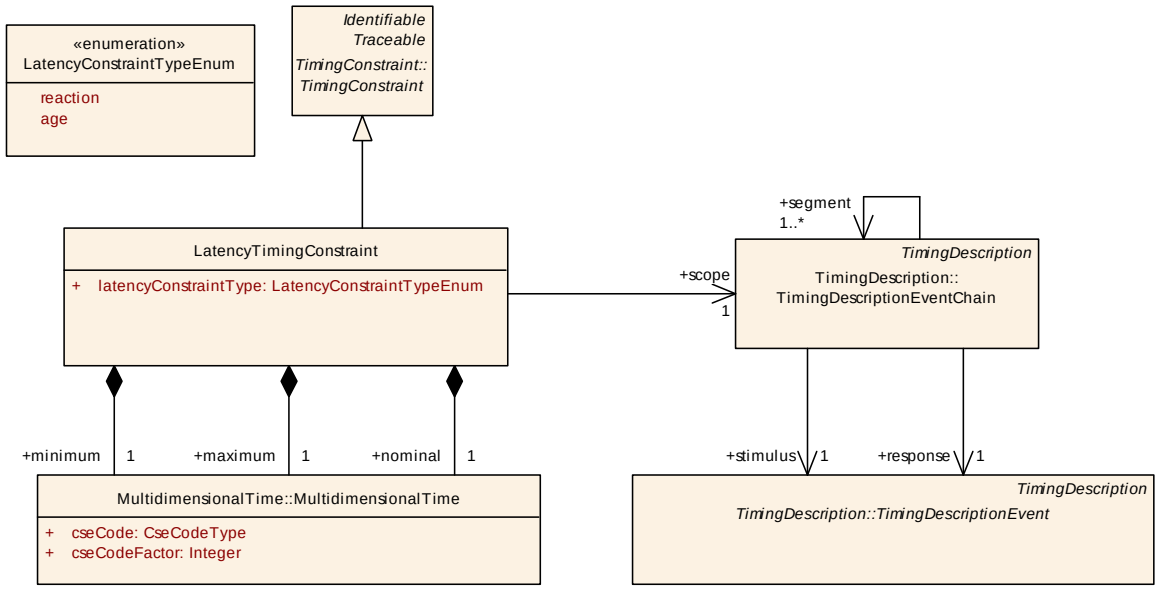


Figure 7.20: Latency constraint

Class	**LatencyTimingConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::LatencyTimingConstraint			
Note	This constraint type restricts the time duration between the occurrence of the stimulus and the occurrence of the corresponding response of that chain. Two latency constraint types are of interest for automotive systems. These are the age of a certain response and the reaction to a certain stimulus. In contrast to OffsetTimingConstraint, a causal dependency between the stimulus and response event of the associated event chain is required.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
latency ConstraintType	LatencyConstraintType Enum	1	attr	The specific type of this latency constraint
maximum	MultidimensionalTime	1	aggr	The maximum latency between the occurrence of the stimulus and the occureence of the corresponding response of the associatied event chain. **Tags:** xml.sequenceOffset=20
minimum	MultidimensionalTime	1	aggr	The minimum latency between the occurrence of the stimulus and the occureence of the corresponding response of the associatied event chain. **Tags:** xml.sequenceOffset=10
nominal	MultidimensionalTime	1	aggr	The nominal latency between the occurrence of the stimulus and the occureence of the corresponding response of the associatied event chain. **Tags:** xml.sequenceOffset=30
scope	TimingDescriptionEvent Chain	1	ref	The event chain that defines the scope of the constraint.

Table 7.9: LatencyTimingConstraint

Enumeration	**LatencyConstraintTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::LatencyTimingConstraint
Note	This is used to describe the type of the latency timing constraint.
Literal	***Description***
age	In this case, the latency constraint is seen from the perspective of the response event of the associated event chain. Given a certain response event, the age interval of the latest stimulus is constrained. **Tags:** atp.EnumerationValue=0
reaction	In this case, the latency constraint is seen from the perspective of the stimulus event of the associated event chain. Given a certain stimulus event, the reaction interval of the first response is constrained. **Tags:** atp.EnumerationValue=1

Table 7.10: LatencyConstraintTypeEnum

The attributes `minimum`, `maximum`, and `nominal` of a `LatencyTimingConstraint` can be used to define a lower and upper bound, as well as a nominal value for the latency of the event chain in the scope.

The application of latency constraints leads to some interesting observations:

- In systems without over- and undersampling, *age* and *reaction* are the same. But timing constraints are implementation-independent. Thus, at specification time when the implementation is not necessarily known, the correct latency constraint semantics has to be specified.
- The minimum reaction and the minimum age latency of an event chain are always equal.

7.3 AgeConstraint

Sometimes it is necessary to specify the age of data, when it arrives at a component on its required port with `SenderReceiverInterface`. If the sender of the data is known, a `TimingDescriptionEventChain` can be defined from the sender to the receiver port and a `LatencyTimingConstraint` with *age* semantic represents the specification of the data age. However, the actual sender of the data may be unknown. In this case the definition of a `TimingDescriptionEventChain` is not possible.

[TPS_TIMEX_00005] `AgeConstraint` to specify age constraints ⌈ The element `AgeConstraint` is used to specify a minimum and maximum age that is tolerated when a variable data prototypes is received. ⌋*(RS_TIMEX_00001, RS_TIMEX_00009)*

Instead of an event chain, the scope of an age constraint is a `TDEventVariableDataPrototype`. Every time the scoped event occurs, the `VariableDataPrototype` shall have the specified data age.

At a later stage during the development, when the refined software architecture exposes the relation between the actual sender of the data and the receiver, an event chain between the sending and receiving point in time shall be defined and associated

with a LatencyTimingConstraint (see 7.2) in order to refine the previous defined age constraint.

Typically, the age constraint restricts the time interval between the physical creation of the original sensor data by the corresponding sensor hardware and the availability of the data in the communication buffer (of the RTE) of the receiving SWC.

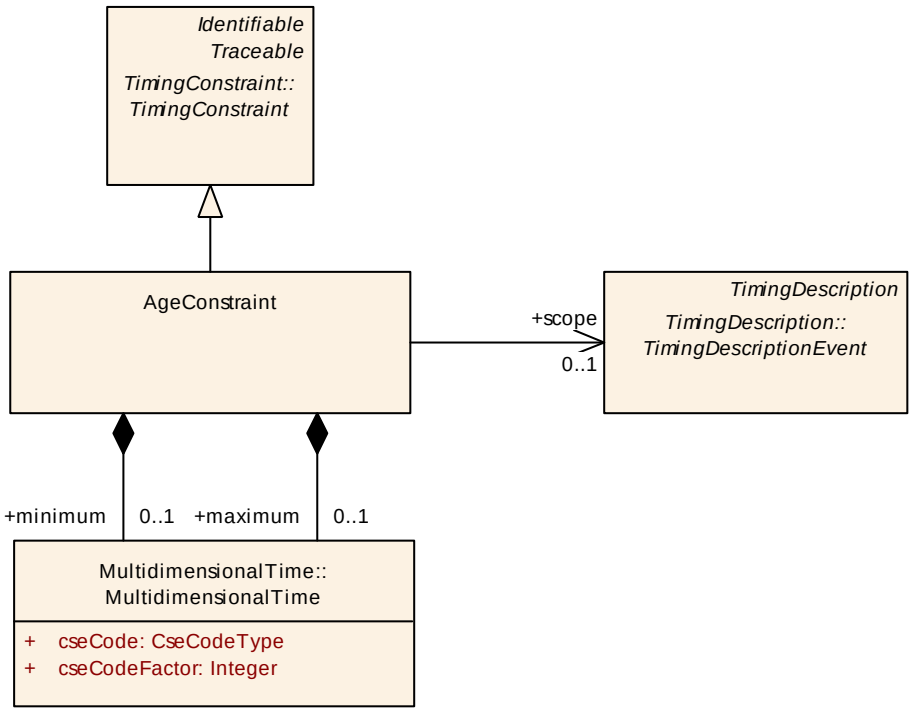


Figure 7.21: Age constraint

An AgeConstraint can define a minimum and maximum age for the VariableDataPrototype referenced by the TDEventVariableDataPrototype scope.

[constr_4504] Restricted usage of AgeConstraint ⌈ An AgeConstraint shall only be defined for events of type TimingDescriptionEvent associated with the receipt and reading of data. ⌋()

Class	**AgeConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::AgeConstraint			
Note	The AgeConstraint is used to impose a constraint on an Timing Description Event referenced by the scope. A minimum and a maximum age can be specified.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
maximum	MultidimensionalTime	0..1	aggr	The maximum age.
minimum	MultidimensionalTime	0..1	aggr	The minimum age.
scope	TimingDescriptionEvent	0..1	ref	The scope of an AgeConstraint is any TimingDescription Event that indicates any receipt of data.

Table 7.11: AgeConstraint

7.4 SynchronizationTimingConstraint

The objective of synchronization in a distributed environment is to establish and maintain a consistent time base for the interaction between different subsystems, in order to obtain correct runtime order and avoid unexpected race conditions. While mechanisms to establish synchronization need to be provided at the implementation level, the necessity for synchronization needs to be expressed at design level. For this purpose, synchronization constraints are used.

[TPS_TIMEX_00006] `SynchronizationTimingConstraint` **specifies synchronicity constraints** ⌈ The element `SynchronizationTimingConstraint` is used to specify a synchronization constraint among the occurrences of two or more timing description events. ⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00007, RS_TIMEX_00008, RS_TIMEX_00017)*

A `SynchronizationTimingConstraint` is imposed either on events (7.4.2) or on event chains (7.4.1).

Class	**SynchronizationTimingConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::SynchronizationTiming Constraint			
Note	This constraint is used to restrict the timing behavior of different, but correlated events or event chains, with regard to synchronization. Thereby, in case of imposing a synchronization timing constraint on events or event chains the following two scenarios are supported: 1) [synchronizationConstraintType=responseSynchronization] Events: An arbitrary number of correlated events which play the role of responses shall occur synchronously with respect to a predefined tolerance. Event Chains: An arbitrary number of correlated event chains with a common stimulus, but different responses, where the responses shall occur synchronously with respect to a predefined tolerance. 2) [synchronizationConstraintType=stimulusSynchronization] Events: An arbitrary number of correlated events which play the role of stimuli shall occur synchronously with respect to a predefined tolerance. Event Chains: An arbitrary number of correlated event chains with a common response, but different stimuli, where the stimuli shall occur synchronously with respect to a predefined tolerance. In case of imposing a synchronization timing constraint on events the following two scenarios are supported: 1) [eventOccurrenceKind=singleOccurrence] Any of the events shall occur only once in the given time interval. 2) [eventOccurrenceKind=multipleOccurrences] Any of the events may occur more than once in the given time interval. In other words multiple occurrences of an event within the given time interval are permitted.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
event OccurrenceKind	EventOccurrenceKind Enum	0..1	attr	The specific occurrence kind of an event occurring within the given time interval.
scope	TimingDescriptionEvent Chain	*	ref	The event chains that are in the scope of the constraint.
scopeEvent	TimingDescriptionEvent	*	ref	The events that are in the scope of the constraint.
synchronization ConstraintType	SynchronizationType Enum	1	attr	The specific type of this synchronization constraint.
tolerance	MultidimensionalTime	1	aggr	The maximum time interval, within which the synchronized events must occur.

Table 7.12: SynchronizationTimingConstraint

Enumeration	**EventOccurrenceKindEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint
Note	This is used to describe the type of the occurrence of an event within a given time interval.
Literal	***Description***
multiple Occurrences	Specifies that an event may occur more than once in a given time interval. **Tags:** atp.EnumerationValue=0
singleOccurrence	Specifies that an event shall occur only once in a given time interval. **Tags:** atp.EnumerationValue=1

Table 7.13: EventOccurrenceKindEnum

Enumeration	**SynchronizationTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::SynchronizationTiming Constraint
Note	This is used to describe the type of the synchronization timing constraint.
Literal	***Description***
response Synchronization	In case that the Synchronization Timing Constraint is specified for event chains, the response events of the associated event chains must occur synchronously with respect to the specified tolerance. All associated event chains must have the same stimulus event. In case that the Synchronization Timing Constraint is specified for events, the associated events must occur synchronously with respect to the specified tolerance. All associated events represent the response events of a common stimulus event, even such a stimulus event is not known yet or not available in the scope of the model. **Tags:** atp.EnumerationValue=0
stimulus Synchronization	In case that the Synchronization Timing Constraint is specified for event chains, the stimulus events of the associated event chains must occur synchronously with respect to the specified tolerance. All associated event chains must have the same response event. In case that the Synchronization Timing Constraint is specified for events, the associated events must occur synchronously with respect to the specified tolerance. All associated events represent the stimulus events of a common response event, even such a response event is not known yet or not available in the scope of the model. **Tags:** atp.EnumerationValue=1

Table 7.14: SynchronizationTypeEnum

[constr_4522] **`SynchronizationTimingConstraint` shall either reference events or event chains** ⌈ The `SynchronizationTimingConstraint` shall either reference timing description events or timing description event chains, but not both at the same time. ⌋*()*

7.4.1 SynchronizationTimingConstraint on Event Chains

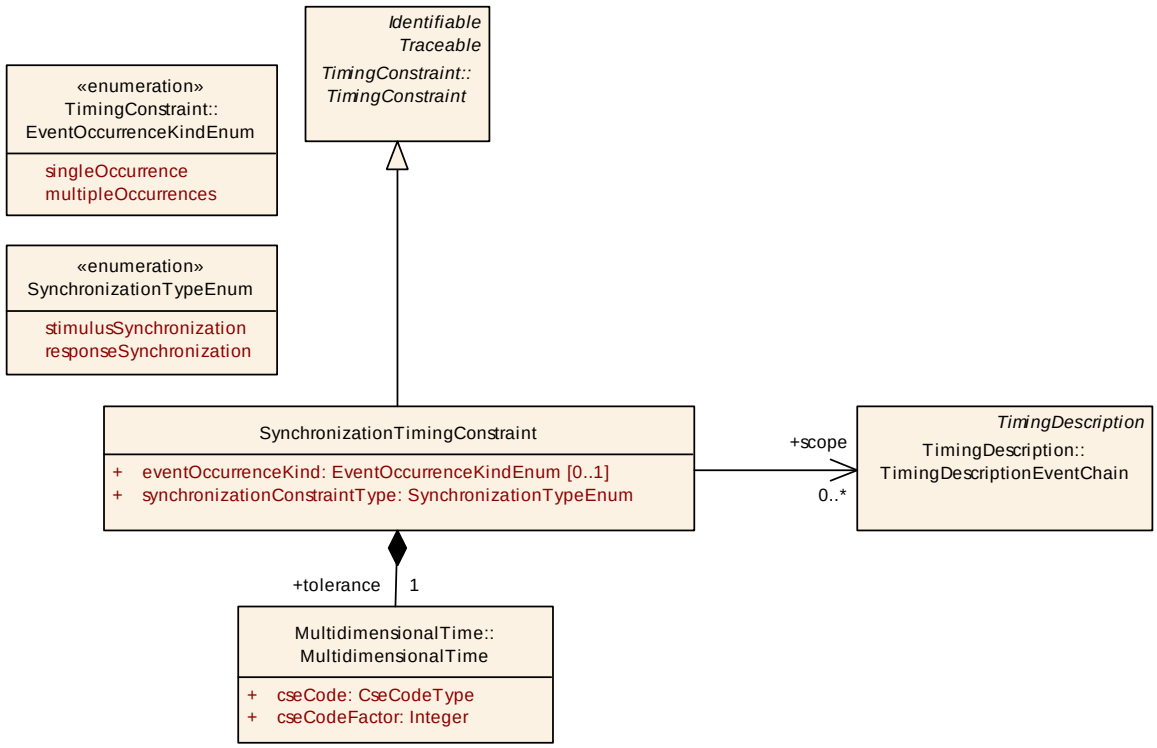


Figure 7.22: Synchronization Timing Constraint on Event Chains

The purpose of the `SynchronizationTimingConstraint` is to impose a synchronization constraint among either the stimulus or response event occurrences of two or more event chains. In the former case (stimulus synchronization) the referenced event chains shall have the same response event (join), or in the latter case (response synchronization) they shall have the same stimulus event (fork).

The `SynchronizationTimingConstraint` is characterized by the following parameters:

- Tolerance
- Event Occurrence Kind
- Synchronization Constraint Type

The parameters are described in the following and are illustrated in Figure 7.23 and Figure 7.24.

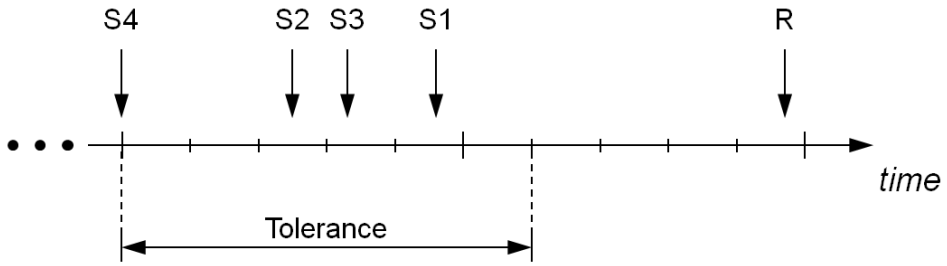
Tolerance The parameter `tolerance` specifies the time interval within which the referenced events shall occur synchronously. The events may occur in any order within this time interval. The time interval starts at the point-in-time when one of the referenced events occurs.

Event Occurrence Kind The optional parameter `eventOccurrenceKind` specifies whether the referenced events shall occur only once (single occurrence) or may occur multiple times (multiple occurrences) in the given time interval.

Synchronization Constraint Type The parameter `synchronizationConstraintType` specifies whether the `SynchronizationTimingConstraint` is imposed on the stimulus or response events of the referenced event chains.

[constr_4514] `SynchronizationTimingConstraint` shall reference at least two event chains ⌈ In the case, that the `SynchronizationTimingConstraint` is imposed on event chains then at least two (2) timing description event chains shall be referenced. ⌋*()*

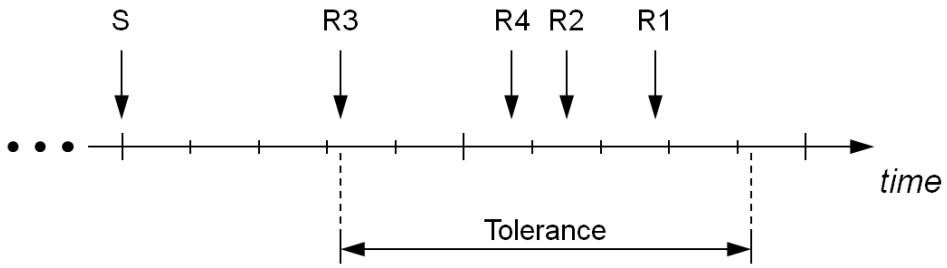
[constr_4521] Specifying attribute `synchronizationConstraintType` ⌈ The attribute `synchronizationConstraintType` shall be specified if the `SynchronizationTimingConstraint` is imposed on event chains. ⌋*()*



S# Stimulus Event
R Response Event

Event Occurrence

Figure 7.23: Parameters characterizing the Synchronization Timing Constraint imposed on the stimulus events of event chains.



S Stimulus Event
R# Response Event

Event Occurrence

Figure 7.24: Parameters characterizing the Synchronization Timing Constraint imposed on the response events of event chains.

An example for synchronizing on *stimuli* of event chains would be an adaptive cruise control that expects data from different sensors, which shall be sampled (quasi) simultaneously with respect to a predefined tolerance.

An example for synchronizing on *responses* of event chains would be the blinking of different indicator lights, which shall occur (quasi) simultaneously with respect to a predefined tolerance.

7.4.2 SynchronizationTimingConstraint on Events

As mentioned above, the purpose of the `SynchronizationTimingConstraint` is to impose a synchronization constraint among either the stimulus or response event occurrences of two or more event chains. However, in some cases the complete event chains are not entirely known, or not available in the scope of the model, at the point in time the timing constraint shall be specified. For this purpose, the AUTOSAR Timing Extensions allow the specification of synchronization constraints on events. In this case, the events referenced by the constraint are related implicitly, because they have a common stimulus (in case of constraint type `responseSynchronization` or a common response (in case of constraint type `stimulusSynchronization` not known yet, or not available in the scope of the model.

At a later stage during the development, when the refined software architecture exposes the complete event chains (e.g. because the common stimulus gets known), the respective event chains shall be specified and associated with a `SynchronizationTimingConstraint` on event chains (see 7.4.1) in order to refine the previously defined `SynchronizationTimingConstraint` on events.

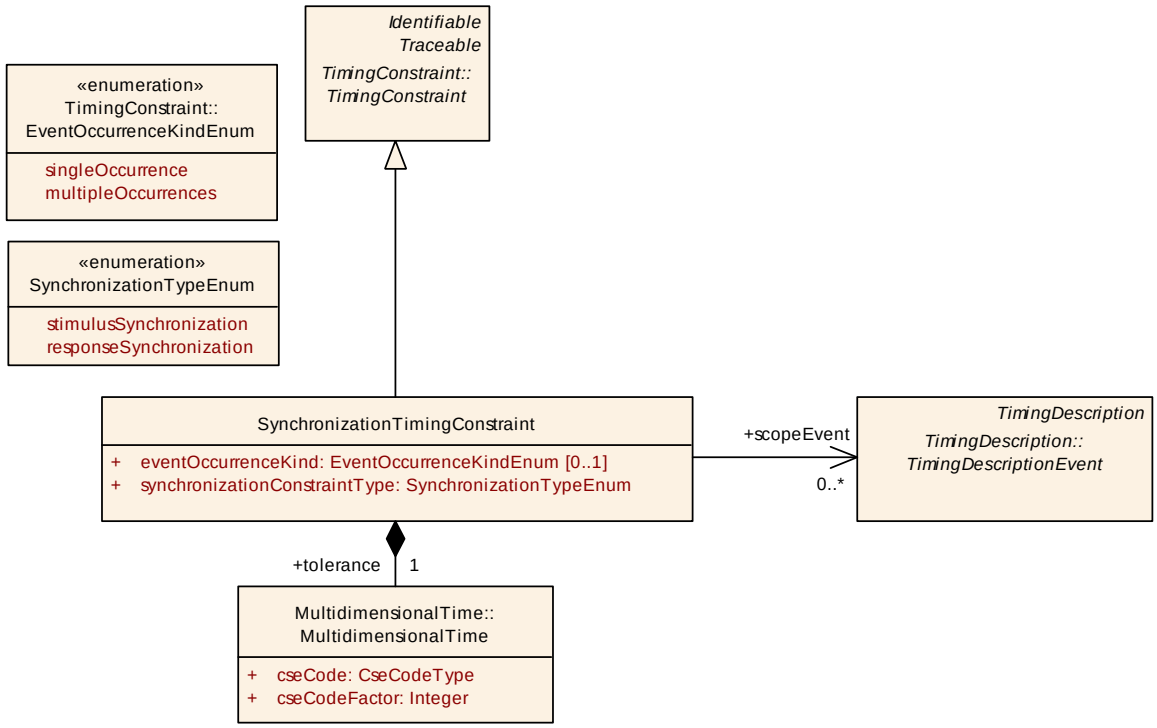


Figure 7.25: Synchronization Timing Constraint on Events

The purpose of the SynchronizationTimingConstraint is to impose a synchronization constraint among the occurrences of two or more events. The SynchronizationTimingConstraint is characterized by the following parameters:

- Tolerance
- Event Occurrence Kind
- Synchronization Constraint Type

The parameters are described in the following and are illustrated in Figure 7.26.

Tolerance The parameter tolerance specifies the time interval within which the referenced events shall occur synchronously. The events may occur in any order within this time interval. The time interval starts at the point-in-time when one of the referenced events occurs.

Event Occurrence Kind The parameter eventOccurrenceKind specifies whether the referenced events shall occur only once (single occurrence) or may occur multiple times (multiple occurrences) in the given time interval.

Synchronization Constraint Type The parameter synchronizationConstraintType specifies whether the associated events of the SynchronizationTimingConstraint have a common stimulus or response.

[constr_4513] SynchronizationTimingConstraint shall reference at least two events ⌈ In the case, that the SynchronizationTimingConstraint is imposed on events then at least two (2) timing description events shall be referenced. ⌋ *()*

[constr_4520] Specifying attribute synchronizationConstraintType ⌈ The attribute synchronizationConstraintType shall be specified if the SynchronizationTimingConstraint is imposed on events. ⌋*()*

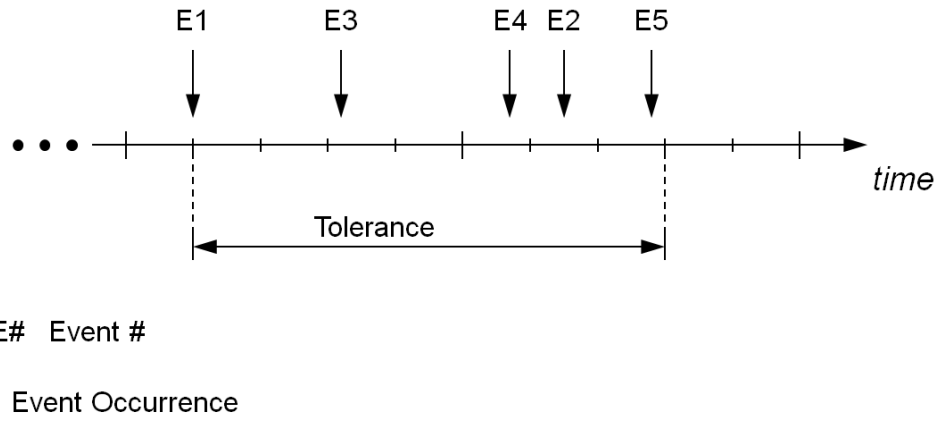


Figure 7.26: Parameter characterizing the Synchronization Constraint

7.5 SynchronizationPointConstraint

The purpose of the execution order constraint is to specify an order of execution among a number of executable entities. The capabilities of the execution order constraint are not sufficient to *explicitly* express that the execution of one or more executable entities shall be started if and only if one or more other executable entities finished execution. In other words, one or more executable entities shall succeed the execution of one or more other executable entities when those executable entities have all finished their execution. In order to specify this behavior the SynchronizationPointConstraint is used which specifies synchronization points between executable entities, referenced by their corresponding RTE or BSW events, or groups of executable entities specified by the EOCExecutableEntityRefGroups.

[TPS_TIMEX_00054] SynchronizationPointConstraint explicitly specifies a synchronization point between executable entities ⌈ The element SynchronizationPointConstraint is used to specify synchronization points between ExecutableEntitys. ⌋ *(RS_TIMEX_00023)*

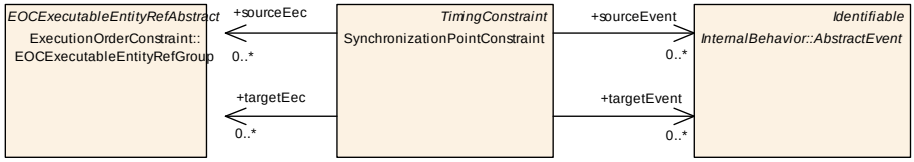


Figure 7.27: Synchronization Point Constraint

Class	**SynchronizationPointConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::SynchronizationPointConstraint			
Note	This constraint is used to specify a synchronization point either between groups of executable entities or individual executable entities referenced via their corresponding RTE or BSW events.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
sourceEec	EOCExecutableEntity RefGroup	*	ref	The source executable entities cluster containing the executable entities that shall finish execution before the synchronization point.
sourceEvent	AbstractEvent	*	ref	The executable entities — referenced by their events — that shall finish execution before the synchronization point.
targetEec	EOCExecutableEntity RefGroup	*	ref	The target executable entities cluster containing the executable entities that shall start execution after the synchronization point.
targetEvent	AbstractEvent	*	ref	The executable entities — referenced by their events — that shall start execution after the synchronization point.

Table 7.15: SynchronizationPointConstraint

Note that the element SynchronizationTimingConstraint is also related to synchronization, but this element imposes *temporal* constraints on the occurrence of events, like the occurrence of activation, start and termination of an executable entity. For example, a number of executable entities shall terminate and a number of executable entities shall start within a given time interval. But it is not possible to specify that the executable entities shall terminate in this time interval *before* the executable

entities start in this time interval. Therefore, this capability is not sufficient to express a synchronization point.

Indeed, it is possible to combine the two constraints, `SynchronizationPointConstraint` and `SynchronizationTimingConstraint` in order to specify that a synchronization point between a number of executable entities is mandatory and these executable entities shall terminate and start within a given time interval.

7.6 OffsetTimingConstraint

[TPS_TIMEX_00015] `OffsetTimingConstraint` specifies offset between occurrences of events ⌈ The element `OffsetTimingConstraint` is used to specify an offset between the occurrences of two timing description events. ⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00008)*

An `OffsetTimingConstraint` bounds the time offset between the occurrence of two timing events, without requiring a direct functional dependency between the source and the target.

This constraint type is frequently used in combination with the timing event `TDEventCycleStart` as source. In this case, the target event (e.g. the start of a `RunnableEntity`) is in most of the cases functional independent from the the source event.

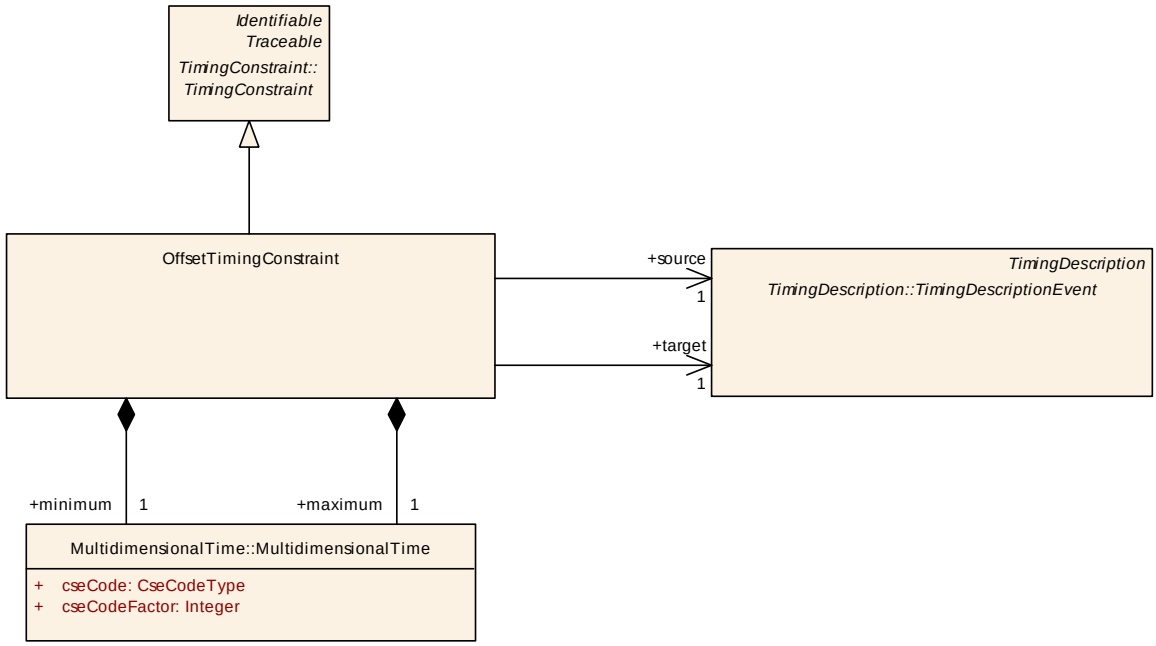


Figure 7.28: Offset Timing Constraint

Class	**OffsetTimingConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::OffsetConstraint			
Note	Bounds the time offset between the occurrence of two timing events, without requiring a direct functional dependency between the source and the target. If the target event occurs, it is expected to occur earliest with the minimum, and latest with the maximum offset relatively after the occurrence of the source event. Note: not every source event occurrence must be followed by a target event occurrence. In contrast to LatencyTimingConstraint, there must not necessarily be a causal dependency between the source and target event.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
maximum	MultidimensionalTime	1	aggr	The maximum offset the target event occurs relatively after the occurrence of the source event. **Tags:** xml.sequenceOffset=20
minimum	MultidimensionalTime	1	aggr	The mimum offset the target event occurs relatively after the occurrence of the source event. **Tags:** xml.sequenceOffset=10
source	TimingDescriptionEvent	1	ref	The timing event that the target event is to be synchronized with.
target	TimingDescriptionEvent	1	ref	The timing event which is expected to occur timely after the source event.

Table 7.16: OffsetTimingConstraint

7.7 ExecutionOrderConstraint

[TPS_TIMEX_00007] `ExecutionOrderConstraint` specifies sequence of executing executable entities ⌈ The element `ExecutionOrderConstraint` is used to specify the order of execution of a number of `ExecutableEntity`s. ⌋ *(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00014)*

An `ExecutionOrderConstraint` can be used in any of the timing views specified by the AUTOSAR Timing Extensions, as long as the `ExecutableEntity`s and/or the `AbstractEvent`s to be referenced are available in other AUTOSAR models, namely the Software Component Description and the Basic Software Module Description.

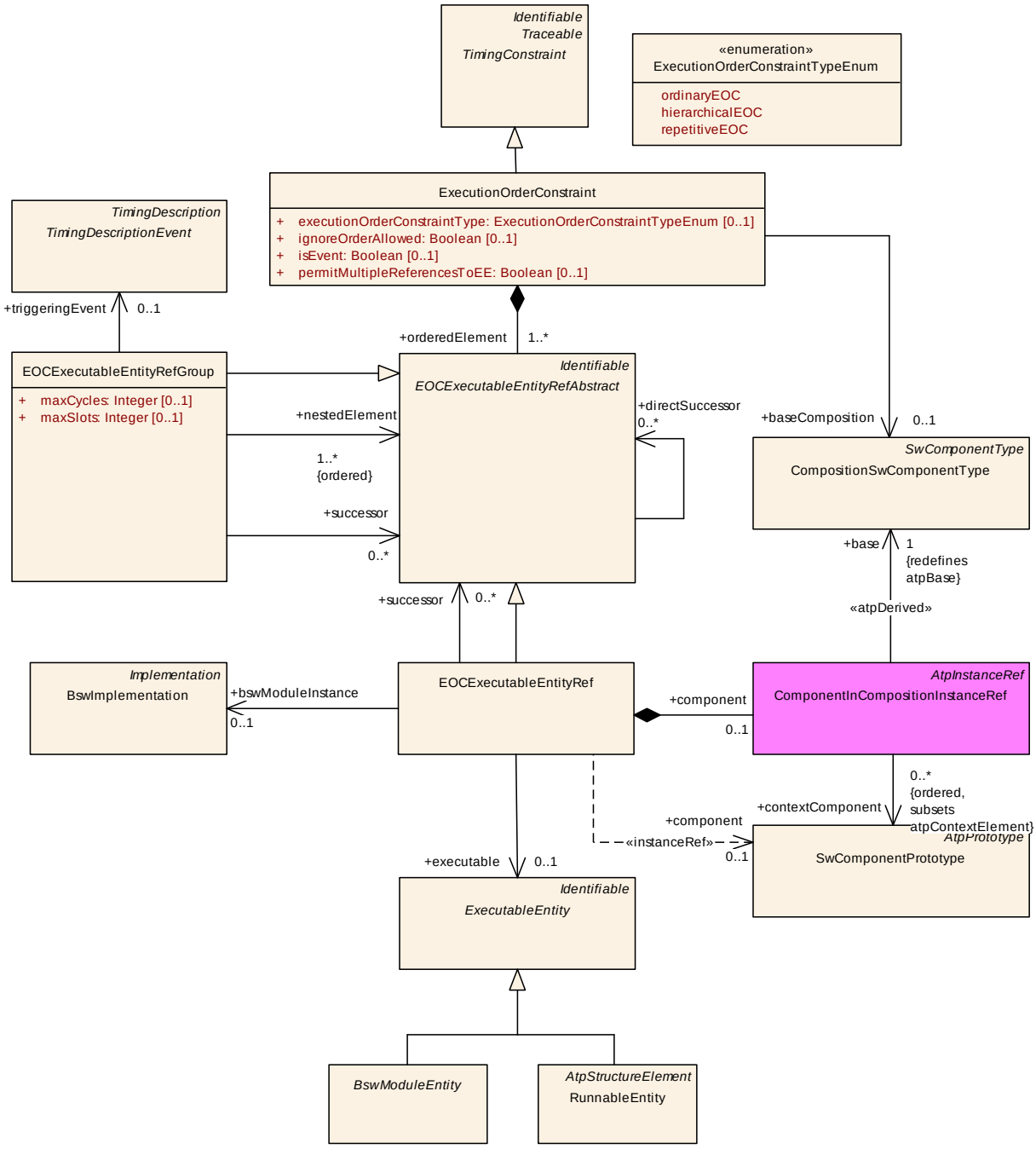


Figure 7.29: Specification of execution order constraints using Executable Entities.

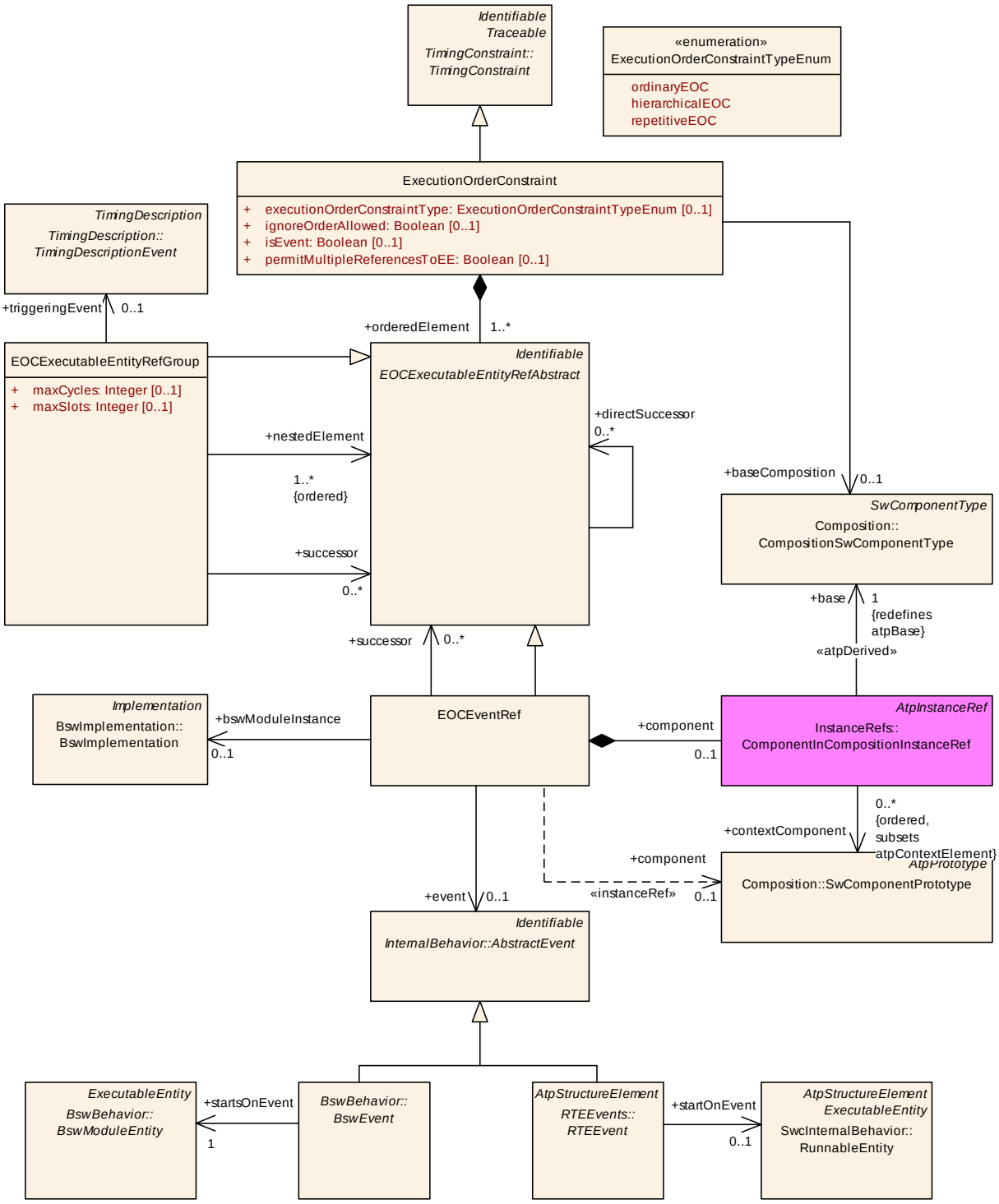


Figure 7.30: Specification of execution order constraints using RTE and BSW events.

Class	**ExecutionOrderConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint			
Note	This constraint is used to restrict the order of execution for a set of ExecutableEntities. The Execution OrderConstraint can be used in any timing view. The various scopes for ExecutionOrderConstraint are described below. Generally, each ExecutionOrder Constraint has a scope of software components and can reference all executable entities available in the corresponding internal behavior (RunnableEntity and BswModuleEntity) either directly or by the events activating respectively starting them (RteEvent and BswEvent). On VFB level an ExecutionOrderConstraint can be specified for RunnableEntities part of the composition hierarchy referenced by the VfbTiming. The ExecutionOrderConstraint is aggregated by the VfbTiming. On SW-C level an ExecutionOrderConstraint can be specified for RunnableEntities part of the Internal Behavior referenced by the SwcTiming. The ExecutionOrderConstraint is aggregated by the SwcTiming. On System level an ExecutionOrderConstraint can be specified for RunnableEntities part of the composition hierarchy of the system referenced by the SystemTiming. The ExecutionOrderConstraint is aggregated by the SystemTiming. On BSW Module level, an ExectionOrderConstraint can be specified for BswModuleEntities part of an BswInternalBehavior referenced by the BswModuleTiming. The ExecutionOrderConstraint is aggregated by the BswModuleTiming. On ECU level an ExecutionOrderConstraint can be specified for all ExecutableEntities and Events available via the EcucValueCollection, covering ECU Extract and BSW Module Configuration, referenced by the EcuTiming. The ExecutionOrderConstraint is aggregated by the EcuTiming.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base Composition	CompositionSw ComponentType	0..1	ref	Specifies the composition SW-C type playing the role of a SW-C containing further SW-Cs and represents the scope of the Execution Order Constraint.
executionOrder ConstraintType	ExecutionOrder ConstraintTypeEnum	0..1	attr	Specifies the specific type of ExecutionOrderConstraint.
ignoreOrder Allowed	Boolean	0..1	attr	Controls whether the order of execution specified by this constraint can be intentionally ignored (TRUE), or shall be respected (FALSE).
isEvent	Boolean	0..1	attr	Indicates whether the ExecutionOrderConstraint is only referring to Executable Entities (FALSE) or only to RTE and/or BSW Events (TRUE).
orderedElement	EOCExecutableEntity RefAbstract	1..*	aggr	The list of references to ExecutableEntities which shall be ordered.
permitMultiple ReferencesTo EE	Boolean	0..1	attr	Indicates that the ExecutionOrderConstraints permits that an Executable Entity is referenced multiple times (TRUE) or only once (FALSE) in the constraint.

Table 7.17: ExecutionOrderConstraint

Enumeration	**ExecutionOrderConstraintTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint
Note	This is used to describe the specific type of the ExecutionOrderConstraint.
Literal	***Description***
hierarchicalEOC	Specifies that the Execution Order Constraint specifies a hierarchical execution order constraint. **Tags:** atp.EnumerationValue=0
ordinaryEOC	Specifies that the Execution Order Constraint specifies an ordinary execution order constraint. **Tags:** atp.EnumerationValue=1
repetitiveEOC	Specifies that the Execution Order Constraint specifies a repetitive execution order constraint. **Tags:** atp.EnumerationValue=2

Table 7.18: ExecutionOrderConstraintTypeEnum

Class	***EOCExecutableEntityRefAbstract*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint			
Note	This is the abstractions for Execution Order Constraint Executable Entity References (leaves) and Execution Order Constraint Executable Entity Reference Groups (composites).			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	EOCEventRef, EOCExecutableEntityRef, EOCExecutableEntityRefGroup			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
directSuccessor	EOCExecutableEntity RefAbstract	*	ref	The direct successor of an executable entity or a group of executable entities.

Table 7.19: EOCExecutableEntityRefAbstract

Class	**EOCExecutableEntityRefGroup**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint			
Note	This is used to specify a group (composite) consisting of Execution Order Constraint Executable Entity References (leaves) and/or further Execution Order Constraint Executable Entity Reference Groups (composite).			
Base	*ARObject*, *EOCExecutableEntityRefAbstract*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
letInterval	TimingDescriptionEvent Chain	*	ref	This association references the TimingDescriptionEvent Chain that plays the role of a LET interval the executable entities in the group are assigned to.
maxCycles	Integer	0..1	attr	In case of a Repetitive Execution Order Constraint this attribute specifies the number of cycles the Execution Order Constraint is considering.
maxSlots	Integer	0..1	attr	In case of a Repetitive Execution Order Constraint this attribute specifies the number of slots every cycle of the Execution Order Constraint is consisting of.
nestedEle-ment (ordered)	EOCExecutableEntity RefAbstract	1..*	ref	This association is used to establish hierarchies of EOCEER Groups and References.
successor	EOCExecutableEntity RefAbstract	*	ref	The logical successor of an executable entity or a group of executable entities.
triggeringEvent	TimingDescriptionEvent	0..1	ref	In case of a Repetitive Execution Order Constraint this association references the timing description event triggering every cycle.

Table 7.20: EOCExecutableEntityRefGroup

Class	**EOCExecutableEntityRef**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint			
Note	This is used to define a reference to an ExecutableEntity If the ExecutionOrderConstraint is defined on VFB, System or ECU level, a reference to the Sw ComponentPrototype, via the ComponentInCompositionInstanceRef, the referenced ExecutableEntity belongs to, must be provided as context information.			
Base	*ARObject*, *EOCExecutableEntityRefAbstract*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
bswModule Instance	BswImplementation	0..1	ref	Specifies the BSW module instance the BSW module entity belongs to.
component	SwComponent Prototype	0..1	iref	This association references the specific instance of the SW-C prototype.

▽

△

Class	**EOCExecutableEntityRef**			
executable	ExecutableEntity	0..1	ref	The ExecutableEntity whose execution order is restricted by the contraint.
successor	EOCExecutableEntity RefAbstract	*	ref	The logical successor of an executable entity or a group of executable entities.

Table 7.21: EOCExecutableEntityRef

Class	**EOCEventRef**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint			
Note	This is used to define a reference to an RTE or BSW Event.			
Base	*ARObject*, *EOCExecutableEntityRefAbstract*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
bswModule Instance	BswImplementation	0..1	ref	Specifies the BSW module instance the BSW event is related to.
component	SwComponent Prototype	0..1	iref	This association references the specific instance of the SW-C prototype.
event	AbstractEvent	0..1	ref	The AbstractEvent (event) whose execution order is restricted by the contraint.
successor	EOCExecutableEntity RefAbstract	*	ref	The logical successor of an executable entity or a group of executable entities.

Table 7.22: EOCEventRef

[TPS_TIMEX_00038] Purpose of `EOCExecutableEntityRefAbstract` ⌈ The element `EOCExecutableEntityRefAbstract` is an abstract class that represents *both* primitives `EOCExecutableEntityRef`/`EOCEventRef` and their composites `EOCExecutableEntityRefGroup`. Essentially, it is used to compose `EOCExecutableEntityRefGroup`s, `EOCExecutableEntityRef`s and `EOCEventRef`s into tree structures to represent part-whole hierarchies. ⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00014)*

[TPS_TIMEX_00047] Purpose of `ExecutionOrderConstraintTypeEnum` ⌈ The element `ExecutionOrderConstraintTypeEnum` is used to specify the type of the execution order constraint and indicates whether it is an *ordinary*, *hierarchical*, or *repetitive* one. ⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00014)*

[TPS_TIMEX_00041] Purpose of `EOCExecutableEntityRefGroup` ⌈ The element `EOCExecutableEntityRefGroup` is used to define composites of `EOCExecutableEntityRefGroup`s, `EOCExecutableEntityRef`s and `EOCEventRef`s. ⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00014)*

[TPS_TIMEX_00046] Purpose of `EOCExecutableEntityRef` ⌈ The element `EOCExecutableEntityRef` is used to reference executable entities (`ExecutableEntity`) which shall be executed in a specific order. ⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00014)*

[TPS_TIMEX_00048] Purpose of `EOCEventRef` ⌈ The element `EOCEventRef` is used to reference `RTEEvent`s and `BswEvent`s in order to specify an execution order of executable entities. ⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00014)*

The elements described above can be used for creating different patterns of Execution Order Constraints for various purposes. These patterns are described in the following subsections. The constraints listed below applied to all of these patterns.

[constr_4525] Precedence of successor relationships `successor` and `directSuccessor` ⌈ The successor relationships `successor` and `directSuccessor` take always precedence over the `ordered` multiplicity of the association `nestedElement`. ⌋*()*

[constr_4532] Successor relationship is not self-referencing ⌈ The target and source of the successor relationships `successor` and `directSuccessor` shall not be the same. In other words an `EOCExecutableEntityRef` and `EOCExecutableEntityRefGroup` shall not reference itself as its logical or direct successor. ⌋*()*

[constr_4533] Maximum number of successor relationships ⌈ The maximum number of successor relationships, namely `successor` or `directSuccessor`, between two `EOCExecutableEntityRef`s, between two `EOCEventRef`s, between two `EOCExecutableEntityRefGroup`s, between an `EOCExecutableEntityRef` and an `EOCExecutableEntityRefGroup`, or between an `EOCEventRef` and an `EOCExecutableEntityRefGroup` is one (1). ⌋*()*

[constr_4534] Maximum number of `directSuccessor` relationships ⌈ The number of `directSuccessor` relationships of an `EOCExecutableEntityRef`, an `EOCEventRef`, or an `EOCExecutableEntityRefGroup` shall not exceed the number of independent execution units available in a system. ⌋*()*

[constr_4536] Compatible recurrence of any `ExecutableEntity` ⌈ In an `ExecutionOrderConstraint` the `ExecutableEntity`s, referenced by all `EOCExecutableEntityRef`s respectively all `EOCEventRef`s, shall be compatible with regard to their recurrence. ⌋*()*

[constr_4537] References among elements in an `ExecutionOrderConstraint` ⌈ An `EOCExecutableEntityRef` respectively `EOCEventRef` or an `EOCExecutableEntityRefGroup` shall reference only `EOCExecutableEntityRef`s, respectively all `EOCEventRef`s, or `EOCExecutableEntityRefGroup`s which are part of the same `ExecutionOrderConstraint`. ⌋*()*

[constr_4545] Referring either `ExecutableEntity`s or `AbstractEvent`s ⌈ An `ExecutionOrderConstraint` shall contain either only `EOCExecutableEntityRef` or only `EOCEventRef`, but not both. In the former case `ExecutableEntity`s are referenced and in the latter case `AbstractEvent`s are referenced. ⌋*()*

[constr_4546] Setting the attribute `isEvent` ⌈ The value of the attribute `isEvent` shall be set to ”TRUE” if and only if the execution order constraint refers to events only (refer to [constr_4545]). The value of the attribute `isEvent` shall be set to ”FALSE” if and only if the execution order constraint refers to executable entities only (refer to [constr_4545]). ⌋*()*

[constr_4547] Setting the attribute `permitMultipleReferencesToEE` ⌈ The value of the attribute `permitMultipleReferencesToEE` shall be specified if and only if the value of the attribute `isEvent` (refer to [constr_4546]) is set to ”FALSE”. In other words specifying whether an executable entity is permitted to be referenced more than once in an execution order constraint is only allowed in case of an execution order constraint refering to executable entities only. ⌋*()*

7.7.1 Ordinary Execution Order Constraint

The *Ordinary* Execution Order Constraint is used to specify an order of execution of executable entities. As sketched in Figure 7.31 the execution order constraint contains a number of execution entity references which reference the executable entities the execution order is imposed on. The associations `successor` and `directSuccessor` are used to specify the type of successor relationship and enforce the order of execution.

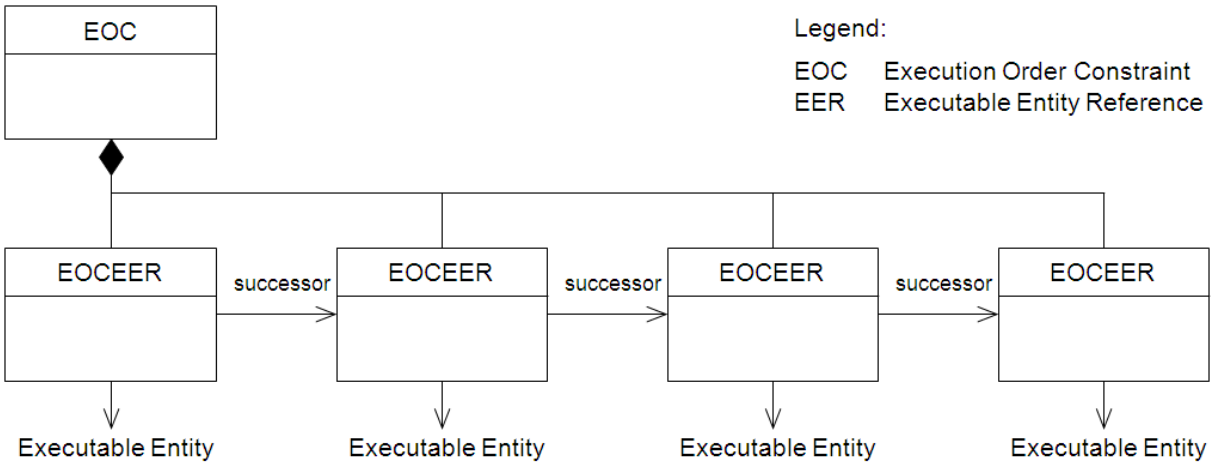


Figure 7.31: Example of an Ordinary Execution Order Constraint

In Figure 7.32 the execution order constraint contains a number of event references which reference the specific events—RTE events or BSW events—the execution order is imposed on.

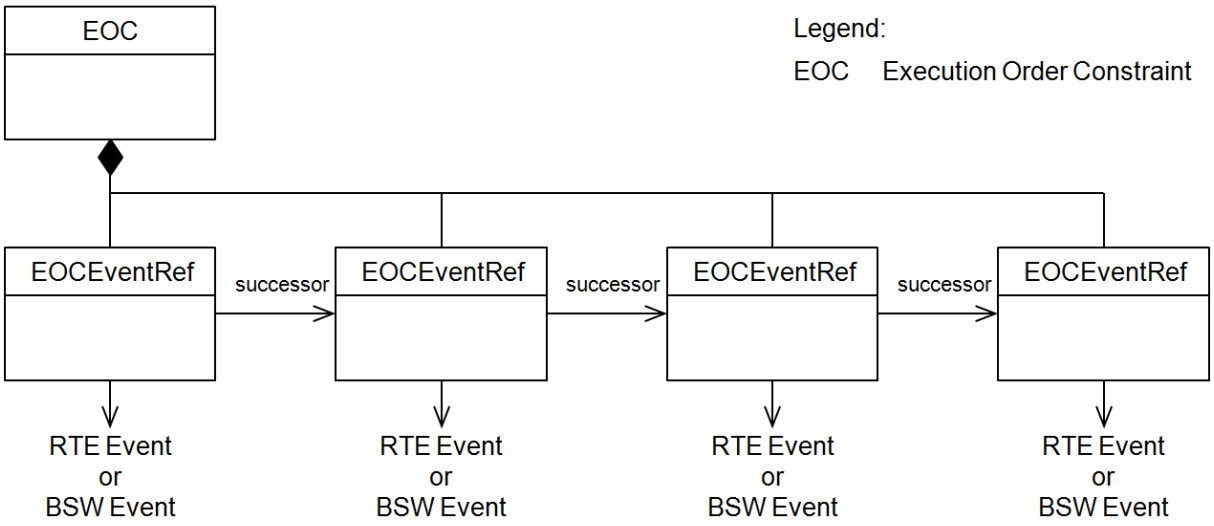


Figure 7.32: Example of an Ordinary Execution Order Constraint referencing events—RTE Events or BSW Events

[constr_4541] `EOCExecutableEntityRef` shall reference `ExecutableEntity` in Ordinary Execution Order Constraint ⌈ In an Ordinary Execution Order Constraint all `EOCExecutableEntityRef`s shall reference an `ExecutableEntity`. ⌋*()*

[constr_4548] `EOCEventRef` shall reference `AbstractEvent` in Ordinary Execution Order Constraint ⌈ In an Ordinary Execution Order Constraint all `EOCEventRef`s shall reference an `AbstractEvent`. ⌋*()*

7.7.2 Hierarchical Execution Order Constraint

The *Hierarchical* Execution Order Constraint is used to specify an order of execution of executable entities using the capability of creating groups of executable entities. In other words, it enables to specify tree-like structures of executable entity references respectively event references and executable entity reference groups. As sketched in Figure 7.33 the execution order constraint contains a number of execution entity references and one executable entity reference group, which in turn references a number of executable entity references referencing executable entities. The associations `successor` and `directSuccessor` between these references and the group are used to specify the type of successor relationship and enforce the order of execution.

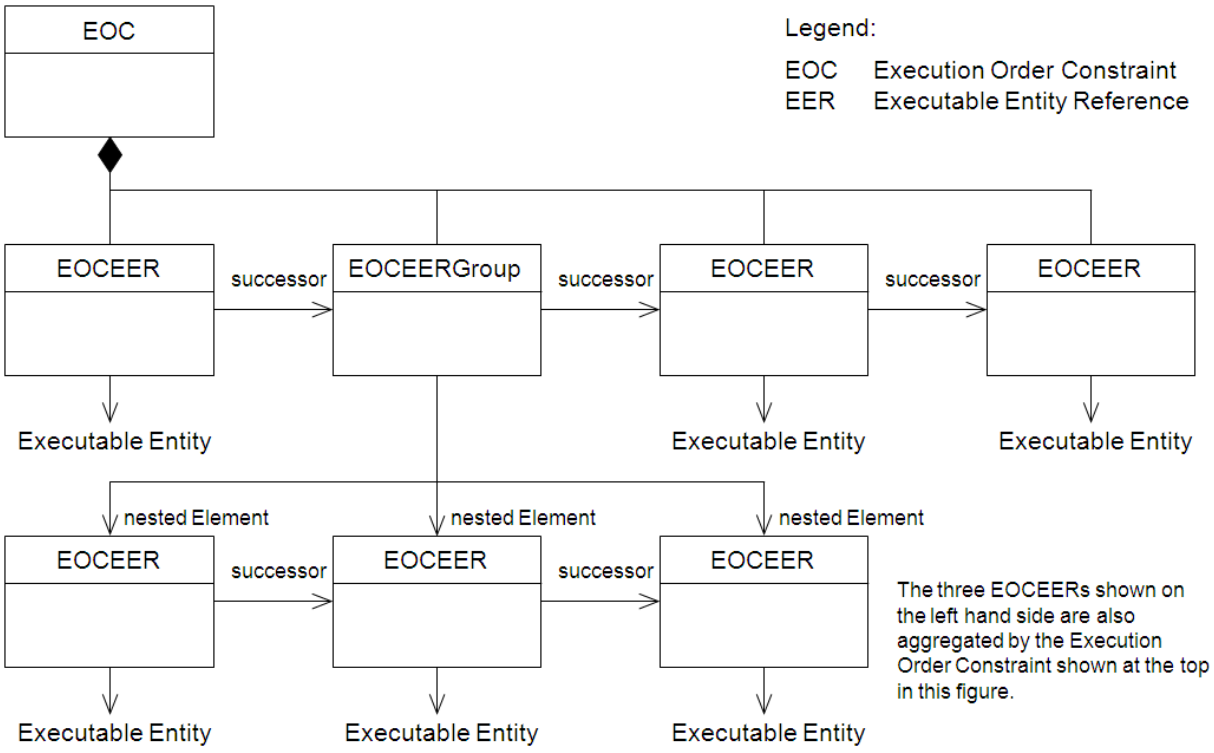


Figure 7.33: Example of a Hierarchical Execution Order Constraint

The following constraints shall be considered when creating Hierarchical Execution Order Constraints:

[constr_4523] Specifying attributes `maxCycles` and `maxSlots` ⌈ The optional attributes `maxCycles` and `maxSlots` shall never be specified in any element `EOCExecutableEntityRefGroup` that is part of a hierarchical execution order constraint. ⌋ *()*

[constr_4524] Referencing `TimingDescriptionEvent` ⌈ Any element `EOCExecutableEntityRefGroup` that is part of a hierarchical execution order constraint shall not reference any timing description event `TimingDescriptionEvent`. ⌋*()*

[constr_4538] Hierarchical Execution Order Constraint: `EOCExecutableEntityRef`, `EOCEventRef`, and `EOCExecutableEntityRefGroup` shall be target or source of a successor relationship ⌈ In a given Hierarchical Execution Order Constraint, each `EOCExecutableEntityRef`, `EOCEventRef`, and `EOCExecutableEntityRefGroup` which is not part of an `EOCExecutableEntityRefGroup` shall be target or source of at least one successor relationship.⌋*()*

[constr_4542] `EOCExecutableEntityRef` shall reference `ExecutableEntity` in Hierarchical Execution Order Constraint ⌈ In an Hierarchical Execution Order Constraint all `EOCExecutableEntityRef`s shall reference an `ExecutableEntity`. ⌋*()*

[constr_4549] `EOCEventRef` shall reference `AbstractEvent` in Hierarchical Execution Order Constraint ⌈ In an Hierarchical Execution Order Constraint all `EOCEventRef`s shall reference an `AbstractEvent`. ⌋*()*

[constr_4550] A Hierarchical Execution Order Constraint shall have an unambiguous root `EOCExecutableEntityRefGroup` ⌈ A Hierarchical Execution Order Constraint may contain multiple `orderedElement`s, which may be any combination of any number of `EOCExecutableEntityRef`s respectively `EOCEventRef`s and `EOCExecutableEntityRefGroup`s. Among these needs to be exactly one `EOCExecutableEntityRefGroup` being neither target nor source of any `successor` or `directSuccessor` relationship. This `EOCExecutableEntityRefGroup` is the *root* of the Hierarchical Execution Order Constraint. ⌋*()*

7.7.3 Repetitive Execution Order Constraint

The *Repetitive* Execution Order Constraint is used to specify *varying* execution order constraints depending on subsequent occurrences of a specific event. This enables one to specify that specific execution order constraints are imposed on a given number of executable entities whenever the particular timing description event occurs. For example, if the event A occurs the first time then the executable entities one (1), two (2) and three (3) shall be executed in this given order; if the event occurs the second time then the executable entities one (1), four (4) and five (5) shall be executed in this given order. And if the event A occurs a third time then the executable entities one (1), two (2) and three (3) shall be executed in this order again. And if the event A occurs a fourth time then the executable entities one (1), four (4) and five (5) shall be executed in this given order; and so forth. The occurrences of the specified event are called *cycles* and the order of the executable entities within a cycle is arranged by *slots*.

As sketched in Figure 7.34 the Repetitive Execution Order Constraints follows a specific pattern.

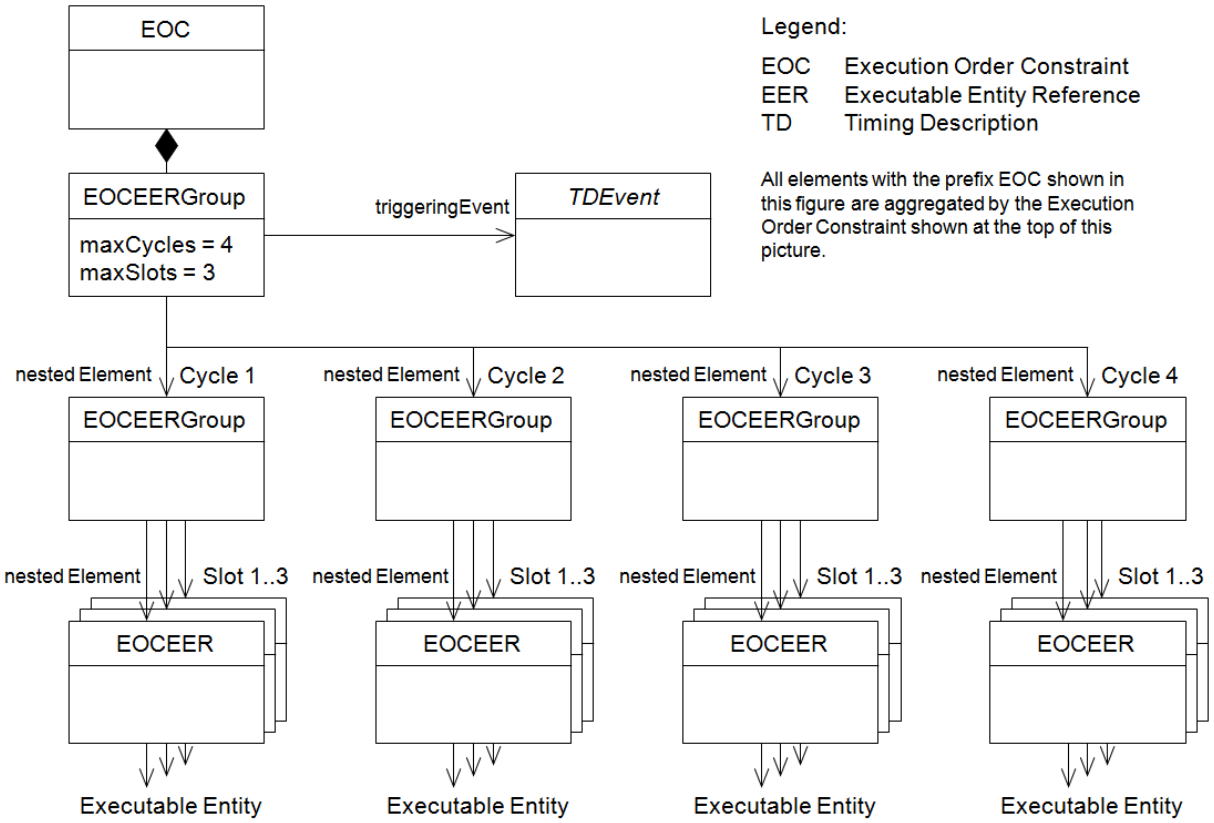


Figure 7.34: Example of a Repetitive Execution Order Constraint

The Repetitive Execution Order Constraint follows the pattern of the Hierarchical Execution Order Constraint, but some restrictions apply to the use and structure of groups of executable entity references. The execution order constraint consist of one group of executable entity references, called the *root* group, which references only other groups of executable entity references. And these groups in turn reference executable entity references respectively event references which eventually reference the specific executable entities respectively events. The *root* group specifies the maximum number of cycles `maxCycles` and the maximum number of slots `maxSlots`. The maximum number of cycles specifies the number of subsequent event occurrences after which the execution order constraint repeats, hence the name Repetitive Execution Order Constraint. And the maximum number of slots specifies the number of executable entities that are executed in a given order within a cycle.

Please note that the term `maxCycles`, respectively *cycle*, is a synonym for the term *maxRepetitions*, respectively *repetition*.

The table below presents the repetitive execution order constraint sketched in Figure 7.34 in a tabular way.

	Slots		
	1	2	3
Cycle 1	RE1	RE2	RE3
Cycle 2	RE1	RE4	RE5

Cycle 3	RE7	RE8	RE9
Cycle 4	RE1	RE8	RE6

Table 7.23: Example Repetitive Execution Order Constraint

The following constraints shall be considered when creating Repetitive Execution Order Constraints:

[constr_4526] Specifying `maxCycles` and `maxSlots` in a Repetitive Execution Order Constraint ⌈ The optional attributes `maxCycles` and `maxSlots` shall be specified only by the *root* group of executable entity references `EOCExecutableEntityRefGroup`. ⌋*()*

[constr_4527] Referencing `TimingDescriptionEvent` in a Repetitive Execution Order Constraint ⌈ The `TimingDescriptionEvent` shall be specified only by the *root* group of executable entity references `EOCExecutableEntityRefGroup`. ⌋*()*

[constr_4528] The *root* `EOCExecutableEntityRefGroup` shall reference only `EOCExecutableEntityRefGroup`s ⌈ The *root* `EOCExecutableEntityRefGroup` shall reference only groups of executable entity references respectively event references grouped by the element `EOCExecutableEntityRefGroup`s. ⌋*()*

[constr_4529] Number of nested elements referenced by the *root* `EOCExecutableEntityRefGroup` ⌈ The number of nested elements referenced by the *root* `EOCExecutableEntityRefGroup` shall be exactly the number given by the attribute `maxCycles`. ⌋*()*

[constr_4530] An `EOCExecutableEntityRefGroup` representing a cycle shall reference only `EOCExecutableEntityRef`s respectively `EOCEventRef`s ⌈ The `EOCExecutableEntityRefGroup` representing a cycle shall reference only executable entity references `EOCExecutableEntityRef`s respectively event references `EOCEventRef`s. ⌋*()*

[constr_4531] Number of nested elements referenced by `EOCExecutableEntityRefGroup` representing a cycle ⌈ The number of nested elements referenced by a `EOCExecutableEntityRefGroup` representing a cycle shall be exactly the number given by the attribute `maxSlots`. ⌋*()*

[constr_4539] The successor relationships `successor` and `directSuccessor` shall not be used ⌈ The successor relationships `successor` and `directSuccessor` shall not be used in a Repetitive Execution Order Constraint. ⌋*()*

[constr_4540] `maxCycles` and `maxSlots` shall not be zero ⌈ If the optional attributes `maxCycles` and `maxSlots` are used, then the values of the optional attributes `maxCycles` and `maxSlots` shall be greater than zero (0). ⌋*()*

7.8 ExecutionTimeConstraint

The AUTOSAR model provides a method to describe the execution time of an ExecutableEntity. Therefore the package ResourceConsumption contains the class ExecutionTime. The concept is described in the Basic Software Module Template document [6]. This execution time description represents a timing property of the system.

[TPS_TIMEX_00008] ExecutionTimeConstraint to specify execution time constraints ⌈ The element ExecutionTimeConstraint is used to specify minimum and maximum execution time constraints of executable entities. ⌋*(RS_TIMEX_00001, RS_TIMEX_00013)*

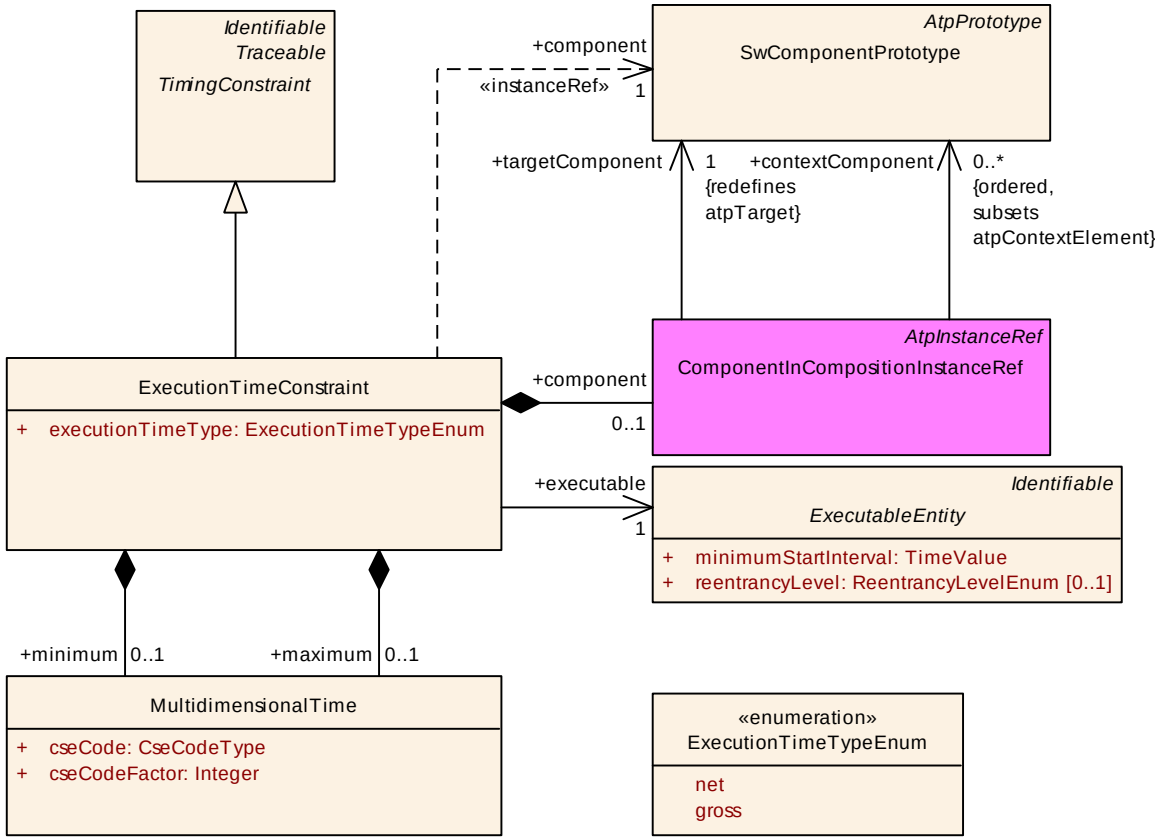


Figure 7.35: Execution time constraint

An ExecutionTimeConstraint references the ExecutableEntity, whose execution time shall be constrained. The ComponentInCompositionInstanceRef referenced by *component* defines the component instance, which contains the RunnableEntity (in case of a BSW ExecutableEntity, the *component* reference is omitted).

Class	**ExecutionTimeConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionTimeConstraint			
Note	An ExecutionTimeConstraint is used to specify the execution time of the referenced ExecutableEntity in the referenced component. A minimum and maximum execution time can be defined. Two types of execution time semantics can be used. The desired semantics can be set by the attribute executionTimeType: The "net" execution time is the time used to execute the ExecutableEntity without interruption and without external calls. The "gross" execution time is the time used to execute the ExecutableEntity without interruption including external calls to other entities. The time to execute the ExecutableEntity including interruptions by other entities and including external calls is commonly called "response time". The TimingExtensions provide the concept of event chains and latency constraints for that purpose. An event chain from the start of the entity to the termination of the entity with according latency constraint represents a response time constraint for that executable entity.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
component	SwComponent Prototype	0..1	iref	The component that containts the referenced Executable Entity for the ExecutionTimeConstraint. If the entity is in a basic software module no component must be provided.
executable	ExecutableEntity	1	ref	The referenced ExecutableEntity for the ExecutionTime Constraint.
executionTime Type	ExecutionTimeType Enum	1	attr	Specifies the type of the execution time constrained by ExecutionTimeConstraint,
maximum	MultidimensionalTime	0..1	aggr	The maximum execution time.
minimum	MultidimensionalTime	0..1	aggr	The minimum execution time.

Table 7.24: ExecutionTimeConstraint

Enumeration	**ExecutionTimeTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionTimeConstraint
Note	Specifies the type of the execution time constrained by ExecutionTimeConstraint.
Literal	***Description***
gross	Indicates that the given execution time is the time used to execute the ExecutableEntity without any interruption and and including external calls. **Tags:** atp.EnumerationValue=0
net	Indicates that the given execution time is the time used to execute the ExecutableEntity without any interruption and without any external calls. **Tags:** atp.EnumerationValue=1

Table 7.25: ExecutionTimeTypeEnum

8 Application Notes

This chapter outlines two application examples describing a potential approach to use the Specification of Timing Extensions in a practical way. Furthermore, chapter 8.3 describes the use of external VFB events in more detail.

8.1 Component integration

One of the main concerns for the usage of the AUTOSAR development methodology and AUTOSAR exchange formats is the need of OEMs and suppliers to exchange specification data in a machine-readable, reliable and straightforward way in order for example to integrate components in systems. The primary purpose of the ”Specification of Timing Extensions” is to facilitate requesting a specific timing behavior of such components. And this topic is described in this section in more detailed based on the basis of an integration scenario.

Integrating a software component instance delivered by an external party requires the provision of timing information related to this component. As this information can be attached to specific `SwComponentType`, with regards to its communication partners, the according view `VfbTiming` (see 3.2) is used. Additionally, specific timing constraints for implementing this software component are given, too.

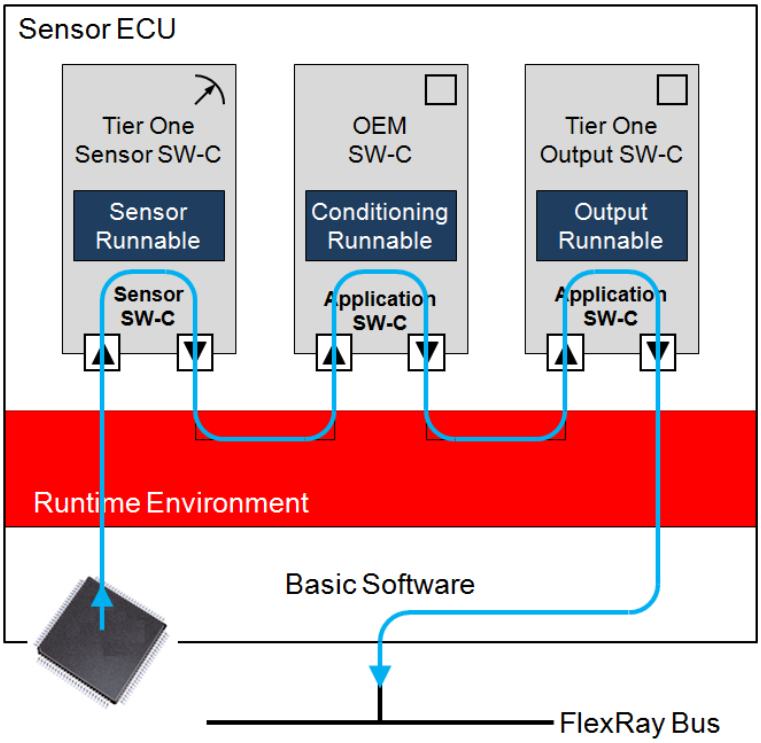


Figure 8.1: A Sensor ECU connected with a FlexRay Network and three software components

Figure 8.1 outlines the scenario in a demonstrative way. The shown ECU holds three software components: the first one, called ”Sensor SW-C”, reads data from a hardware sensor; the second one, called ”Conditioning SW-C”, performs signal data conditioning like filtering and averaging. And last but not least, the third one, called ‘”Output SW-C”, converts internal data representations (like 32bit Float) to ready-to-send data representations (like UInt16). As certain requirements for sensor data conditioning as an input for several other functions within the vehicle exist, the software component

”Conditioning SW-C” may be delivered by the OEM, directly. A partial description of these components, ports, and interfaces is shown in listing 8.1.

Listing 8.1: Software Component Descriptions and Interface Definitions

```
<AR-PACKAGE>
  <SHORT-NAME>Interfaces</SHORT-NAME>
  <ELEMENTS>
    <SENDER-RECEIVER-INTERFACE>
      <SHORT-NAME>InternalSensorData</SHORT-NAME>
      <DATA-ELEMENTS>
        <VARIABLE-DATA-PROTOTYPE>
          <SHORT-NAME>internValueX</SHORT-NAME>
          <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/
              Float</TYPE-TREF>
        </VARIABLE-DATA-PROTOTYPE>
        <VARIABLE-DATA-PROTOTYPE>
          <SHORT-NAME>internValueY</SHORT-NAME>
          <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/
              Float</TYPE-TREF>
        </VARIABLE-DATA-PROTOTYPE>
        <VARIABLE-DATA-PROTOTYPE>
          <SHORT-NAME>internValueZ</SHORT-NAME>
          <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/
              Float</TYPE-TREF>
        </VARIABLE-DATA-PROTOTYPE>
      </DATA-ELEMENTS>
    </SENDER-RECEIVER-INTERFACE>
    <SENDER-RECEIVER-INTERFACE>
      <SHORT-NAME>OutputSensorData</SHORT-NAME>
      <CATEGORY />
      <DATA-ELEMENTS>
        <VARIABLE-DATA-PROTOTYPE>
          <SHORT-NAME>outValueX</SHORT-NAME>
          <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/
              UInt16</TYPE-TREF>
        </VARIABLE-DATA-PROTOTYPE>
        <VARIABLE-DATA-PROTOTYPE>
          <SHORT-NAME>outValueY</SHORT-NAME>
          <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/
              UInt16</TYPE-TREF>
        </VARIABLE-DATA-PROTOTYPE>
        <VARIABLE-DATA-PROTOTYPE>
          <SHORT-NAME>outValueZ</SHORT-NAME>
          <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/
              UInt16</TYPE-TREF>
        </VARIABLE-DATA-PROTOTYPE>
      </DATA-ELEMENTS>
    </SENDER-RECEIVER-INTERFACE>
  </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
  <SHORT-NAME>SensorPackage</SHORT-NAME>
  <ELEMENTS>
    <SENSOR-ACTUATOR-SW-COMPONENT-TYPE>
      <SHORT-NAME>SensorComponent</SHORT-NAME>
```

```
    <PORTS>
      <P-PORT-PROTOTYPE>
        <SHORT-NAME>InternalSensorData</SHORT-NAME>
        <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
            Interfaces/InternalSensorData</PROVIDED-INTERFACE-TREF>
      </P-PORT-PROTOTYPE>
    </PORTS>
    <INTERNAL-BEHAVIORS>
      <SWC-INTERNAL-BEHAVIOR>
        <SHORT-NAME>SensorBehavior</SHORT-NAME>
        <RUNNABLES>
          <RUNNABLE-ENTITY>
            <SHORT-NAME>SensorRunnable</SHORT-NAME>
          </RUNNABLE-ENTITY>
        </RUNNABLES>
      </SWC-INTERNAL-BEHAVIOR>
    </INTERNAL-BEHAVIORS>
  </SENSOR-ACTUATOR-SW-COMPONENT-TYPE>
  <APPLICATION-SW-COMPONENT-TYPE>
    <SHORT-NAME>ConditioningComponent</SHORT-NAME>
    <PORTS>
      <R-PORT-PROTOTYPE>
        <SHORT-NAME>UnprocessedSensorData</SHORT-NAME>
        <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
            Interfaces/InternalSensorData</REQUIRED-INTERFACE-TREF>
      </R-PORT-PROTOTYPE>
      <P-PORT-PROTOTYPE>
        <SHORT-NAME>ProcessedSensorData</SHORT-NAME>
        <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
            Interfaces/InternalSensorData</PROVIDED-INTERFACE-TREF>
      </P-PORT-PROTOTYPE>
    </PORTS>
    <INTERNAL-BEHAVIORS>
      <SWC-INTERNAL-BEHAVIOR>
        <SHORT-NAME>ConditioningBehavior</SHORT-NAME>
        <RUNNABLES>
          <RUNNABLE-ENTITY>
            <SHORT-NAME>ConditioningRunnable</SHORT-NAME>
          </RUNNABLE-ENTITY>
        </RUNNABLES>
      </SWC-INTERNAL-BEHAVIOR>
    </INTERNAL-BEHAVIORS>
  </APPLICATION-SW-COMPONENT-TYPE>
  <APPLICATION-SW-COMPONENT-TYPE>
    <SHORT-NAME>OutputComponent</SHORT-NAME>
    <PORTS>
      <R-PORT-PROTOTYPE>
        <SHORT-NAME>UnprocessedSensorData</SHORT-NAME>
        <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
            Interfaces/InternalSensorData</REQUIRED-INTERFACE-TREF>
      </R-PORT-PROTOTYPE>
      <P-PORT-PROTOTYPE>
        <SHORT-NAME>ProcessedSensorData</SHORT-NAME>
        <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
            Interfaces/OutputSensorData</PROVIDED-INTERFACE-TREF>
      </P-PORT-PROTOTYPE>
```

```
      </PORTS>
      <INTERNAL-BEHAVIORS>
        <SWC-INTERNAL-BEHAVIOR>
          <SHORT-NAME>OutputBehavior</SHORT-NAME>
          <RUNNABLES>
            <RUNNABLE-ENTITY>
              <SHORT-NAME>OutputRunnable</SHORT-NAME>
            </RUNNABLE-ENTITY>
          </RUNNABLES>
        </SWC-INTERNAL-BEHAVIOR>
      </INTERNAL-BEHAVIORS>
    </APPLICATION-SW-COMPONENT-TYPE>
    <COMPOSITION-SW-COMPONENT-TYPE>
      <SHORT-NAME>TopLevelComposition</SHORT-NAME>
      <COMPONENTS>
        <SW-COMPONENT-PROTOTYPE>
          <SHORT-NAME>sensorComponent</SHORT-NAME>
          <TYPE-TREF DEST="SENSOR-ACTUATOR-SW-COMPONENT-TYPE">/
            SensorPackage/SensorComponent</TYPE-TREF>
        </SW-COMPONENT-PROTOTYPE>
        <SW-COMPONENT-PROTOTYPE>
          <SHORT-NAME>conditioningComponent</SHORT-NAME>
          <TYPE-TREF DEST="APPLICATION-SW-COMPONENT-TYPE">/
            SensorPackage/ConditioningComponent</TYPE-TREF>
        </SW-COMPONENT-PROTOTYPE>
        <SW-COMPONENT-PROTOTYPE>
          <SHORT-NAME>outputComponent</SHORT-NAME>
          <TYPE-TREF DEST="APPLICATION-SW-COMPONENT-TYPE">/
            SensorPackage/OutputComponent</TYPE-TREF>
        </SW-COMPONENT-PROTOTYPE>
      </COMPONENTS>
    </COMPOSITION-SW-COMPONENT-TYPE>
  </ELEMENTS>
</AR-PACKAGE>
```

In addition to this, receiving the sensor data by other ECUs, which are not shown in the figure, requires this data to fulfill certain timing requirements regarding their maximum age, for example. Mapped to the figure this means that the blue data path drawn shall has a specific temporal length. This requirement is the other hand side of the actual scenario.

The software component ”Conditioning SW-C” is delivered by the OEM for the sake of implementing special filtering or averaging algorithms applied on the measured sensor data. Thus, the mapping of software component to this ECU is fixed, already. To fulfill certain non-functional requirements, the implementing `RunnableEntity` of software component ”Conditioning” needs to be executed straight away after `RunnableEntity` of component ”Sensor SW-C” and right before `RunnableEntity` of component ”Output SW-C”. In addition, the Tier-1 needs information about the execution times of the runnable entity he has to expect when integrating the software component ”Conditioning SW-C”. Specifying this can be done by describing the measured (or simulated, estimated, etc.) execution times of the `RunnableEntity`. The following subsections give

a brief idea how this can be accomplished by utilizing the capabilities of the AUTOSAR Specification of Timing Extensions.

8.1.1 VFB view

At first, timing descriptions and constraints on VFB level are defined. The component ”Conditioning SW-C” receives data via its required port ”UnprocessedSensorData” at a specific point in time. This point is denoted by the event ”ConditioningReceived”, whereas the event ”ConditioningSent” denotes the point in time data is sent via the provided port ”ProcessedSensorData”. To prescribe a ”maximum age” for the reading input the `LatencyTimingConstraint` is used. For this, the *external* event ”SensorDataProduced” is defined. Based on this, an event chain between this external event ”SensorDataProduced” and the ”ConditioningReceived” event is specified — the event ”SensorDataProduced” plays the role of the stimulus event and the event ”ConditioningReceived” is playing the role of the response event with regard to the specified event chain. The latency timing constraint is pointing to this event chain. The representation of the events, event chain and the corresponding timing constraint is shown in listing 8.2.

Listing 8.2: LatencyConstraint and related events on VfbTiming view

```
<VFB-TIMING>
  <SHORT-NAME>SensorVfbTiming</SHORT-NAME>
  <TIMING-DESCRIPTIONS>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>ConditioningReceived</SHORT-NAME>
      <IS-EXTERNAL>false</IS-EXTERNAL>
      <PORT-REF DEST="R-PORT-PROTOTYPE">/SensorPackage/
          ConditioningComponent/UnprocessedSensorData</PORT-REF>
      <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
          InternalSensorData/internValueX</DATA-ELEMENT-REF>
      <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
          PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
    </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>ConditioningSent</SHORT-NAME>
      <IS-EXTERNAL>false</IS-EXTERNAL>
      <PORT-REF DEST="P-PORT-PROTOTYPE">/SensorPackage/
          ConditioningComponent/ProcessedSensorData</PORT-REF>
      <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
          InternalSensorData/internValueX</DATA-ELEMENT-REF>
      <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
          PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
    </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>SensorDataProduced</SHORT-NAME>
      <IS-EXTERNAL>true</IS-EXTERNAL>
      <PORT-REF DEST="R-PORT-PROTOTYPE">/SensorPackage/
          ConditioningComponent/UnprocessedSensorData</PORT-REF>
      <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
          InternalSensorData/internValueX</DATA-ELEMENT-REF>
```

```
      <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
          PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
    </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>InputVfbChain</SHORT-NAME>
      <CATEGORY />
      <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/
          TimingPackage/SensorVfbTiming/SensorDataProduced</STIMULUS
          -REF>
      <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/
          TimingPackage/SensorVfbTiming/ConditioningReceived</
          RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/
            TimingPackage/SensorVfbTiming/InputVfbChain</SEGMENT-REF
            >
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
  </TIMING-DESCRIPTIONS>
  <TIMING-REQUIREMENTS>
    <LATENCY-TIMING-CONSTRAINT>
      <SHORT-NAME>InputVfbLatency</SHORT-NAME>
      <LATENCY-CONSTRAINT-TYPE>AGE</LATENCY-CONSTRAINT-TYPE>
      <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/
          TimingPackage/SensorVfbTiming/InputVfbChain</SCOPE-REF>
      <MINIMUM>
        <CSE-CODE>1</CSE-CODE>
        <CSE-CODE-FACTOR>40</CSE-CODE-FACTOR>
      </MINIMUM>
      <MAXIMUM>
        <CSE-CODE>1</CSE-CODE>
        <CSE-CODE-FACTOR>50</CSE-CODE-FACTOR>
      </MAXIMUM>
    </LATENCY-TIMING-CONSTRAINT>
    <EXECUTION-ORDER-CONSTRAINT>
      <SHORT-NAME>EOC1</SHORT-NAME>
      <ORDERED-ELEMENTS>
        <EOC-EXECUTABLE-ENTITY-REF>
          <SHORT-NAME>SensorRunnableRef</SHORT-NAME>
          <COMPONENT-IREF>
            <TARGET-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">/
                SensorPackage/TopLevelComposition/sensorComponent</
                TARGET-COMPONENT-REF>
          </COMPONENT-IREF>
          <EXECUTABLE-REF DEST="RUNNABLE-ENTITY">/SensorPackage/
              SensorComponent/SensorBehavior/SensorRunnable</
              EXECUTABLE-REF>
          <SUCCESSOR-REFS>
            <SUCCESSOR-REF DEST="EOC-EXECUTABLE-ENTITY-REF">/
                TimingPackage/SensorVfbTiming/EOC1/
                ConditioningRunnableRef</SUCCESSOR-REF>
          </SUCCESSOR-REFS>
        </EOC-EXECUTABLE-ENTITY-REF>
        <EOC-EXECUTABLE-ENTITY-REF>
          <SHORT-NAME>ConditioningRunnableRef</SHORT-NAME>
          <COMPONENT-IREF>
```

```
            <TARGET-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">/
                SensorPackage/TopLevelComposition/
                conditioningComponent</TARGET-COMPONENT-REF>
          </COMPONENT-IREF>
          <EXECUTABLE-REF DEST="RUNNABLE-ENTITY">/SensorPackage/
             ConditioningComponent/ConditioningBehavior/
             ConditioningRunnable</EXECUTABLE-REF>
          <SUCCESSOR-REFS>
            <SUCCESSOR-REF DEST="EOC-EXECUTABLE-ENTITY-REF">/
               TimingPackage/SensorVfbTiming/EOC1/OutputRunnableRef
               </SUCCESSOR-REF>
          </SUCCESSOR-REFS>
        </EOC-EXECUTABLE-ENTITY-REF>
        <EOC-EXECUTABLE-ENTITY-REF>
          <SHORT-NAME>OutputRunnableRef</SHORT-NAME>
          <COMPONENT-IREF>
            <TARGET-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">/
               SensorPackage/TopLevelComposition/outputComponent</
               TARGET-COMPONENT-REF>
          </COMPONENT-IREF>
          <EXECUTABLE-REF DEST="RUNNABLE-ENTITY">/SensorPackage/
             OutputComponent/OutputBehavior/OutputRunnable</
             EXECUTABLE-REF>
        </EOC-EXECUTABLE-ENTITY-REF>
      </ORDERED-ELEMENTS>
    </EXECUTION-ORDER-CONSTRAINT>
  </TIMING-REQUIREMENTS>
  <COMPONENT-REF DEST="COMPOSITION-SW-COMPONENT-TYPE">/
     SensorPackage/TopLevelComposition</COMPONENT-REF>
</VFB-TIMING>
```

8.1.2 ECU view

After generating the ECU extract, implementation related details of the ECU are available and execution order constraints for the mapped software components — more precise, their runnable entities — exist. For the sake of easiness, each software component implements one `RunnableEntity`. Constraining their execution order using the `ExecutionOrderConstraint` is shown in listing 8.3.

Listing 8.3: Execution Order Constraint for Three Runnable Entities

```
<EXECUTION-ORDER-CONSTRAINT>
  <SHORT-NAME>EOC1</SHORT-NAME>
  <ORDERED-ELEMENTS>
    <EOC-EXECUTABLE-ENTITY-REF>
      <SHORT-NAME>SensorRunnableRef</SHORT-NAME>
      <COMPONENT-IREF>
        <TARGET-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">/
           SensorPackage/TopLevelComposition/sensorComponent</
           TARGET-COMPONENT-REF>
      </COMPONENT-IREF>
```

```
        <EXECUTABLE-REF DEST="RUNNABLE-ENTITY">/SensorPackage/
            SensorComponent/SensorBehavior/SensorRunnable</
            EXECUTABLE-REF>
        <SUCCESSOR-REFS>
          <SUCCESSOR-REF DEST="EOC-EXECUTABLE-ENTITY-REF">/
              TimingPackage/SensorVfbTiming/EOC1/
              ConditioningRunnableRef</SUCCESSOR-REF>
        </SUCCESSOR-REFS>
      </EOC-EXECUTABLE-ENTITY-REF>
      <EOC-EXECUTABLE-ENTITY-REF>
        <SHORT-NAME>ConditioningRunnableRef</SHORT-NAME>
        <COMPONENT-IREF>
          <TARGET-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">/
              SensorPackage/TopLevelComposition/
              conditioningComponent</TARGET-COMPONENT-REF>
        </COMPONENT-IREF>
        <EXECUTABLE-REF DEST="RUNNABLE-ENTITY">/SensorPackage/
            ConditioningComponent/ConditioningBehavior/
            ConditioningRunnable</EXECUTABLE-REF>
        <SUCCESSOR-REFS>
          <SUCCESSOR-REF DEST="EOC-EXECUTABLE-ENTITY-REF">/
              TimingPackage/SensorVfbTiming/EOC1/OutputRunnableRef
              </SUCCESSOR-REF>
        </SUCCESSOR-REFS>
      </EOC-EXECUTABLE-ENTITY-REF>
      <EOC-EXECUTABLE-ENTITY-REF>
        <SHORT-NAME>OutputRunnableRef</SHORT-NAME>
        <COMPONENT-IREF>
          <TARGET-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">/
              SensorPackage/TopLevelComposition/outputComponent</
              TARGET-COMPONENT-REF>
        </COMPONENT-IREF>
        <EXECUTABLE-REF DEST="RUNNABLE-ENTITY">/SensorPackage/
            OutputComponent/OutputBehavior/OutputRunnable</
            EXECUTABLE-REF>
      </EOC-EXECUTABLE-ENTITY-REF>
    </ORDERED-ELEMENTS>
  </EXECUTION-ORDER-CONSTRAINT>
```

Another typical constraint describes the maximum time to be elapsed for sending data on the bus. Therefore, an event ”DataTransmitted” representing the point in time the data is sent on the bus is specified using the event type `TDEventFrame` (listing 8.4). Additionally a `TimingDescriptionEventChain` is specified having ”ConditioningSent” as stimulus event and ”DataTransmitted” as response event (see listing 8.5).

Listing 8.4: Event describing the point in time where data is sent on the bus

```
<TD-EVENT-FRAME>
  <SHORT-NAME>DataTransmitted</SHORT-NAME>
  <FRAME-REF DEST="FLEXRAY-FRAME">/SystemDescriptionPackage/
      SensorFrame</FRAME-REF>
  <PHYSICAL-CHANNEL-REF DEST="FLEXRAY-PHYSICAL-CHANNEL">/
      SystemDescriptionPackage/SampleFrCluster/FrChannel110MBit<
      /PHYSICAL-CHANNEL-REF>
```

```
    <TD-EVENT-TYPE>FRAME-TRANSMITTED-ON-BUS</TD-EVENT-TYPE>
  </TD-EVENT-FRAME>
```

Listing 8.5: Event chain describing the sending path of data

```
<TIMING-DESCRIPTION-EVENT-CHAIN>
  <SHORT-NAME>SensorEcuChain</SHORT-NAME>
  <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/
     TimingPackage/SensorVfbTiming/ConditioningSent</STIMULUS-
     REF>
  <RESPONSE-REF DEST="TD-EVENT-FRAME">/TimingPackage/
     SensorEcuTiming/DataTransmitted</RESPONSE-REF>
  <SEGMENT-REFS>
    <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/
       TimingPackage/SensorEcuTiming/SensorEcuChain</SEGMENT-
       REF>
  </SEGMENT-REFS>
</TIMING-DESCRIPTION-EVENT-CHAIN>
```

The constraint prescribing the maximum latency between the point in time the stimulus event occurs and the point in time the response event occurs is shown in listing 8.6.

Listing 8.6: Latency constraint prescribing the maximum latency of sending path within the ECU

```
<TIMING-REQUIREMENTS>
  <LATENCY-TIMING-CONSTRAINT>
    <SHORT-NAME>SensorEcuLatency</SHORT-NAME>
    <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
    <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/
       TimingPackage/SensorEcuTiming/SensorEcuChain</SCOPE-REF>
    <MAXIMUM>
      <CSE-CODE>1</CSE-CODE>
      <CSE-CODE-FACTOR>400</CSE-CODE-FACTOR>
    </MAXIMUM>
  </LATENCY-TIMING-CONSTRAINT>
</TIMING-REQUIREMENTS>
```

8.2 Engine control

This example illustrates an example for the definition of timing constraints in an engine management system. Although the system is simplified to be included within this chapter it is based on a real world example in its basic concepts.

8.2.1 Overview

The example system is an air mass controlled gasoline internal combustion engine control system. Roughly, the functionality of software components can be categorized as described in the following:

Sensors Three SensorActuatorSwComponentTypes called ”MassAirFlowSensor”, ”AcceleratorPedalSensor”, and ”ThrottleSensor” are responsible for reading in the most important control factors.

Application Based Calculation Most ApplicationSwComponentTypes calculate the new control factors for the engine. In summary these components are ”AcceleratorPedalVoter”, ”ThrottleController”, ”ThrottleChange”, ”BaseFuelMass”, ”TransientFuelMass”, ”Ignition”, and ”TotalFuelMass”.

Actuators The control of the actuators is encapsulated by the SensorActuatorSwComponentTypes ”ThrottleActuator”, ”InjectionActuator”, and ”IgnitionAcutator”.

Engine Mode and Control The engine can be operated in different operation modes. The AtomicSwComponentType ”OperatingMode” includes a state machine which decides what setting for the application based calculation is used depending on the current mode, for example normal drive, idle speed, etc. Similar values are delivered by the ”IdleSpeedControl” which determines important inputs for application calculation during idle speed.

Miscellaneous The AtomicSwComponentType ”InjBatVoltCorrectionSensor” provides the input from the battery voltage sensor. The AtomicSwComponentType”CylNumObserver” is checking whether a change in the cylinder number is sensed and afterwards schedules the application based calculation. In this example application it is assumed that the cylinder number is provided externally within a rate of 2,5ms.

Since giving a complete overview of the system would result in an highly connected graph, Figure 8.2 shows a simplified sketch of the System because a detailed presentation of such a system would go far beyond the scope of this section. The blue colored lines show important signal paths that are considered to be important for timing analysis and are typically subject to be constrained by timing requirements.

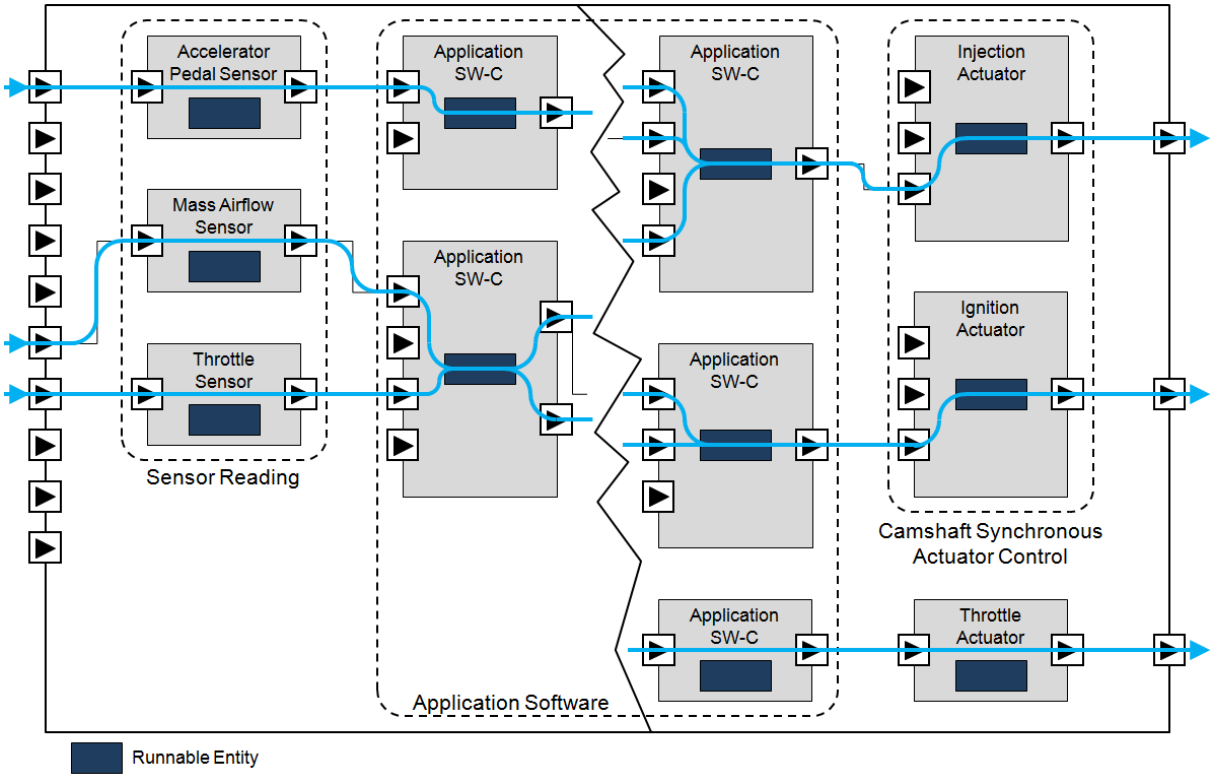


Figure 8.2: Rough sketch of an internal combustion engine control system including important signal flow paths.

8.2.2 Timing Requirements

Assumed the following timing requirements are stated and imposed on the sketched engine control application:

1. When the position of the accelerator pedal changes then the throttle shall be actuated within 30ms.
2. The maximum age of the throttle position value tolerated by the application software shall not exceed 10ms.
3. The calculation of ignition timing shall be completed latest 50ms after the a change of the position of the accelerator pedal has been detected.
4. The calculation of the ignition timing shall be completed 3ms after the `BswInterruptEntity` of the Basic Software Module called ”Camshaft” has been activated.
5. For each cylinder the calculation of the corresponding injection mass shall be activated every 20ms (50Hz) and shall be completed not later than 20ms after its activation.

The listed requirements above need to be transformed into timing requirements captured by the capabilities of the AUTOSAR Specification of Timing Extensions. The following subsections present how the timing models look alike for every of those requirements.

8.2.3 Formal description of timing constraints in VFB View

It should be understood that the requirements from section 8.2.2 can be mapped to timing constraints that reference different parts of the system. Since a comprehensive and detailed overview of the whole system would go beyond the scope of this section only the important parts of the system and its timing are given to convey the idea behind using the AUTOSAR Specification of Timing Extensions for each presented timing requirement.

The requirements 1 to 3 are expressed in the VFB view respectively VFB Timing (`VfbTiming`).

8.2.3.1 Requirement 1

Figure 8.3 shows the simplified signal flow and involved components. It has been identified that the critical path of execution has an effect on four software components. The sensor software component ”AccelleratorPedal” is responsible for reading in the signal and passes it to the application software component ”AccelleratorPedalVoter”. Afterwards the processed signal is further processed in the application software component ”ThrottleController” until it is finally sent to the actuator via the actuator software component ”ThrottleActuator”. For specification of the timing constraint a timing description event chain must be defined along with the appropriate timing description events. These timing descriptions and timing constraints are presented in listing 8.7 and cover the whole path from the sensor to the actuator.

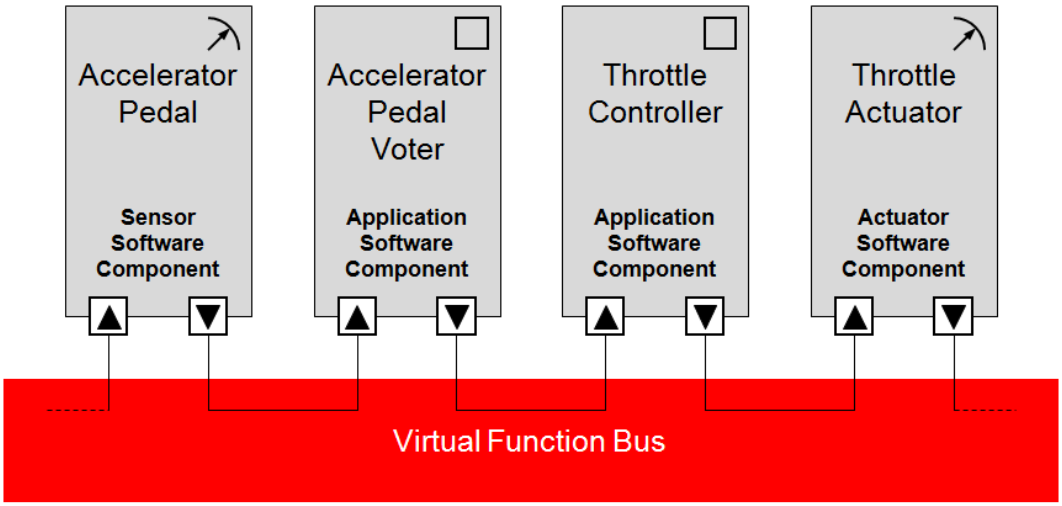


Figure 8.3: Involved components for signal flow from ”AcelleratorPedal” to ”ThrottleActuator” for timing requirement 1.

Since a timing constraint is imposed on the ”sensor to actuator” communication the chosen constraint is a `LatencyTimingConstraint` and its type is ”Reaction”. Also note that the overall timing event chain references all event chain segments the event chain consists of.

Listing 8.7: Event Definitions and Constraints for Requirement 1

```
<VFB-TIMING>
  <SHORT-NAME>EngineControlVfbTiming1</SHORT-NAME>
  <TIMING-DESCRIPTIONS>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>RAcceleratorPedalPositionSensorReceived</SHORT-
          NAME>
      <IS-EXTERNAL>false</IS-EXTERNAL>
      <PORT-REF DEST="R-PORT-PROTOTYPE">/Components/
          AcceleratorPedal/RAcceleratorPedalPositionSensor</PORT-REF
          >
      <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
          IAcceleratorPedalPositionSensor/
          AcceleratorPedalPositionSensor</DATA-ELEMENT-REF>
      <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
          PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
    </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>PAcceleratorPedalPositionSent</SHORT-NAME>
      <IS-EXTERNAL>false</IS-EXTERNAL>
      <PORT-REF DEST="P-PORT-PROTOTYPE">/Components/
          AcceleratorPedal/PAcceleratorPedalPosition</PORT-REF>
      <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
          IAcceleratorPedalPosition/AcceleratorPedalPosition</DATA-
          ELEMENT-REF>
      <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
          PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
    </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
```

```
    <SHORT-NAME>RAcceleratorPedalPositionReceived</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="R-PORT-PROTOTYPE">/Components/
      AcceleratorPedalVoter/RAcceleratorPedalPosition</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
      IAcceleratorPedalPosition/AcceleratorPedalPosition</DATA-
      ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
      PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>PVotedPedalPositionSent</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="P-PORT-PROTOTYPE">/Components/
      AcceleratorPedalVoter/PVotedPedalPosition</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
      IVotedPedalPosition/VotedPedalPosition</DATA-ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
      PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>RVotedPedalPositionReceived</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="R-PORT-PROTOTYPE">/Components/
      ThrottleController/RVotedPedalPosition</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
      IVotedPedalPosition/VotedPedalPosition</DATA-ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
      PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>PUnlimitedThrottlePositionSent</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="P-PORT-PROTOTYPE">/Components/
      ThrottleController/PUnlimitedThrottlePosition</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
      IUnlimitedThrottlePosition/UnlimitedThrottlePosition</DATA
      -ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
      PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>RUnlimitedThrottlePositionReceived</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="R-PORT-PROTOTYPE">/Components/
      ThrottleActuator/RUnlimitedThrottlePosition</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
      IUnlimitedThrottlePosition/UnlimitedThrottlePosition</DATA
      -ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
      PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>PUnlimitedThrottlePositionActuatorSent</SHORT-
      NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
```

```
    <PORT-REF DEST="P-PORT-PROTOTYPE">/Components/
      ThrottleActuator/PUnlimitedThrottlePositionActuator</PORT-
      REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
      IUnlimitedThrottlePosition/UnlimitedThrottlePosition</DATA
      -ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
      PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg0</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/
      RAcceleratorPedalPositionSensorReceived</STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/PAcceleratorPedalPositionSent</
      RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming1/TimingChain1Seg0</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg1</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/PAcceleratorPedalPositionSent</
      STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/RAcceleratorPedalPositionReceived
      </RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming1/TimingChain1Seg1</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg1_1</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/RAcceleratorPedalPositionReceived
      </STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/PVotedPedalPositionSent</RESPONSE
      -REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming1/TimingChain1Seg1_1</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg2</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/PVotedPedalPositionSent</STIMULUS
      -REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/RVotedPedalPositionReceived</
      RESPONSE-REF>
```

```
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming1/TimingChain1Seg2</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg2_1</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/RVotedPedalPositionReceived</
      STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/PUnlimitedThrottlePositionSent</
      RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming1/TimingChain1Seg2_1</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg3</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/PUnlimitedThrottlePositionSent</
      STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/
      RUnlimitedThrottlePositionReceived</RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming1/TimingChain1Seg3</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg4</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/
      RUnlimitedThrottlePositionReceived</STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/
      PUnlimitedThrottlePositionActuatorSent</RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming1/TimingChain1Seg4</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1AllSeg</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/
      RAcceleratorPedalPositionSensorReceived</STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/
      PUnlimitedThrottlePositionActuatorSent</RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming1/TimingChain1Seg0</SEGMENT-REF>
```

```
            <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
               EngineControlVfbTiming1/TimingChain1Seg1</SEGMENT-REF>
            <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
               EngineControlVfbTiming1/TimingChain1Seg1_1</SEGMENT-REF>
            <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
               EngineControlVfbTiming1/TimingChain1Seg2</SEGMENT-REF>
            <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
               EngineControlVfbTiming1/TimingChain1Seg2_1</SEGMENT-REF>
            <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
               EngineControlVfbTiming1/TimingChain1Seg3</SEGMENT-REF>
            <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
               EngineControlVfbTiming1/TimingChain1Seg4</SEGMENT-REF>
          </SEGMENT-REFS>
        </TIMING-DESCRIPTION-EVENT-CHAIN>
      </TIMING-DESCRIPTIONS>
      <TIMING-REQUIREMENTS>
        <LATENCY-TIMING-CONSTRAINT>
          <SHORT-NAME>TimingChain1AllSegLatency</SHORT-NAME>
          <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
          <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
             EngineControlVfbTiming1/TimingChain1AllSeg</SCOPE-REF>
          <MINIMUM>
            <CSE-CODE>1</CSE-CODE>
            <CSE-CODE-FACTOR>28</CSE-CODE-FACTOR>
          </MINIMUM>
          <MAXIMUM>
            <CSE-CODE>1</CSE-CODE>
            <CSE-CODE-FACTOR>30</CSE-CODE-FACTOR>
          </MAXIMUM>
        </LATENCY-TIMING-CONSTRAINT>
      </TIMING-REQUIREMENTS>
      <COMPONENT-REF DEST="COMPOSITION-SW-COMPONENT-TYPE">/Components/
         EngineControl</COMPONENT-REF>
    </VFB-TIMING>
```

8.2.3.2 Requirement 2

Requirement 2 specifies a typical timing constraint concerning the age of data provided by a sensor. For calculation in the AtomicSwComponentType called ”BaseFuelMass” — which is here chosen as an example of the application software — a maximum age of input data concerning the throttle angle must be guaranteed. The sensor value is determined in the SensorActuatorSwComponentType called ”ThrottleSensor” and is passed to the application software. Figure 8.4 shows all involved software components.

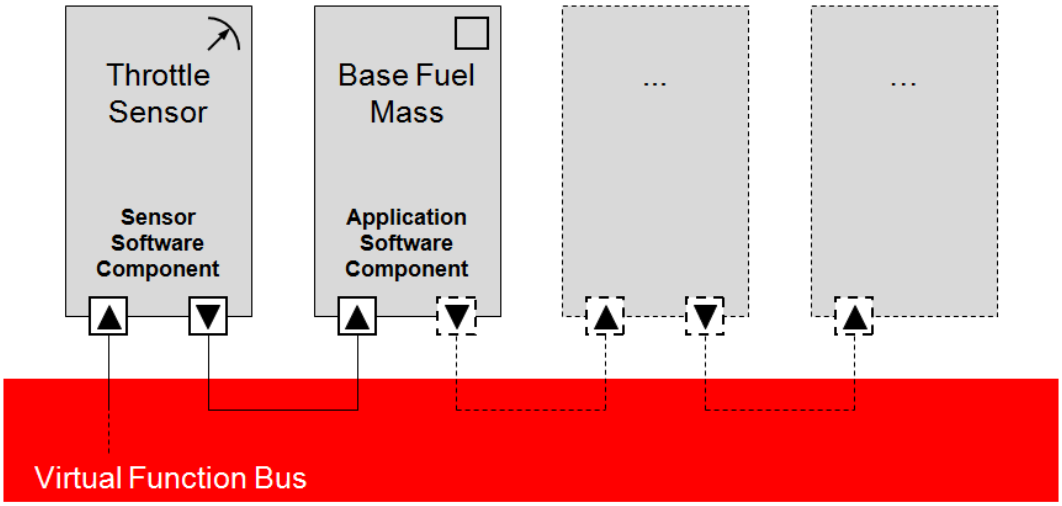


Figure 8.4: Involved components for signal flow from ”ThrottleSensor” to application component ”BaseFuelMass” for timing requirement 2.

Please note that even if the signal flow continuous to other parts of the system, it is possible to specify only this aspect of the desired timing behavior as shown in listing 8.8.

Listing 8.8: Event Definitions and Constraints for Requirement 2

```
<VFB-TIMING>
  <SHORT-NAME>EngineControlVfbTiming2</SHORT-NAME>
  <TIMING-DESCRIPTIONS>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>RThrottlePositionSensorReceived</SHORT-NAME>
      <IS-EXTERNAL>false</IS-EXTERNAL>
      <PORT-REF DEST="R-PORT-PROTOTYPE">/Components/ThrottleSensor/
          RThrottlePositionSensor</PORT-REF>
      <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
          IThrottlePositionSensor/ThrottlePositionSensor</DATA-
          ELEMENT-REF>
      <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
          PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
    </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>PThrottlePositionSent</SHORT-NAME>
      <IS-EXTERNAL>false</IS-EXTERNAL>
      <PORT-REF DEST="P-PORT-PROTOTYPE">/Components/ThrottleSensor/
          PThrottlePosition</PORT-REF>
      <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
          IThrottlePosition/ThrottlePosition</DATA-ELEMENT-REF>
      <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
          PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
    </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>RThrottlePositionReceived</SHORT-NAME>
      <IS-EXTERNAL>false</IS-EXTERNAL>
      <PORT-REF DEST="R-PORT-PROTOTYPE">/Components/BaseFuelMass/
          RThrottlePosition</PORT-REF>
```

```
        <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
            IThrottlePosition/ThrottlePosition</DATA-ELEMENT-REF>
        <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
            PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
      </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <TIMING-DESCRIPTION-EVENT-CHAIN>
        <SHORT-NAME>TimingChain2Seg0</SHORT-NAME>
        <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
            /EngineControlVfbTiming2/RThrottlePositionSensorReceived</
            STIMULUS-REF>
        <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
            /EngineControlVfbTiming2/PThrottlePositionSent</RESPONSE-
            REF>
        <SEGMENT-REFS>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
              EngineControlVfbTiming2/TimingChain2Seg0</SEGMENT-REF>
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
      <TIMING-DESCRIPTION-EVENT-CHAIN>
        <SHORT-NAME>TimingChain2Seg1</SHORT-NAME>
        <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
            /EngineControlVfbTiming2/PThrottlePositionSent</STIMULUS-
            REF>
        <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
            /EngineControlVfbTiming2/RThrottlePositionReceived</
            RESPONSE-REF>
        <SEGMENT-REFS>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
              EngineControlVfbTiming2/TimingChain2Seg1</SEGMENT-REF>
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
      <TIMING-DESCRIPTION-EVENT-CHAIN>
        <SHORT-NAME>TimingChain2AllSeg</SHORT-NAME>
        <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
            /EngineControlVfbTiming2/RThrottlePositionSensorReceived</
            STIMULUS-REF>
        <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
            /EngineControlVfbTiming2/RThrottlePositionReceived</
            RESPONSE-REF>
        <SEGMENT-REFS>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
              EngineControlVfbTiming2/TimingChain2Seg0</SEGMENT-REF>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
              EngineControlVfbTiming2/TimingChain2Seg1</SEGMENT-REF>
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
    </TIMING-DESCRIPTIONS>
    <TIMING-REQUIREMENTS>
      <LATENCY-TIMING-CONSTRAINT>
        <SHORT-NAME>TimingChain2AllSegLatency</SHORT-NAME>
        <LATENCY-CONSTRAINT-TYPE>AGE</LATENCY-CONSTRAINT-TYPE>
        <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
            EngineControlVfbTiming2/TimingChain2Seg1</SCOPE-REF>
        <MAXIMUM>
          <CSE-CODE>3</CSE-CODE>
          <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
```

```
        </MAXIMUM>
      </LATENCY-TIMING-CONSTRAINT>
    </TIMING-REQUIREMENTS>
  </VFB-TIMING>
```

8.2.3.3 Requirement 3

In requirement 3 a more complex timing description event chain is constrained. The first part of the event chain is already defined in the context of requirement 1. Thus, one can reference the set of defined events as well as the already specified timing description event chains. The second part of the event chain captures the feedback in the system that observes the sensor, in particular ”ThrottleSensor”. Please note that all events must have a functional dependency, so it is important to understand that the `SensorActuatorSwComponentType` ”ThrottleSensor” must utilize up-to-date information of the `SensorActuatorSwComponentType` ”ThrottleActuator”. Figure 8.5 shows the entire signal path and listing 8.9 presents the entire timing information consisting of timing descriptions and timing constraints.

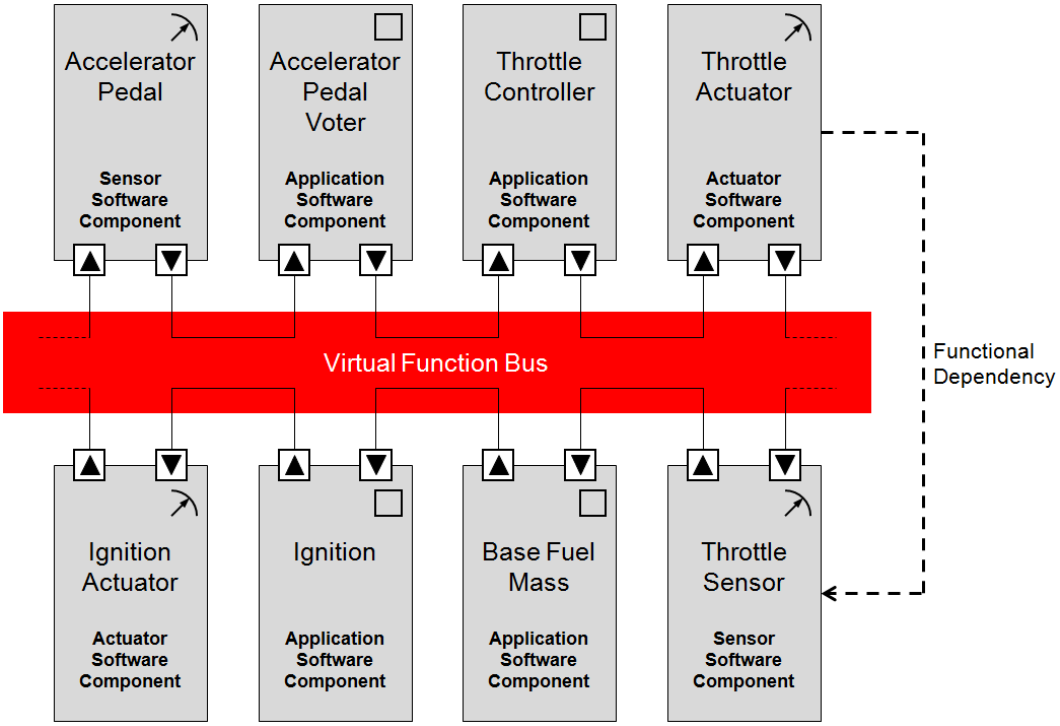


Figure 8.5: Involved components for signal flow from sensor software component ”AccelleratorPedal” to actuator software component ”IgnitionActuator” for timing requirement 3.

Listing 8.9: Event Definitions and Constraints for Requirement 3

```
<VFB-TIMING>
  <SHORT-NAME>EngineControlVfbTiming3</SHORT-NAME>
  <TIMING-DESCRIPTIONS>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
```

```
    <SHORT-NAME>PMafRateOutSent</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="P-PORT-PROTOTYPE">/Components/BaseFuelMass/
        PMafRateOut</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
        IMafRateOut/MafRateOut</DATA-ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
        PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>RMafRateOutReceived</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="R-PORT-PROTOTYPE">/Components/Ignition/
        RMafRateOut</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
        IMafRateOut/MafRateOut</DATA-ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
        PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>PIgnitionTimingSent</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="P-PORT-PROTOTYPE">/Components/Ignition/
        PIgnitionTiming</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
        IIgnitionTiming/IgnitionTiming</DATA-ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
        PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>RIgnitionTimingReceived</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="R-PORT-PROTOTYPE">/Components/
        IngnitionActuator/RIgnitionTiming</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/
        IIgnitionTiming/IgnitionTiming</DATA-ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-
        PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain3Seg0</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
        /EngineControlVfbTiming1/
        RUnlimitedThrottlePositionReceived</STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
        /EngineControlVfbTiming3/PMafRateOutSent</RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
          EngineControlVfbTiming3/TimingChain3Seg0</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain3Seg1</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
        /EngineControlVfbTiming3/PMafRateOutSent</STIMULUS-REF>
```

```
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming3/RMafRateOutReceived</RESPONSE-REF
      >
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming3/TimingChain3Seg1_1</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain3Seg1_1</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming3/RMafRateOutReceived</STIMULUS-REF
      >
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming3/PIgnitionTimingSent</RESPONSE-REF
      >
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming3/TimingChain3Seg1_1</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain3Seg2</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming3/PIgnitionTimingSent</STIMULUS-REF
      >
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming3/RIgnitionTimingReceived</RESPONSE
      -REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming3/TimingChain3Seg2</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain3AllSeg</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming1/
      RAcceleratorPedalPositionSensorReceived</STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing
      /EngineControlVfbTiming3/RIgnitionTimingReceived</RESPONSE
      -REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming1/TimingChain1AllSeg</SEGMENT-REF>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming2/TimingChain2Seg0</SEGMENT-REF>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming2/TimingChain2Seg1</SEGMENT-REF>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming3/TimingChain3Seg0</SEGMENT-REF>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming3/TimingChain3Seg1</SEGMENT-REF>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
        EngineControlVfbTiming3/TimingChain3Seg1_1</SEGMENT-REF>
```

```
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
             EngineControlVfbTiming3/TimingChain3Seg2</SEGMENT-REF>
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
    </TIMING-DESCRIPTIONS>
    <TIMING-GUARANTEES>
      <LATENCY-TIMING-CONSTRAINT>
        <SHORT-NAME>TimingChain3AllSegLatency</SHORT-NAME>
        <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
        <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
           EngineControlVfbTiming3/TimingChain3AllSeg</SCOPE-REF>
        <MAXIMUM>
          <CSE-CODE>3</CSE-CODE>
          <CSE-CODE-FACTOR>50</CSE-CODE-FACTOR>
        </MAXIMUM>
      </LATENCY-TIMING-CONSTRAINT>
    </TIMING-GUARANTEES>
    <TIMING-REQUIREMENTS>
      <OFFSET-TIMING-CONSTRAINT>
        <SHORT-NAME>Otc</SHORT-NAME>
        <SOURCE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/
           EngineControlVfbTiming1/
           PUnlimitedThrottlePositionActuatorSent</SOURCE-REF>
        <TARGET-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/
           EngineControlVfbTiming2/RThrottlePositionReceived</TARGET-
           REF>
        <MAXIMUM>
          <CSE-CODE>3</CSE-CODE>
          <CSE-CODE-FACTOR>5</CSE-CODE-FACTOR>
        </MAXIMUM>
      </OFFSET-TIMING-CONSTRAINT>
      <LATENCY-TIMING-CONSTRAINT>
        <SHORT-NAME>TimingChain3AllSegLatency</SHORT-NAME>
        <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
        <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
           EngineControlVfbTiming3/TimingChain3AllSeg</SCOPE-REF>
        <MAXIMUM>
          <CSE-CODE>3</CSE-CODE>
          <CSE-CODE-FACTOR>50</CSE-CODE-FACTOR>
        </MAXIMUM>
      </LATENCY-TIMING-CONSTRAINT>
    </TIMING-REQUIREMENTS>
  </VFB-TIMING>
```

8.2.4 Formal description of timing constraints in ECU View

Since requirement 4 references to events that are related to basic software modules, namely the interrupt system, the events must be defined in the scope of the ECU View respectively ECU Timing (EcuTiming).

8.2.4.1 Requirement 4

The stimulus event of the timing description event chain for this requirements is the start of the `BswInterruptEntity` of the basic software module called ”Camshaft”. And as a result the runnable entity of the software component ”IgnitionActuator” is activated. The response event of the timing description event chain is the termination of the `RunnableEntity` belonging to the software component ”IgnitionActuator” as shown in Figure 8.6.

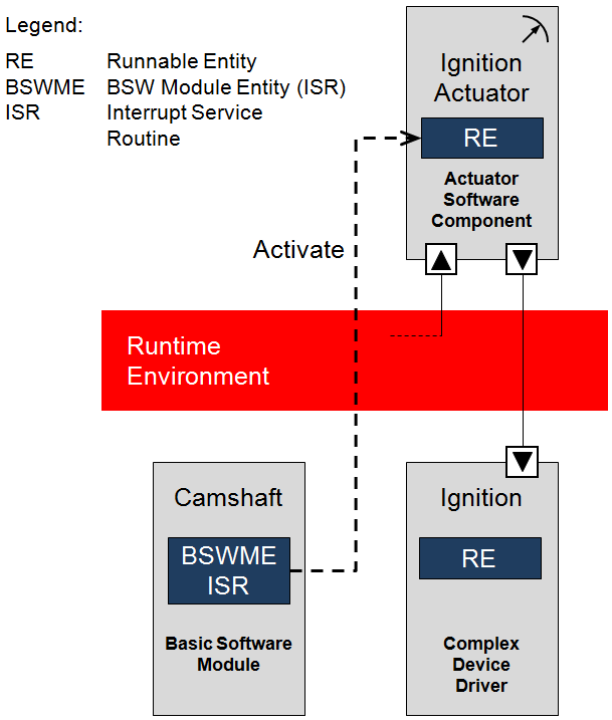


Figure 8.6: Involved components and control flow for timing requirement 4.

The timing description events and event chains for this case are presented in listing 8.10.

Listing 8.10: Event Definitions and Constraints for Requirement 4

```
<ECU-TIMING>
  <SHORT-NAME>EngineControlEcuTiming</SHORT-NAME>
  <TIMING-DESCRIPTIONS>
    <TD-EVENT-BSW-INTERNAL-BEHAVIOR>
      <SHORT-NAME>CamShaftISRActivated</SHORT-NAME>
      <BSW-MODULE-ENTITY-REF DEST="BSW-INTERRUPT-ENTITY">/Modules/
          CamShaft/CamShaftBehavior/CamShaftISR</BSW-MODULE-ENTITY-
          REF>
      <TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>BSW-MODULE-ENTITY-
          ACTIVATED</TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>
    </TD-EVENT-BSW-INTERNAL-BEHAVIOR>
    <TD-EVENT-BSW-INTERNAL-BEHAVIOR>
      <SHORT-NAME>CamShaftISRStarted</SHORT-NAME>
```

```
    <BSW-MODULE-ENTITY-REF DEST="BSW-INTERRUPT-ENTITY">/Modules/
        CamShaft/CamShaftBehavior/CamShaftISR</BSW-MODULE-ENTITY-
        REF>
    <TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>BSW-MODULE-ENTITY-
        STARTED</TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>
  </TD-EVENT-BSW-INTERNAL-BEHAVIOR>
  <TD-EVENT-BSW-INTERNAL-BEHAVIOR>
    <SHORT-NAME>CamShaftIsrTerminated</SHORT-NAME>
    <BSW-MODULE-ENTITY-REF DEST="BSW-INTERRUPT-ENTITY">/Modules/
        CamShaft/CamShaftBehavior/CamShaftISR</BSW-MODULE-ENTITY-
        REF>
    <TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>BSW-MODULE-ENTITY-
        TERMINATED</TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>
  </TD-EVENT-BSW-INTERNAL-BEHAVIOR>
  <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
    <SHORT-NAME>IgnitionActuatorCalculationActivated</SHORT-NAME>
    <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/Components/
        IngnitionActuator/IgnitionActuatorBehavior/
        IngnitionActuatorCalculation</RUNNABLE-REF>
    <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-
        ACTIVATED</TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
  </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
  <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
    <SHORT-NAME>IgnitionActuatorCalculationStarted</SHORT-NAME>
    <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/Components/
        IngnitionActuator/IgnitionActuatorBehavior/
        IngnitionActuatorCalculation</RUNNABLE-REF>
    <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-STARTED<
        /TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
  </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
  <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
    <SHORT-NAME>IgnitionActuatorCalculationTerminated</SHORT-NAME
        >
    <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/Components/
        IngnitionActuator/IgnitionActuatorBehavior/
        IngnitionActuatorCalculation</RUNNABLE-REF>
    <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-
        TERMINATED</TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
  </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain4Seg1</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-BSW-INTERNAL-BEHAVIOR">/Timing/
        EngineControlEcuTiming/CamShaftISRStarted</STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-BSW-INTERNAL-BEHAVIOR">/Timing/
        EngineControlEcuTiming/CamShaftIsrTerminated</RESPONSE-REF
        >
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
          EngineControlEcuTiming/TimingChain4Seg1</SEGMENT-REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain4Seg2</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-BSW-INTERNAL-BEHAVIOR">/Timing/
        EngineControlEcuTiming/CamShaftIsrTerminated</STIMULUS-REF
        >
```

```
        <RESPONSE-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/Timing/
          EngineControlEcuTiming/IgnitionActuatorCalculationStarted<
          /RESPONSE-REF>
        <SEGMENT-REFS>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
            EngineControlEcuTiming/TimingChain4Seg2</SEGMENT-REF>
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
      <TIMING-DESCRIPTION-EVENT-CHAIN>
        <SHORT-NAME>TimingChain4Seg3</SHORT-NAME>
        <STIMULUS-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/Timing/
          EngineControlEcuTiming/IgnitionActuatorCalculationStarted<
          /STIMULUS-REF>
        <RESPONSE-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/Timing/
          EngineControlEcuTiming/
          IgnitionActuatorCalculationTerminated</RESPONSE-REF>
        <SEGMENT-REFS>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
            EngineControlEcuTiming/TimingChain4Seg3</SEGMENT-REF>
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
      <TIMING-DESCRIPTION-EVENT-CHAIN>
        <SHORT-NAME>TimingChain4AllSeg</SHORT-NAME>
        <STIMULUS-REF DEST="TD-EVENT-BSW-INTERNAL-BEHAVIOR">/Timing/
          EngineControlEcuTiming/CamShaftISRStarted</STIMULUS-REF>
        <RESPONSE-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/Timing/
          EngineControlEcuTiming/
          IgnitionActuatorCalculationTerminated</RESPONSE-REF>
        <SEGMENT-REFS>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
            EngineControlEcuTiming/TimingChain4Seg1</SEGMENT-REF>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
            EngineControlEcuTiming/TimingChain4Seg2</SEGMENT-REF>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
            EngineControlEcuTiming/TimingChain4Seg3</SEGMENT-REF>
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
    </TIMING-DESCRIPTIONS>
    <TIMING-REQUIREMENTS>
      <LATENCY-TIMING-CONSTRAINT>
        <SHORT-NAME>TimingChain4AllSegLatency</SHORT-NAME>
        <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
        <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
          EngineControlEcuTiming/TimingChain4AllSeg</SCOPE-REF>
        <MAXIMUM>
          <CSE-CODE>3</CSE-CODE>
          <CSE-CODE-FACTOR>3</CSE-CODE-FACTOR>
        </MAXIMUM>
      </LATENCY-TIMING-CONSTRAINT>
    </TIMING-REQUIREMENTS>
  </ECU-TIMING>
```

8.2.5 Formal description of timing constraints in SW-C View

Requirement 5 refers to execution behavior of a software component's RunnableEntity and therefore the scope is the Software Component (SW-C) View respectively Software Component (SW-C) Timing (SwcTiming).

8.2.5.1 Requirement 5

The SW-C timing references the internal behavior of RunnableEntity of SW-Cs. Here one reference the RunnableEntity of the software component "Ignition" which is a ComplexDeviceDriverSwComponentType. In essence, the stated timing requirement requires firstly that the delay between activation and termination of the RunnableEntity is less than or equal 20 ms and secondly that the RunnableEntity is triggered at a frequency of 50 Hz which means that the runnable entity is periodically activated every 20 ms.

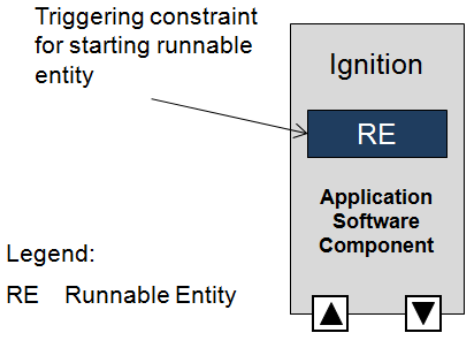


Figure 8.7: Involved component and control flow for timing requirement 5.

Listing 8.11: Event Definitions and Constraints for Requirement 5

```
<SWC-TIMING>
  <SHORT-NAME>EngineControlSwcTimingIgnition</SHORT-NAME>
  <CATEGORY />
  <TIMING-DESCRIPTIONS>
    <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
      <SHORT-NAME>IgnitionTimingCalculationActivated</SHORT-NAME>
      <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/Components/Ignition/
          IngnitionBehavior/IgnitionTimingCalculation</RUNNABLE-REF>
      <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-
          ACTIVATED</TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
    </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
    <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
      <SHORT-NAME>IgnitionTimingCalculationStarted</SHORT-NAME>
      <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/Components/Ignition/
          IngnitionBehavior/IgnitionTimingCalculation</RUNNABLE-REF>
      <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-STARTED<
          /TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
    </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
    <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
      <SHORT-NAME>IgnitionTimingCalculationTerminated</SHORT-NAME>
```

```
        <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/Components/Ignition/
            IngnitionBehavior/IgnitionTimingCalculation</RUNNABLE-REF>
        <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-
            TERMINATED</TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
      </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
      <TIMING-DESCRIPTION-EVENT-CHAIN>
        <SHORT-NAME>TimingChain5</SHORT-NAME>
        <STIMULUS-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/Timing/
            EngineControlSwcTimingIgnition/
            IgnitionTimingCalculationActivated</STIMULUS-REF>
        <RESPONSE-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/Timing/
            EngineControlSwcTimingIgnition/
            IgnitionTimingCalculationTerminated</RESPONSE-REF>
        <SEGMENT-REFS>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
              EngineControlSwcTimingIgnition/TimingChain5</SEGMENT-REF
              >
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
    </TIMING-DESCRIPTIONS>
    <TIMING-REQUIREMENTS>
      <PERIODIC-EVENT-TRIGGERING>
        <SHORT-NAME>CalculateIgnitionTimingTriggeringConstraint</
            SHORT-NAME>
        <JITTER>
          <CSE-CODE>3</CSE-CODE>
          <CSE-CODE-FACTOR>1</CSE-CODE-FACTOR>
        </JITTER>
        <PERIOD>
          <CSE-CODE>3</CSE-CODE>
          <CSE-CODE-FACTOR>20</CSE-CODE-FACTOR>
        </PERIOD>
      </PERIODIC-EVENT-TRIGGERING>
      <LATENCY-TIMING-CONSTRAINT>
        <SHORT-NAME>TimingChain5Latency</SHORT-NAME>
        <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
        <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/
            EngineControlSwcTimingIgnition/TimingChain5</SCOPE-REF>
        <MAXIMUM>
          <CSE-CODE>3</CSE-CODE>
          <CSE-CODE-FACTOR>20</CSE-CODE-FACTOR>
        </MAXIMUM>
      </LATENCY-TIMING-CONSTRAINT>
    </TIMING-REQUIREMENTS>
    <BEHAVIOR-REF DEST="SWC-INTERNAL-BEHAVIOR">/Components/Ignition/
        IngnitionBehavior</BEHAVIOR-REF>
  </SWC-TIMING>
```

8.3 Describing and Constraining Sensor and Actuator Timing

Chapter 5.1 describes the specification of VFB timing description events and introduces the attribute `isExternal` of such events. If the attribute is set to TRUE,

then the event is considered to be *external*, which means that the event is supposed to occur on the physical sensor and/or actuator a SensorActuatorSwComponentType, a ComplexDeviceDriverSwComponentType and EcuAbstractionSwComponentType is dealing with. This chapter describes how this attribute is used to describe events for sensor and actuator timing, how the different events of such kind relate to each other in event chains, and how the timing can be constrained using TimingConstraints.

One of the important purposes of the Timing Extensions is to specify end-to-end timing constraints already in early development phases. However, in the VFB view there does not exist such elements like physical sensors, physical actuators, or other hardware related elements to attach events to. Therefore, timing description events related to the VFB View can be used to declare ”external” events. For sensor and actuator timing four cases can be distinguished: external events can be observed between a SensorActuatorSwComponentType and a ComplexDeviceDriverSwComponentType, as well as between a SensorActuatorSwComponentType and an EcuAbstractionSwComponentType.

8.3.1 External Event of a Sensor accessed via S/R

In this case the SensorActuatorSwComponentType receives data from the EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType through a sender-receiver interface via its required port. Two events TDEventVariableDataPrototype, indicating the receipt of data, are specified and both referencing the same required port and pointing to the the same VariableDataPrototype. The attribute isExternal of one of those events is set to TRUE and the same attribute of the other event is set to FALSE.

The semantics of the external event is that it occurs at the hardware level. The semantics of the other event is that it indicates the receipt of the data via the corresponding required port of the SensorActuatorSwComponentType. And the notion is that the external event occurs before the event indicating the receipt of data.

8.3.2 External Event of an Actuator accessed via S/R

In this case the SensorActuatorSwComponentType sends data to the EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType through a S/R interface via its provided port. Two events TDEventVariableDataPrototype, indicating the sending of data, are specified and both referencing the same provided port and pointing to the the same VariableDataPrototype. The attribute isExternal of one of those events is set to TRUE and the same attribute of the other event is set to FALSE.

The semantics of the external event is that it occurs at the hardware level. The semantics of the other event is that it indicates the sending of the data via the corresponding

provided port of the SensorActuatorSwComponentType. And the notion is that the event indicating the sending of data occurs before the external event.

8.3.3 External Event of a Sensor accessed via C/S

In this case the SensorActuatorSwComponentType receives data from the EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType through a C/S interface on its required port. Two events TDEventOperation, indicating the receipt of the results of such an operation call, are specified and both referencing the same required port and pointing to the the same ClientServerOperation. The attribute isExternal of one of those events is set to TRUE and the same attribute of the other event is set to FALSE.

The semantics of the external event is that it occurs at the hardware level. The semantics of the other event is that it indicates the receipt of the data via the corresponding required port of the SensorActuatorSwComponentType. And the notion is that the external event occurs before the event indicating the receipt of the result of the operation call.

8.3.4 External Event of an Actuator accessed via C/S

In this case the SensorActuatorSwComponentType sends data to the EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType through a C/S interface on its required port.

Two events TDEventOperation, indicating the invocation of the such an operation call, are specified and both referencing the same required port and pointing to the the same ClientServerOperation. The attribute isExternal of one of those events is set to TRUE and the same attribute of the other event is set to FALSE.

The semantics of the external event is that it occurs at the hardware level. The semantics of the other event is that it indicates the receipt of the data via the corresponding required port of the SensorActuatorSwComponentType. And the notion is that the event indicating the invocation of the operation occurs before the external event.

8.3.5 Considering hardware I/O latency of EventChains at VFB-level

To express an end-to-end sensor or actuator timing description event chain that also comprises hardware related latencies, already at VFB level, it is necessary to set the attribute isExternal of the stimulus and/or response accordingly. The overall end-to-end timing description event chain thus also comprises the ”Input Latency” and/or the ”Output Latency”.

8.3.5.1 Input latency

The input latency is defined as the time latency between the point in time where the data is generated by a hardware I/O (e.g. a physical sensor) and the point in time where it is available for the application component, e.g. a SensorActuatorSwComponentType. The input latency is the time between the two events described in 8.3.1 and 8.3.3, respectively, depending on the communication type.

8.3.5.2 Output latency

The output latency is defined as the time latency between the point in time where the data is sent by the application component, e.g. a SensorActuatorSwComponentType, and the point in time where it is consumed by a hardware I/O (e.g. a physical actuator). The output latency is the time between the two events described in 8.3.2 and 8.3.4, respectively, depending on the communication type.

8.3.6 Constraining Input or Output Latency

The input or output latency can, for example, be modeled as event chain playing the role of a segment of the overall end-to-end chain. The overall end-to-end chain and also the input and output event chain segments can have attached timing constraints. This way either the overall end-to-end timing behavior or only the input and output behavior including hardware delay can be constrained already at VFB-level.

A History of Constraints and Specification Items

A.1 Constraint History of this Document related to AUTOSAR R4.0.1

A.1.1 Changed Constraints in R4.0.1

No constraints were changed in this release.

A.1.2 Added Constraints in R4.0.1

No constraints were added in this release.

A.1.3 Deleted Constraints in R4.0.1

No constraints were deleted in this release.

A.2 Constraint History of this Document related to AUTOSAR R4.0.2

A.2.1 Changed Constraints in R4.0.2

No constraints were changed in this release.

A.2.2 Added Constraints in R4.0.2

No constraints were added in this release.

A.2.3 Deleted Constraints in R4.0.2

No constraints were deleted in this release.

A.3 Constraint History of this Document related to AUTOSAR R4.0.3

A.3.1 Changed Constraints in R4.0.3

No constraints were changed in this release.

A.3.2 Added Constraints in R4.0.3

The constraints listed in the table below were added in this release.

Number	Heading
[constr_4500]	Restricted usage of functions
[constr_4501]	Application rule for the occurrence expression
[constr_4502]	Use references only as function operands
[constr_4503]	Restricted usage of `AutosarOperationArgumentInstance` for Content Filter
[constr_4504]	Restricted usage of `AgeConstraint`
[constr_4505]	Specifying minimum and maximum number of occurrences
[constr_4506]	Specifying minimum inter-arrival time and pattern length
[constr_4507]	Specifying pattern length, pattern jitter and patter period

Table A.1: Added Constraints in R4.0.3

A.3.3 Deleted Constraints in R 4.0.3

No constraints were changed in this release.

A.4 Constraint History of this Document related to AUTOSAR R4.1.1

A.4.1 Changed Constraints in R4.1.1

The constraints listed in the table below were changed in this release.

Number	Heading
[constr_4504]	The AgeConstraint is no longer restricted to events of type TDEventVariable-DataPrototype only. All events indicating the receipt and reading of data can be referenced by the AgeConstraint.

Table A.2: Changed Constraints in R4.1.1

A.4.2 Added Constraints in R4.1.1

The constraints listed in the table below were added in this release.

Number	Heading
[constr_4508]	TDEventVfb shall reference PortPrototypeBlueprint only in Blueprints
[constr_4509]	Only VfbTiming shall be a Blueprint
[constr_4510]	Specifying references to RunnableEntity and VariableAccess
[constr_4511]	Validity of referencing RunnableEntity
[constr_4512]	Validity of referencing VariableAccess
[constr_4513]	SynchronizationTimingConstraint shall reference at least two events
[constr_4514]	SynchronizationTimingConstraint shall reference at least two event chains
[constr_4515]	Specifying stimulus and response in TimingDescriptionEventChain
[constr_4516]	Specifying event chain segments
[constr_4517]	Referencing no further event chain segments
[constr_4518]	Specifying stimulus event and response event of first and last event chain segment
[constr_4519]	Specifying patternLength
[constr_4520]	Specifying attribute synchronizationConstraintType
[constr_4521]	Specifying attribute synchronizationConstraintType
[constr_4522]	SynchronizationTimingConstraint shall either reference events or event chains
[constr_4523]	Specifying attributes maxCycles and maxSlots
[constr_4524]	Referencing TimingDescriptionEvent
[constr_4525]	Precedence of successor relationships successor and directSuccessor
[constr_4526]	Specifying maxCycles and maxSlots in a Repetitive Execution Order Constraint
[constr_4527]	Referencing TimingDescriptionEvent in a Repetitive Execution Order Constraint
[constr_4528]	The *root* EOCExecutableEntityRefGroup shall reference only EOCExecutableEntityRefGroups
[constr_4529]	Number of nested elements referenced by the *root* EOCExecutableEntityRef-Group

[constr_4530]	An EOCExecutableEntityRefGroup representing a cycle shall reference only EOCExecutableEntityRefs
[constr_4531]	Number of nested elements referenced by EOCExecutableEntityRefGroup representing a cycle

Table A.3: Added Constraints in R4.1.1

A.4.3 Deleted Constraints in R4.1.1

No constraints were deleted in this release.

A.5 Constraint History of this Document related to AUTOSAR R4.1.2

A.5.1 Changed Constraints in R4.1.2

No constraints were changed in this release.

A.5.2 Added Constraints in R4.1.2

The constraints listed in the table below were added in this release.

Number	Heading
[constr_4532]	Successor relationship is not self-referencing
[constr_4533]	Maximum number of successor relationships
[constr_4534]	Maximum number of directSuccessor relationships
[constr_4535]	Same Mode of any ExecutableEntity
[constr_4536]	Compatible recurrence of any ExecutableEntity
[constr_4537]	References among elements in an ExecutionOrderConstraint
[constr_4538]	Hierarchical Execution Order Constraint: EOCExecutableEntityRef and EOCExecutableEntityRef shall be target or source of a successor relationship
[constr_4539]	The successor relationships successor and directSuccessor shall not be used
[constr_4540]	maxCycles and maxSlots shall not be zero
[constr_4541]	EOCExecutableEntityRef shall reference ExecutableEntity in Ordinary Execution Order Constraint
[constr_4542]	EOCExecutableEntityRef shall reference ExecutableEntity in Hierarchical Execution Order Constraint
[constr_4543]	Maximum value of the parameter minimumInterArrivalTime

Table A.4: Added Constraints in R4.1.2

A.5.3 Deleted Constraints in R4.1.2

No constraints were deleted in this release.

A.6 Constraint History of this Document related to AUTOSAR R4.1.3

A.6.1 Changed Constraints in R4.1.3

No constraints were changed in this release.

A.6.2 Added Constraints in R4.1.3

No constraints were added in this release.

A.6.3 Deleted Constraints in R4.1.3

No constraints were deleted in this release.

A.7 Constraint History of this Document related to AUTOSAR R4.2.1

A.7.1 Changed Constraints in R4.2.1

The constraints listed in the table below were changed in this release.

Number	Heading
[constr_4528]	The *root* EOCExecutableEntityRefGroup shall reference only EOCExecutableEntityRefGroups
[constr_4530]	An EOCExecutableEntityRefGroup representing a cycle shall reference only EOCExecutableEntityRefs respectively EOCEventRefs
[constr_4533]	Maximum number of successor relationships
[constr_4534]	Maximum number of directSuccessor relationships
[constr_4535]	An ExecutionOrderConstraint needs to be consistent regarding effective modes
[constr_4536]	Compatible recurrence of any ExecutableEntity
[constr_4537]	References among elements in an ExecutionOrderConstraint

Table A.5: Changed Constraints in R4.2.1

A.7.2 Added Constraints in R4.2.1

The constraints listed in the table below were added in this release.

Number	Heading
[constr_4544]	Specifying patternLength, patternJitter and patternPeriod
[constr_4545]	Referring either ExecutableEntitys or AbstractEvents
[constr_4546]	Setting the attribute isEvent
[constr_4547]	Setting the attribute permitMultipleReferencesToEE

[constr_4548]	EOCEventRef shall reference AbstractEvent in Ordinary Execution Order Constraint
[constr_4549]	EOCEventRef shall reference AbstractEvent in Hierarchical Execution Order Constraint
[constr_4550]	A Hierarchical Execution Order Constraint shall have an unambiguous root EOCExecutableEntityRefGroup

Table A.6: Added Constraints in R4.2.1

A.7.3 Deleted Constraints in R4.2.1

No constraints were deleted in this release.

A.8 Constraint History of this Document related to AUTOSAR R4.2.2

A.8.1 Changed Constraints in R4.2.2

No constraints were changed in this release.

A.8.2 Added Constraints in R4.2.2

No constraints were added in this release.

A.8.3 Deleted Constraints in R4.2.2

No constraints were deleted in this release.

A.9 Constraint History of this Document related to AUTOSAR R4.3.0

A.9.1 Changed Constraints in R4.3.0

The constraints listed in the table below were changed in this release.

Number	Heading
[constr_4501]	Application rule for the occurrence expression

Table A.7: Added Constraints in R4.3.0

A.9.2 Added Constraints in R4.3.0

The constraints listed in the table below were added in this release.

Number	Heading
[constr_4551]	Use only Numericals in `TDEventOccurrenceExpression`
[constr_4552]	Restricted usage of `AutosarVariableInstance` for Content Filter

Table A.8: Added Constraints in R4.3.0

A.9.3 Deleted Constraints in R4.3.0

No constraints were deleted in this release.

A.10 Constraint History of this Document related to AUTOSAR R4.3.1

A.10.1 Changed Constraints in R4.3.1

No constraints were changed in this release.

A.10.2 Added Constraints in R4.3.1

No constraints were added in this release.

A.10.3 Deleted Constraints in R4.3.1

No constraints were deleted in this release.

A.11 Constraint History of this Document related to AUTOSAR R4.4.0

A.11.1 Changed Constraints in R4.4.0

No constraints were changed in this release.

A.11.2 Added Constraints in R4.4.0

The constraints listed in the table below were added in this release.

Number	Heading
[constr_4553]	Usage of optional attribute ignoreOrderAllowed
[constr_4554]	Usage of optional directed association/reference letInterval
[constr_4555]	Usage of the category value LET_RELEASE in TimingDescriptionEvent
[constr_4556]	Usage of the category value LET_TERMINATE in TimingDescriptionEvent
[constr_4557]	Usage of the category value LET_INTERVAL in TimingDescriptionEventChain
[constr_4558]	Applicability of LET semantics
[constr_4559]	category of TimingDescriptionEvent shall not be extended
[constr_4560]	category of TimingDescriptionEventChain shall not be extended

Table A.9: Added Constraints in R4.4.0

A.11.3 Deleted Constraints in R4.4.0

Number	Heading
[constr_4535]	An ExecutionOrderConstraint needs to be consistent regarding effective modes

Table A.10: Deleted Constraints in R4.4.0

A.12 Added Specification Items in R4.0.3

Number	Heading
[TPS_TIMEX_00001]	Purpose of TimingDescriptionEvent
[TPS_TIMEX_00002]	Purpose of TimingDescriptionEventChain
[TPS_TIMEX_00003]	EventTriggeringConstraint specifies occurrence behavior respectively model
[TPS_TIMEX_00004]	LatencyTimingConstraint specifies latency constraints
[TPS_TIMEX_00005]	AgeConstraint to specify age constraints
[TPS_TIMEX_00006]	SynchronizationTimingConstraint specifies synchronicity constraints
[TPS_TIMEX_00007]	ExecutionOrderConstraint specifies sequence of executing executable entities
[TPS_TIMEX_00008]	ExecutionTimeConstraint to specify execution time constraints
[TPS_TIMEX_00009]	Optional use of timing extensions
[TPS_TIMEX_00010]	PeriodicEventTriggering specifies periodic occurrences of events
[TPS_TIMEX_00011]	SporadicEventTriggering specifies sporadic occurrences of events
[TPS_TIMEX_00012]	ConcretePatternEventTriggering specifies concrete pattern of occurrences of events
[TPS_TIMEX_00013]	BurstPatternEventTriggering specifies burst of occurrences of events
[TPS_TIMEX_00014]	ArbitraryEventTriggering specifies arbitrary occurrences of an event
[TPS_TIMEX_00015]	OffsetTimingConstraint specifies offset between occurrences of events
[TPS_TIMEX_00016]	Purpose of TDEventVfb
[TPS_TIMEX_00017]	TDEventVariableDataPrototype specifies events observable at sender/receiver ports
[TPS_TIMEX_00018]	TDEventOperation specifies events obeservable at client/server ports.
[TPS_TIMEX_00019]	TDEventModeDeclaration specifies events obeservable at mode ports.
[TPS_TIMEX_00020]	TDEventSwcInternalBehavior specifies observable events of runnable entities
[TPS_TIMEX_00021]	Purpose of TDEventCom
[TPS_TIMEX_00022]	TDEventISignal specifies events related to the exchange of I-Signals
[TPS_TIMEX_00023]	TDEventIPdu specifies events related to the exchange of I-PDUs

[TPS_TIMEX_00024]	TDEventFrame specifies events related to the exchange of network frames
[TPS_TIMEX_00025]	TDEventFrClusterCycleStart specifies the event related to the start of a FlexRay communication cycle
[TPS_TIMEX_00026]	TDEventTTCanCycleStart specifies the event related to the start of a TTCAN communication cycle
[TPS_TIMEX_00027]	Purpose of TDEventComplex
[TPS_TIMEX_00028]	TDEventBswInternalBehavior specifies observable events of BSW module entities
[TPS_TIMEX_00029]	Purpose of TDEventBsw
[TPS_TIMEX_00030]	TDEventBswModule specifies observable events when basic software entries are called
[TPS_TIMEX_00031]	TDEventBswModeDeclaration specifies observable events in case of BSW mode communication
[TPS_TIMEX_00032]	Purpose of VfbTiming
[TPS_TIMEX_00033]	Purpose of SwcTiming
[TPS_TIMEX_00034]	Purpose of SystemTiming
[TPS_TIMEX_00035]	Purpose of BswModuleTiming
[TPS_TIMEX_00036]	Purpose of EcuTiming
[TPS_TIMEX_00037]	TimingConstraint is a Traceable

Table A.11: Added Specification Items in 4.0.3

A.13 Added Specification Items in R4.1.1

Number	Heading
[TPS_TIMEX_00038]	Purpose of EOCExecutableEntityRefAbstract
[TPS_TIMEX_00039]	TDEventTrigger specifies events observable at trigger ports
[TPS_TIMEX_00040]	Blueprinting VfbTiming
[TPS_TIMEX_00041]	Purpose of EOCExecutableEntityRefGroup
[TPS_TIMEX_00042]	Purpose of TDEventVfbPort
[TPS_TIMEX_00043]	Purpose of TDEventVfbReference
[TPS_TIMEX_00044]	Purpose of TDEventSwc
[TPS_TIMEX_00045]	Purpose of TDEventSwcInternalBehaviorReference
[TPS_TIMEX_00046]	Purpose of EOCExecutableEntityRef

Table A.12: Added Specification Items in 4.1.1

A.14 Added Specification Items in R4.1.2

No specification items were added in this release.

A.15 Added Specification Items in R4.1.3

No specification items were added in this release.

A.16 Added Specification Items in R4.2.1

Number	Heading
[TPS_TIMEX_00047]	Purpose of ExecutionOrderConstraintTypeEnum
[TPS_TIMEX_00048]	Purpose of EOCEventRef

Table A.13: Added Specification Items in 4.2.1

A.17 Changed Specification Items in R4.2.1

Number	Heading
[TPS_TIMEX_00007]	Added the possibility to reference RTEEvents and BswEvents in addition to ExecutableEntitys. The optional attributes executionOrderConstraintType, isEvent and permitMultipleReferencesToEE have been specified to support consistency checking of an execution order constraint.

Table A.14: Changed Specification Items in 4.2.1

A.18 Added Specification Items in R4.2.2

No specification items were added in this release.

A.19 Changed Specification Items in R4.2.2

No specification items were changed in this release.

A.20 Added Specification Items in R4.3.0

Number	Heading
[TPS_TIMEX_00049]	Purpose of TimingCondition
[TPS_TIMEX_00050]	Purpose of TimingConditionFormula
[TPS_TIMEX_00051]	Purpose of TimingExtensionResource
[TPS_TIMEX_00052]	Purpose of TDEventFrameEthernet

Table A.15: Added Specification Items in 4.3.0

A.21 Changed Specification Items in R4.3.0

No specification items were changed in this release.

A.22 Added Specification Items in R4.3.1

No specification items were added in this release.

A.23 Changed Specification Items in R4.3.1

No specification items were changed in this release.

A.24 Added Specification Items in R4.4.0

Number	Heading
[TPS_TIMEX_00053]	Purpose of `BswCompositionTiming`
[TPS_TIMEX_00054]	Purpose of `SynchronizationPointConstraint`
[TPS_TIMEX_00055]	Purpose of `letInterval`
[TPS_TIMEX_00056]	Attribute `category` used in context of `TimingDescriptionEvent` and LET
[TPS_TIMEX_00057]	Attribute `category` used in context of `TimingDescriptionEventChain` and LET

Table A.16: Added Specification Items in 4.4.0

A.25 Changed Specification Items in R4.4.0

No specification items were changed in this release.

B Mentioned Class Tables

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

Class	***ARElement*** (abstract)
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage
Note	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*
Subclasses	AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, ApplicationPartition, *AutosarDataType*, *BaseType*, BlueprintMappingSet, BswEntryRelationshipSet, BswModuleDescription, BswModuleEntry, BuildActionManifest, CalibrationParameterValueSet, ClientIdDefinitionSet, ClientServerInterfaceToBswModuleEntryBlueprintMapping, Collection, CompuMethod, ConsistencyNeedsBlueprintSet, ConstantSpecification, ConstantSpecificationMappingSet, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, DataConstr, DataExchangePoint, DataTransformationSet, DataTypeMappingSet, *DiagnosticCommonElement*, DiagnosticConnection, DiagnosticContributionSet, DiagnosticMasterToSlaveEventMappingSet, Documentation, EcucDefinitionCollection, EcucDestinationUriDefSet, EcucModuleConfigurationValues, EcucModuleDef, EcucValueCollection, EndToEndProtectionSet, EvaluatedVariantSet, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FlatMap, GeneralPurposeConnection, HwCategory, HwElement, HwType, IPv6ExtHeaderFilterSet, *Implementation*, InterpolationRoutineMappingSet, J1939ControllerApplication, KeywordSet, LifeCycleInfoSet, LifeCycleStateDefinitionGroup, McFunction, McGroup, ModeDeclarationGroup, ModeDeclarationMappingSet, PhysicalDimension, PhysicalDimensionMappingSet, *PortInterface*, PortInterfaceMappingSet, Port ▽

▽

△

Class	**ARElement** (abstract)			
	△ PrototypeBlueprint, PostBuildVariantCriterion, PostBuildVariantCriterionValueSet, PredefinedVariant, RapidPrototypingScenario, SdgDef, SwAddrMethod, SwAxisType, *SwComponentType*, SwRecord Layout, SwSystemconst, SwSystemconstantValueSet, SwcBswMapping, System, SystemSignal, System SignalGroup, TcpOptionFilterSet, *TimingExtension*, TransformationPropsSet, Unit, UnitGroup, ViewMap Set			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table B.1: ARElement

Class	**AbstractEvent** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	This meta-class represents the abstract ability to model an event that can be taken to implement application software or basic software in AUTOSAR.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*BswEvent*, *RTEEvent*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
activation Reason Representation	ExecutableEntity ActivationReason	0..1	ref	If the activationReasonRepresentation is referenced from the enclosing AbstractEvent this shall be taken as an indication that the latter contributes to the activating vector of this ExecutableEntity that owns the referenced ExecutableEntityActivationReason.

Table B.2: AbstractEvent

Class	**ApplicationSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The ApplicationSwComponentType is used to represent the application software. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
–	–	–	–	–

Table B.3: ApplicationSwComponentType

Class	**AtomicSwComponentType** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	An atomic software component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Subclasses	ApplicationSwComponentType, ComplexDeviceDriverSwComponentType, EcuAbstractionSwComponent Type, NvBlockSwComponentType, SensorActuatorSwComponentType, ServiceProxySwComponent Type, ServiceSwComponentType			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

▽

△

Class	***AtomicSwComponentType*** (abstract)			
internalBehavior	SwcInternalBehavior	0..1	aggr	The SwcInternalBehaviors owned by an AtomicSw ComponentType can be located in a different physical file. Therefore the aggregation is «atpSplitable». **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalBehavior, variationPoint.short Label vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the AtomicSw ComponentType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName

Table B.4: AtomicSwComponentType

Class	***AtpInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::AbstractStructure			
Note	An M0 instance of a classifier may be represented as a tree rooted at that instance, where under each node come the sub-trees representing the instances which act as features under that node. An instance ref specifies a navigation path from any M0 tree-instance of the base (which is a classifier) to a leaf (which is an instance of the target).			
Base	*ARObject*			
Subclasses	AnyInstanceRef, ApplicationCompositeElementInPortInterfaceInstanceRef, ApplicationDataPrototypeIn SystemInstanceRef, ComponentInCompositionInstanceRef, ComponentInSystemInstanceRef, Data PrototypeInSystemInstanceRef, InnerDataPrototypeGroupInCompositionInstanceRef, InnerPortGroupIn CompositionInstanceRef, InnerRunnableEntityGroupInCompositionInstanceRef, InstanceEventIn CompositionInstanceRef, *ModeGroupInAtomicSwcInstanceRef*, ModeInBswModuleDescriptionInstance Ref, ModeInSwcInstanceRef, OperationArgumentInComponentInstanceRef, *OperationInAtomicSwc InstanceRef*, OperationInSystemInstanceRef, PModeInSystemInstanceRef, ParameterInAtomicSWC TypeInstanceRef, PortGroupInSystemInstanceRef, *PortInCompositionTypeInstanceRef*, RModeInAtomic SwcInstanceRef, RteEventInEcuInstanceRef, RunnableEntityInCompositionInstanceRef, SwcService DependencyInSystemInstanceRef, *TriggerInAtomicSwcInstanceRef*, TriggerInSystemInstanceRef, VariableAccessInEcuInstanceRef, VariableDataPrototypeInCompositionInstanceRef, VariableData PrototypeInSystemInstanceRef, VariableInAtomicSWCTypeInstanceRef, *VariableInAtomicSwcInstance Ref*, VariableInComponentInstanceRef			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
atpBase	AtpClassifier	1	ref	This is the base from which the navigaion path starts. **Stereotypes:** atpAbstract; atpDerived
atpContextEle- ment (ordered)	AtpPrototype	*	ref	This is one particular step in the navigation path. **Stereotypes:** atpAbstract
atpTarget	AtpFeature	1	ref	This is the target of the instance ref. In other words it is the terminal of the navigation path. **Stereotypes:** atpAbstract

Table B.5: AtpInstanceRef

Primitive	**Boolean**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	A Boolean value denotes a logical condition that is either 'true' or 'false'. It can be one of "0", "1", "true", "false" **Tags:** xml.xsd.customType=BOOLEAN xml.xsd.pattern=0|1|true|false xml.xsd.type=string

Table B.6: Boolean

Class	***BswEvent*** (abstract)			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	Base class of various kinds of events which are used to trigger a BswModuleEntity of this BSW module or cluster. The event is local to the BSW module or cluster. The short name of the meta-class instance is intended as an input to configure the required API of the BSW Scheduler.			
Base	*ARObject*, *AbstractEvent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswOperationInvokedEvent, *BswScheduleEvent*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
context Limitation	BswDistinguished Partition	*	ref	The existence of this reference indicates that the usage of the event is limited to the context of the referred Bsw DistinguishedPartitions.
disabledInMode	ModeDeclaration	*	iref	The modes, in which this event is disabled. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=disabledInMode
startsOnEvent	BswModuleEntity	1	ref	The entity which is started by the event.

Table B.7: BswEvent

Class	**BswInternalBehavior**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	Specifies the behavior of a BSW module or a BSW cluster w.r.t. the code entities visible by the BSW Scheduler. It is possible to have several different BswInternalBehaviors referring to the same BswModuleDescription.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *InternalBehavior*, *Multilanguage Referrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
arTypedPer Instance Memory	VariableDataPrototype	*	aggr	Defines an AUTOSAR typed memory-block that needs to be available for each instance of the Basic Software Module. The aggregation of arTypedPerInstanceMemory is subject to variability with the purpose to support variability in the Basic Software Module's implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

△

Class	**BswInternalBehavior**			
bswPerInstance MemoryPolicy	BswPerInstance MemoryPolicy	*	aggr	Policy for a arTypedPerInstanceMemory The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
clientPolicy	BswClientPolicy	*	aggr	Policy for a requiredClientServerEntry. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=clientPolicy, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
distinguished Partition	BswDistinguished Partition	*	aggr	Indicates an abstract partition context in which the enclosing BswModuleEntity can be executed. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.ShortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
entity	BswModuleEntity	*	aggr	A code entity for which the behavior is described **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=5
event	BswEvent	*	aggr	An event required by this module behavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=10
exclusiveArea Policy	BswExclusiveArea Policy	*	aggr	Policy for an ExclusiveArea in this BswInternalBehavior. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveAreaPolicy, variation Point.shortLabel vh.latestBindingTime=preCompileTime
includedData TypeSet	IncludedDataTypeSet	*	aggr	The includedDataTypeSet is used by a basic software module for its implementation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=includedDataTypeSet
internal TriggeringPoint	BswInternalTriggering Point	*	aggr	An internal triggering point. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=2
internal TriggeringPoint Policy	BswInternalTriggering PointPolicy	*	aggr	Policy for an internalTriggeringPoint in this BswInternal Behavior.. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalTriggeringPointPolicy, variation Point.shortPoint vh.latestBindingTime=preCompileTime
modeReceiver Policy	BswModeReceiver Policy	*	aggr	Implementation policy for the reception of mode switches. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeReceiverPolicy, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25

▽

△

Class	**BswInternalBehavior**			
modeSender Policy	BswModeSenderPolicy	*	aggr	Implementation policy for providing a mode group. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeSenderPolicy, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
parameterPolicy	BswParameterPolicy	*	aggr	Policy for a perInstanceParameter in this BswInternal Behavior. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=parameterPolicy, variatioPoint.short Label vh.latestBindingTime=preCompileTime
perInstance Parameter	ParameterData Prototype	*	aggr	Describes a read only memory object containing characteristic value(s) needed by this BswInternal Behavior. The role name perInstanceParameter is chosen in analogy to the similar role in the context of SwcInternal Behavior. In contrast to constantMemory, this object is not allocated locally by the module's code, but by the BSW Scheduler and it is accessed from the BSW module via the BSW Scheduler API. The main use case is the support of software emulation of calibration data. The aggregation is subject to variability with the purpose to support implementation variants. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=atp.Splitkey shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
receptionPolicy	BswDataReception Policy	*	aggr	Data reception policy for inter-partition and/or inter-core communication. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=receptionPolicy, variationPoint.short Label vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
releasedTrigger Policy	BswReleasedTrigger Policy	*	aggr	Policy for a releasedTrigger. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=releasedTriggerPolicy, variation Point.shortLabel vh.latestBindingTime=preCompileTime
schedulerName Prefix	BswSchedulerName Prefix	*	aggr	Optional definition of one or more prefixes to be used for the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=schedulerNamePrefix, variation Point.ShortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

▽

△

Class	**BswInternalBehavior**			
sendPolicy	BswDataSendPolicy	*	aggr	Policy for a providedData. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=sendPolicy, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
service Dependency	BswService Dependency	*	aggr	Defines the requirements on AUTOSAR Services for a particular item. The aggregation is subject to variability with the purpose to support the conditional existence of ServiceNeeds. The aggregation is splitable in order to support that ServiceNeeds might be provided in later development steps. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceDependency, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
triggerDirect Implementation	BswTriggerDirect Implementation	*	aggr	Specifies a trigger to be directly implemented via OS calls. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=triggerDirectImplementation, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=15
variationPoint Proxy	VariationPointProxy	*	aggr	Proxy of a variation points in the C/C++ implementation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName

Table B.8: BswInternalBehavior

Class	**BswInterruptEntity**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	BSW module entity, which is designed to be triggered by an interrupt.			
Base	*ARObject*, *BswModuleEntity*, *ExecutableEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
interrupt Category	BswInterruptCategory	1	attr	Category of the interrupt
interruptSource	String	1	attr	Allows a textual documentation of the intended interrupt source.

Table B.9: BswInterruptEntity

Class	**BswModuleDescription**
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswOverview
Note	Root element for the description of a single BSW module or BSW cluster. In case it describes a BSW module, the short name of this element equals the name of the BSW module. **Tags:** atp.recommendedPackage=BswModuleDescriptions
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	**BswModuleDescription**			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
bswModule Dependency	BswModuleDependency	*	aggr	Describes the dependency to another BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
bswModule Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswModuleDocumentation, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=6
expectedEntry	BswModuleEntry	*	ref	Indicates an entry which is required by this module. Replacement of outgoingCallback / requiredEntry. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=expectedEntry, variationPoint.short Label vh.latestBindingTime=preCompileTime
implemented Entry	BswModuleEntry	*	ref	Specifies an entry provided by this module which can be called by other modules. This includes "main" functions, interrupt routines, and callbacks. Replacement of providedEntry / expectedCallback. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=implementedEntry, variation Point.shortLabel vh.latestBindingTime=preCompileTime
internalBehavior	BswInternalBehavior	*	aggr	The various BswInternalBehaviors associated with a Bsw ModuleDescription can be distributed over several physical files. Therefore the aggregation is «atpSplitable». **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName xml.sequenceOffset=65
moduleId	PositiveInteger	0..1	attr	Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen. **Tags:** xml.sequenceOffset=5
providedClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module provides a client server entry which can be called from another parition or core.This entry is declared locally to this context and will be connected to the requiredClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
providedData	VariableDataPrototype	*	aggr	Specifies a data prototype provided by this module in order to be read from another partition or core.The providedData is declared locally to this context and will be connected to the requiredData of another or the same module via the configuration of the BSW Scheduler. ▽

▽

△

Class	**BswModuleDescription**			
				△ **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
providedMode Group	ModeDeclarationGroup Prototype	*	aggr	A set of modes which is owned and provided by this module or cluster. It can be connected to the required ModeGroups of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with modes provided via ports by an associated ServiceSwComponentType, EcuAbstraction SwComponentType or ComplexDeviceDriverSw ComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25
releasedTrigger	Trigger	*	aggr	A Trigger released by this module or cluster. It can be connected to the requiredTriggers of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with Triggers provided via ports by an associated ServiceSwComponentType, Ecu AbstractionSwComponentType or ComplexDeviceDriver SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=35
requiredClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module requires a client server entry which can be implemented on another parition or core.This entry is declared locally to this context and will be connected to the providedClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
requiredData	VariableDataPrototype	*	aggr	Specifies a data prototype required by this module in oder to be provided from another partition or core.The required Data is declared locally to this context and will be connected to the providedData of another or the same module via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
requiredMode Group	ModeDeclarationGroup Prototype	*	aggr	Specifies that this module or cluster depends on a certain mode group. The requiredModeGroup is local to this context and will be connected to the providedModeGroup of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

▽

△

Class	**BswModuleDescription**			
requiredTrigger	Trigger	*	aggr	Specifies that this module or cluster reacts upon an external trigger.This requiredTrigger is declared locally to this context and will be connected to the providedTrigger of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

Table B.10: BswModuleDescription

Class	***BswModuleEntity*** (abstract)			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	Specifies the smallest code fragment which can be described for a BSW module or cluster within AUTOSAR.			
Base	*ARObject*, *ExecutableEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswCalledEntity, BswInterruptEntity, BswSchedulableEntity			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
accessedMode Group	ModeDeclarationGroup Prototype	*	ref	A mode group which is accessed via API call by this entity. It must be a ModeDeclarationGroupPrototype required by this module or cluster. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
activationPoint	BswInternalTriggering Point	*	ref	Activation point used by the module entity to activate one or more internal triggers. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
callPoint	BswModuleCallPoint	*	aggr	A call point used in the code of this entitiy. The variablity of this association is especially targeted at debug scenarios: It is possible to have one variant calling into the AUTOSAR debug module and another one which doesn't. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
calledEntry	BswModuleEntry	*	ref	The entry of another (or the same) BSW module which is called by this entry (usually via C function call). This information allows to set up a model of call chains. The variablity of this association is especially targeted at debug scenarios: It is possible to have one variant calling into the AUTOSAR debug module and another one which doesn't. Note that this relation has been merked as obsolete, since the more powerful definition of a callPoint should be used. **Stereotypes:** atpVariation **Tags:** atp.Status=removed vh.latestBindingTime=preCompileTime
dataReceive Point	BswVariableAccess	*	aggr	The data is received via the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

▽

△

Class	***BswModuleEntity*** (abstract)			
dataSendPoint	BswVariableAccess	*	aggr	The data is sent via the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
implemented Entry	BswModuleEntry	1	ref	The entry which is implemented by this module entity.
issuedTrigger	Trigger	*	ref	A trigger issued by this entity via BSW Scheduler API call. It must be a BswTrigger released (i.e. owned) by this module or cluster. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
managedMode Group	ModeDeclarationGroup Prototype	*	ref	A mode group which is managed by this entity. It must be a ModeDeclarationGroupPrototype provided by this module or cluster. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
schedulerName Prefix	BswSchedulerName Prefix	0..1	ref	A prefix to be used in generated names for the Bsw ModuleScheduler in the context of this BswModuleEntity, for example entry point prototypes, macros for dealing with exclusive areas, header file names. Details are defined in the SWS RTE. The prefix supersedes default rules for the prefix of those names.

Table B.11: BswModuleEntity

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

Table B.12: ClientServerOperation

Class	**ComplexDeviceDriverSwComponentType**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components
Note	The ComplexDeviceDriverSwComponentType is a special AtomicSwComponentType that has direct access to hardware on an ECU and which is therefore linked to a specific ECU or specific hardware. The ComplexDeviceDriverSwComponentType introduces the possibility to link from the software representation to its hardware description provided by the ECU Resource Template. **Tags:** atp.recommendedPackage=SwComponentTypes
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*

▽

△

Class	**ComplexDeviceDriverSwComponentType**			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hardware Element	HwDescriptionEntity	*	ref	Reference from the ComplexDeviceDriverSwComponent Type to the description of the used HwElements.

Table B.13: ComplexDeviceDriverSwComponentType

Class	**ComponentInCompositionInstanceRef**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs			
Note	The ComponentInCompositionInstanceRef points to a concrete SwComponentPrototype within a CompositionSwComponentType.			
Base	*ARObject*, *AtpInstanceRef*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
base	CompositionSw ComponentType	1	ref	**Stereotypes:** atpDerived **Tags:** xml.sequenceOffset=10
contextCompo-nent (ordered)	SwComponent Prototype	*	ref	The context for the scope of this timing event. **Tags:** xml.sequenceOffset=20
target Component	SwComponent Prototype	1	ref	**Tags:** xml.sequenceOffset=30

Table B.14: ComponentInCompositionInstanceRef

Class	**CompositionSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	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. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
component	SwComponent Prototype	*	aggr	The instantiated components that are part of this composition. The aggregation of SwComponentPrototype is subject to variability with the purpose to support the conditional existence of a SwComponentPrototype. Please be aware: if the conditional existence of SwComponentPrototypes is resolved post-build the deselected SwComponent Prototypes are still contained in the ECUs build but the instances are inactive in in that they are not scheduled by the RTE. The aggregation is marked as atpSplitable in order to allow the addition of service components to the ECU extract during the ECU integration. The use case for having 0 components owned by the CompositionSwComponentType could be to deliver an ▽

▽

△

Class	**CompositionSwComponentType**			
				△ empty CompositionSwComponentType to e.g. a supplier for filling the internal structure. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	SwConnector	*	aggr	SwConnectors have the principal ability to establish a connection among PortPrototypes. They can have many roles in the context of a CompositionSwComponentType. Details are refined by subclasses. The aggregation of SwConnectors is subject to variability with the purpose to support variant data flow. The aggregation is marked as atpSplitable in order to allow the extension of the ECU extract with AssemblySw Connectors between ApplicationSwComponentTypes and ServiceSwComponentTypes during the ECU integration. **Stereotypes:** atpSplitable; atpVariation **Tags:** 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. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the used ApplicationDataTypes in PortInterfaces. Background: when developing subsystems it may happen that ApplicationDataTypes are used on the surface of CompositionSwComponentTypes. In this case it would be reasonable to be able to also provide the intended mapping to the ImplementationDataTypes. However, this mapping shall be informal and not technically binding for the implementers mainly because the RTE generator is not concerned about the CompositionSwComponent Types. Rationale: if the mapping of ApplicationDataTypes on the delegated and inner PortPrototype matches then the mapping to ImplementationDataTypes is not impacting compatibility. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dataTypeMapping
instantiation RTEEventProps	InstantiationRTEEvent Props	*	aggr	This allows to define instantiation specific properties for RTE Events, in particular for instance specific scheduling. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortLabel, variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime

Table B.15: CompositionSwComponentType

Class	**EcuAbstractionSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The ECUAbstraction is a special AtomicSwComponentType that resides between a software-component that wants to access ECU periphery and the Microcontroller Abstraction. The EcuAbstractionSw ComponentType introduces the possibility to link from the software representation to its hardware description provided by the ECU Resource Template. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *Atp Type*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *Sw ComponentType*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
hardware Element	HwDescriptionEntity	*	ref	Reference from the EcuAbstractionComponentType to the description of the used HwElements.

Table B.16: EcuAbstractionSwComponentType

Class	**EcucValueCollection**			
Package	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
Note	This represents the anchor point of the ECU configuration description. **Tags:** atp.recommendedPackage=EcucValueCollections			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
ecuExtract	System	1	ref	Represents the extract of the System Configuration that is relevant for the ECU configured with that ECU Configuration Description.
ecucValue	EcucModule ConfigurationValues	1..*	ref	References to the configuration of individual software modules that are present on this ECU. atpVariation: [RS_ECUC_00079] **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table B.17: EcucValueCollection

Class	***ExecutableEntity*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	Abstraction of executable code.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*BswModuleEntity*, RunnableEntity			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
activation Reason	ExecutableEntity ActivationReason	*	aggr	If the ExecutableEntity provides at least one activation Reason element the RTE resp. BSW Scheduler shall provide means to read the activation vector of this executable entity execution. If no activationReason element is provided the feature of being able to determine the activating RTEEvent is disabled for this ExecutableEntity.
canEnter ExclusiveArea	ExclusiveArea	*	ref	This means that the executable entity can enter/leave the referenced exclusive area through explicit API calls.
exclusiveArea NestingOrder	ExclusiveAreaNesting Order	*	ref	This represents the set of ExclusiveAreaNestingOrders recognized by this ExecutableEntity.

▽

△

Class	***ExecutableEntity*** (abstract)			
minimumStart Interval	TimeValue	1	attr	Specifies the time in seconds by which two consecutive starts of an ExecutableEntity are guaranteed to be separated.
reentrancyLevel	ReentrancyLevelEnum	0..1	attr	The reentrancy level of this ExecutableEntity. See the documentation of the enumeration type ReentrancyLevel Enum for details. Please note that nonReentrant interfaces can have also reentrant or multicoreReentrant implementations, and reentrant interfaces can also have multicoreReentrant implementations.
runsInside ExclusiveArea	ExclusiveArea	*	ref	The executable entity runs completely inside the referenced exclusive area.
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this code entity. Via an association to the same SwAddrMethod, it can be specified that several code entities (even of different modules or components) shall be located in the same memory without already specifying the memory section itself.

Table B.18: ExecutableEntity

Class	***ExecutionTime*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ResourceConsumption::ExecutionTime			
Note	Base class for several means how to describe the ExecutionTime of software. The required context information is provided through this class.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AnalyzedExecutionTime, MeasuredExecutionTime, RoughEstimateOfExecutionTime, Simulated ExecutionTime			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
exclusiveArea	ExclusiveArea	0..1	ref	Reference to the ExclusiveArea this execution time is provided for.
executableEntity	ExecutableEntity	0..1	ref	The executable entity for which this execution time is described.
hardware Configuration	HardwareConfiguration	1	aggr	Provides information on the HardwareConfiguration used to specify this ExecutionTime.
hwElement	HwElement	0..1	ref	The hardware element (e.g. type of ECU) for which the execution time is specified.
includedLibrary	DependencyOnArtifact	*	ref	If this dependency is specified, the execution time of the library code is included in the execution time data for the runnable.
memorySection Location	MemorySectionLocation	*	aggr	Provides information on the MemorySectionLocation which is involved in the ExecutionTime description.
softwareContext	SoftwareContext	1	aggr	Provides information on the detailed SoftwareContext used to provide the ExecutionTime description.

Table B.19: ExecutionTime

Primitive	**Float**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	An instance of Float is an element from the set of real numbers. The value must comply with IEEE 754 and is limited to what can be expressed by a 64 bit binary representation. **Tags:** xml.xsd.customType=FLOAT xml.xsd.type=double

Table B.20: Float

Class	«atpMixedString» ***FormulaExpression*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::FormulaLanguage			
Note	This class represents the syntax of the formula language. The class is modeled as an abstract class in order to be specialized into particular use cases. For each use case the referable objects might be specified in the specialization.			
Base	*ARObject*			
Subclasses	CompuGenericMath, EcucConditionFormula, EcucParameterDerivationFormula, *FMFormulaByFeatures AndAttributes*, *SwSystemconstDependentFormula*, TDEventOccurrenceExpressionFormula, Timing ConditionFormula			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
atpReference	Referrable	*	ref	The referable object shall yield a numerical / boolean value. **Stereotypes:** atpAbstract
atpString Reference	Referrable	*	ref	The referable object shall yield a string value. **Stereotypes:** atpAbstract

Table B.21: FormulaExpression

Class	***FrameTriggering*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The FrameTriggering describes the instance of a frame sent on a channel and defines the manner of triggering (timing information) and identification of a frame on the channel, on which it is sent. For the same frame, if FrameTriggerings exist on more than one channel of the same cluster the fan-out/in is handled by the Bus interface.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CanFrameTriggering, EthernetFrameTriggering, FlexrayFrameTriggering, LinFrameTriggering			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
frame	Frame	1	ref	One frame can be triggered several times, e.g. on different channels. If a frame has no frame triggering, it won't be sent at all. A frame triggering has assigned exactly one frame, which it triggers.
framePort	FramePort	*	ref	References to the FramePort on every ECU of the system which sends and/or receives the frame. References for both the sender and the receiver side shall be included when the system is completely defined.

▽

△

Class	***FrameTriggering*** (abstract)			
pduTriggering	PduTriggering	*	ref	This reference provides the relationship to the Pdu Triggerings that are implemented by the FrameTriggering. The reference is optional since no PduTriggering can be defined for NmPdus and XCP Pdus. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild

Table B.22: FrameTriggering

Class	***Identifiable*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.			
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ARPackage, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractServiceInstance*, ApplicationEndpoint, ApplicationError, ApplicationPartitionToEcuPartitionMapping, AsynchronousServer CallResultPoint, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, AutosarOperationArgument Instance, AutosarVariableInstance, BswInternalTriggeringPoint, BswModuleDependency, *BuildAction Entity*, BuildActionEnvironment, CanTpAddress, CanTpChannel, CanTpNode, Chapter, ClassContent Conditional, ClientIdDefinition, ClientServerOperation, Code, *CollectableElement*, ComManagement Mapping, *CommConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, *CouplingPortStructuralElement*, *Crypto ServiceMapping*, DataPrototypeGroup, DataTransformation, DependencyOnArtifact, *DiagEvent DebounceAlgorithm*, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticFunctionInhibit Source, DiagnosticMasterToSlaveEventMapping, *DiagnosticRoutineSubfunction*, DoIpLogicAddress, EC UMapping, *EOCExecutableEntityRefAbstract*, EcuPartition, EcucContainerValue, *EcucDefinition Element*, EcucDestinationUriDef, EcucEnumerationLiteralDef, EcucQuery, EcucValidationCondition, End ToEndProtection, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMap Assertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FlatInstanceDescriptor, FlexrayArTpNode, FlexrayTpConnectionControl, FlexrayTp Node, FlexrayTpPduPool, *FrameTriggering*, GeneralParameter, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HeapUsage*, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, IPv6Ext HeaderFilterList, ISignalToIPduMapping, ISignalTriggering, *IdentCaption*, InternalTriggeringPoint, J1939 SharedAddressCluster, J1939TpNode, Keyword, LifeCycleState, LinScheduleTable, LinTpNode, Linker, MacMulticastGroup, McDataInstance, MemorySection, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, NetworkEndpoint, *NmCluster*, NmEcu, *NmNode*, NvBlockDescriptor, *Packageable Element*, ParameterAccess, PduToFrameMapping, PduTriggering, PerInstanceMemory, *Physical Channel*, PortGroup, *PortInterfaceMapping*, PossibleErrorReaction, ResourceConsumption, RootSw CompositionPrototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RunnableEntityGroup, *SdgAttribute*, SdgClass, SecureCommunicationAuthenticationProps, SecureCommunicationFreshnessProps, *ServerCallPoint*, *ServiceNeeds*, SocketAddress, SomeipTpChannel, *SpecElementReference*, *StackUsage*, Structured Req, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SwcToApplicationPartition Mapping, SwcToEcuMapping, SwcToImplMapping, SystemMapping, TcpOptionFilterList, Timing Condition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, TimingModeInstance, Tls CryptoCipherSuite, Topic1, TpAddress, TraceableText, *TracedFailure*, *TransformationProps*, TransformationTechnology, Trigger, VariableAccess, VariationPointProxy, ViewMap, VlanConfig, Wait Point			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

▽

△

Class	***Identifiable*** (abstract)			
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:** xml.sequenceOffset=-60
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:** xml.sequenceOffset=-50
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Tags:** xml.sequenceOffset=-40
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** xml.sequenceOffset=-25
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:** xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:** xml.attribute=true

Table B.23: Identifiable

Class	***InternalBehavior*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	Common base class (abstract) for the internal behavior of both software components and basic software modules/clusters.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswInternalBehavior, SwcInternalBehavior			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
constant Memory	ParameterData Prototype	*	aggr	Describes a read only memory object containing characteristic value(s) implemented by this Internal Behavior. The shortName of ParameterDataPrototype has to be equal to the ”C’ identifier of the described constant. The characteristic value(s) might be shared between SwComponentPrototypes of the same SwComponent Type. 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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the ConstanSpecificationMapping to be applied for the particular InternalBehavior **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the particular InternalBehavior **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dataTypeMapping
exclusiveArea	ExclusiveArea	*	aggr	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 BswModule Entities. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
exclusiveArea NestingOrder	ExclusiveAreaNesting Order	*	aggr	This represents the set of ExclusiveAreaNestingOrder owned by the InternalBehavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
staticMemory	VariableDataPrototype	*	aggr	Describes a read and writeable static memory object representing measurerment 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. ▽

▽

△

Class	***InternalBehavior*** (abstract)			
				△ The shortName of the VariableDataPrototype has to be equal with the ”C’ identifier of the described variable. The aggregation of staticMemory is subject to variability with the purpose to support variability in the software component’s implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.24: InternalBehavior

Primitive	**Numerical**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This primitive specifies a numerical value. It can be denoted in different formats such as Decimal, Octal, Hexadecimal, Float. See the xsd pattern for details. The value can be expressed in octal, hexadecimal, binary representation. Negative numbers can only be expressed in decimal or float notation. **Tags:** xml.xsd.customType=NUMERICAL-VALUE xml.xsd.pattern=(0[xX][0-9a-fA-F]+)|(0[0-7]+)|(0[bB][0-1]+)|(([+\-]?[1-9][0-9]+(\.[0-9]+)?|[+\-]?[0-9](\.[0-9]+)?)([eE]([+\-]?)[0-9]+)?)|\.0|INF|-INF|NaN xml.xsd.type=string

Table B.25: Numerical

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

Table B.26: PPortPrototype

Class	***PortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/server communication.

▽

△

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

Table B.27: PortPrototype

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

Table B.28: PortPrototypeBlueprint

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

Table B.29: 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*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AsynchronousServerCallReturnsEvent, BackgroundEvent, DataReceiveErrorEvent, DataReceivedEvent, DataSendCompletedEvent, DataWriteCompletedEvent, ExternalTriggerOccurredEvent, InitEvent, InternalTriggerOccurredEvent, ModeSwitchedAckEvent, OperationInvokedEvent, SwcModeManagerErrorEvent, SwcModeSwitchEvent, TimingEvent, TransformerHardErrorEvent			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
disabledMode	ModeDeclaration	*	iref	Reference to the Modes that disable the Event. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=contextPort, contextModeDeclarationGroupPrototype, targetModeDeclaration
startOnEvent	RunnableEntity	0..1	ref	RunnableEntity starts when the corresponding RTEEvent occurs.

Table B.30: RTEEvent

Class	**ResourceConsumption**			
Package	M2::AUTOSARTemplates::CommonStructure::ResourceConsumption			
Note	Description of consumed resources by one implementation of a software.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
accessCountSet	AccessCountSet	*	aggr	Set of access count values **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
executionTime	ExecutionTime	*	aggr	Collection of the execution time descriptions for this implementation. The aggregation of executionTime is subject to variability with the purpose to support the conditional existence of runnable entities. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
heapUsage	HeapUsage	*	aggr	Collection of the heap memory allocated by this implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
memorySection	MemorySection	*	aggr	An abstract memory section required by this Implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
sectionNamePrefix	SectionNamePrefix	*	aggr	A prefix to be used for the memory section symbol in the code. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

▽

△

Class	ResourceConsumption			
stackUsage	StackUsage	*	aggr	Collection of the stack memory usage for each runnable entity of this implementation. The aggregation of StackUsage is subject to variability with the purpose to support the conditional existence of runnable entities. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.31: ResourceConsumption

Class	RootSwCompositionPrototype			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	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. 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. The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including Port Prototypes, PortInterfaces, VariableDataPrototypes, SwcInternalBehavior etc.), and their ports are interconnected using SwConnectorPrototypes.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
calibration ParameterValue Set	CalibrationParameter ValueSet	*	ref	Used CalibrationParameterValueSet for instance specific initialization of calibration parameters. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=calibrationParameterValueSet
flatMap	FlatMap	0..1	ref	The FlatMap used in the scope of this RootSw CompositionPrototype. **Stereotypes:** atpSplitable **Tags:** 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 **Stereotypes:** isOfType

Table B.32: 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*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**

▽

△

Class	**RunnableEntity**			
argument (ordered)	RunnableEntity Argument	*	aggr	This represents the formal definition of a an argument to a RunnableEntity.
asynchronous ServerCall ResultPoint	AsynchronousServer CallResultPoint	*	aggr	The server call result point admits a runnable to fetch the result of an asynchronous server call. The aggregation of AsynchronousServerCallResultPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes and the variant existence of server call result points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
canBeInvoked 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	RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataReadAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataReadAccess in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataReceive PointBy Argument	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of an argument in the function signature. The aggregation of dataReceivePointByArgument is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data receive points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataReceive PointByValue	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of the return value. The aggregation of dataReceivePointByValue is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of data receive points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**RunnableEntity**			
dataSendPoint	VariableAccess	*	aggr	RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataSendPoint is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data send points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=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. **Stereotypes:** atpSplitable; atpVariation **Tags:** 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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=externalTriggeringPoint, variation Point.shortLabel vh.latestBindingTime=preCompileTime
internal TriggeringPoint	InternalTriggeringPoint	*	aggr	The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeAccessPoint, variation Point.shortLabel vh.latestBindingTime=preCompileTime
modeSwitch Point	ModeSwitchPoint	*	aggr	The runnable has a mode switch point. The aggregation of ModeSwitchPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode switch points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
parameter Access	ParameterAccess	*	aggr	The presence of a ParameterAccess implies that a RunnableEntity needs read only access to a Parameter DataPrototype which may either be local or within a Port Prototype. The aggregation of ParameterAccess is subject to variability with the purpose to support the conditional existence of parameter ports and component local parameters as well as the variant existence of Parameter Access (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Document ID 601: AUTOSAR_TPS_TimingExtensions
— AUTOSAR CONFIDENTIAL —

▽

△

Class	**RunnableEntity**			
readLocal Variable	VariableAccess	*	aggr	The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of read LocalVariable (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
serverCallPoint	ServerCallPoint	*	aggr	The RunnableEntity has a ServerCallPoint. The aggregation of ServerCallPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes or the variant existence of server call points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbol	CIdentifier	1	attr	The symbol describing this RunnableEntity's entry point. This is considered the API of the RunnableEntity and is required during the RTE contract phase.
waitPoint	WaitPoint	*	aggr	The WaitPoint associated with the RunnableEntity.
writtenLocal Variable	VariableAccess	*	aggr	The presence of a writtenLocalVariable implies that a RunnableEntity needs write access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of writtenLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of written LocalVariable (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.33: RunnableEntity

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

Table B.34: SenderReceiverInterface

Class	**SensorActuatorSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The SensorActuatorSwComponentType introduces the possibility to link from the software representation of a sensor/actuator to its hardware description provided by the ECU Resource Template. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
sensorActuator	HwDescriptionEntity	1	ref	Reference from the Sensor Actuator Software Component Type to the description of the actual hardware.

Table B.35: SensorActuatorSwComponentType

Class	**SwComponentPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	Role of a software component within a composition.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
type	SwComponentType	1	tref	Type of the instance. **Stereotypes:** isOfType

Table B.36: SwComponentPrototype

Class	***SwComponentType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for AUTOSAR software components.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AtomicSwComponentType*, CompositionSwComponentType, ParameterSwComponentType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
consistency Needs	ConsistencyNeeds	*	aggr	This represents the collection of ConsistencyNeeds owned by the enclosing SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
port	PortPrototype	*	aggr	The PortPrototypes through which this SwComponent Type can communicate. The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
swComponent Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swComponentDocumentation, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10

▽

△

Class	***SwComponentType*** (abstract)			
unitGroup	UnitGroup	*	ref	This allows for the specification of which UnitGroups are relevant in the context of referencing SwComponentType.

Table B.37: SwComponentType

Class	**SwSystemconstantValueSet**			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
Note	This meta-class represents the ability to specify a set of system constant values. **Tags:** atp.recommendedPackage=SwSystemconstantValueSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
sw Systemconstant Value	SwSystemconstValue	*	aggr	This is one particular value of a system constant.

Table B.38: SwSystemconstantValueSet

Class	**SwcInternalBehavior**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
Note	The SwcInternalBehavior of an AtomicSwComponentType describes the relevant aspects of the software-component with respect to the RTE, i.e. the RunnableEntities and the RTEEvents they respond to.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *InternalBehavior*, *Multilanguage Referrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
arTypedPer Instance Memory	VariableDataPrototype	*	aggr	Defines an AUTOSAR typed memory-block that needs to be available for each instance of the SW-component. This is typically only useful if supportsMultipleInstantiation is set to "true" or if the component defines NVRAM access via permanent blocks. The aggregation of arTypedPerInstanceMemory is subject to variability with the purpose to support variability in the software component's implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
event	RTEEvent	*	aggr	This is a RTEEvent specified for the particular Swc InternalBehavior. The aggregation of RTEEvent is subject to variability with the purpose to support the conditional existence of RTE events. Note: the number of RTE events might vary due to the conditional existence of PortPrototypes using Data ReceivedEvents or due to different scheduling needs of algorithms. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**SwcInternalBehavior**			
exclusiveArea Policy	SwcExclusiveArea Policy	*	aggr	Options how to generate the ExclusiveArea related APIs. When no SwcExclusiveAreaPolicy is specified for an ExclusiveArea the default values apply. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveAreaPolicy vh.latestBindingTime=preCompileTime
explicitInter Runnable Variable	VariableDataPrototype	*	aggr	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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
handle TerminationAnd Restart	HandleTerminationAnd RestartEnum	1	attr	This attribute controls the behavior with respect to stopping and restarting. The corresponding AtomicSw ComponentType may either not support stop and restart, or support only stop, or support both stop and restart.
implicitInter Runnable Variable	VariableDataPrototype	*	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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
includedData TypeSet	IncludedDataTypeSet	*	aggr	The includedDataTypeSet is used by a software component for its implementation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=includedDataTypeSet
includedMode Declaration GroupSet	IncludedMode DeclarationGroupSet	*	aggr	This aggregation represents the included Mode DeclarationGroups **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=includedModeDeclarationGroupSet
instantiation DataDefProps	InstantiationDataDef Props	*	aggr	The purpose of this is that within the context of a given SwComponentType some data def properties of individual instantiations can be modified. The aggregation of InstantiationDataDefProps is subject to variability with the purpose to support the conditional existence of PortPrototypes and component local memories like "perInstanceParameter" or "arTypedPer InstanceMemory". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=instantiationDataDefProps, variation Point.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**SwcInternalBehavior**			
perInstance Memory	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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
perInstance Parameter	ParameterData Prototype	*	aggr	Defines parameter(s) or characteristic value(s) that needs to be available for each instance of the software-component. This is typically only useful if supportsMultipleInstantiation is set to "true". The aggregation of perInstanceParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** 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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=portAPIOption, variationPoint.short Label vh.latestBindingTime=preCompileTime
runnable	RunnableEntity	*	aggr	This is a RunnableEntity specified for the particular Swc InternalBehavior. The aggregation of RunnableEntity is subject to variability with the purpose to support the conditional existence of RunnableEntities. Note: the number of RunnableEntities might vary due to the conditional existence of Port Prototypes using DataReceivedEvents or due to different scheduling needs of algorithms. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=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 ▽

▽

△

Class	SwcInternalBehavior			
				△ for Obd related Service Needs) tools. Therefore the aggregation is «atpSplitable». **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
shared Parameter	ParameterData Prototype	*	aggr	Defines parameter(s) or characteristic value(s) shared between SwComponentPrototypes of the same Sw ComponentType The aggregation of sharedParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
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. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName

Table B.39: SwcInternalBehavior

Class	System			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The top level element of the System Description. The System description defines five major elements: Topology, Software, Communication, Mapping and Mapping Constraints. The System element directly aggregates the elements describing the Software, Mapping and Mapping Constraints; it contains a reference to an ASAM FIBEX description specifying Communication and Topology. **Tags:** atp.recommendedPackage=Systems			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
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	Reference to ASAM FIBEX elements specifying Communication and Topology. All Fibex Elements used within a System Description shall be referenced from the System Element. ▽

▽

△

Class	**System**			
				△ atpVariation: In order to describe a product-line, all Fibex Elements can be optional. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
j1939Shared AddressCluster	J1939SharedAddress Cluster	*	aggr	Collection of J1939Clusters that share a common address space for the routing of messages. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects (mapping of SW components to ECUs, mapping of data elements to signals, and mapping constraints). In order to support OEM / Tier 1 interaction and shared development for one common System this aggregation is atpSplitable and atpVariation. The content of 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. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
pncVector Length	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.
rootSoftware Composition	RootSwComposition Prototype	0..1	aggr	Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case. atpVariation: The RootSwCompositionPrototype can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
system Documentation	Chapter	*	aggr	Possibility to provide additional documentation while defining the System. The System documentation can be composed of several chapters. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=-10
systemVersion	RevisionLabelString	1	attr	Version number of the System Description.

Table B.40: System

Class	***TDEventBsw*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventBsw			
Note	This is used to describe timing events related to BSW modules.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	TDEventBswModeDeclaration, TDEventBswModule			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
bswModule Description	BswModuleDescription	1	ref	The scope of this timing event.

Table B.41: TDEventBsw

Class	**TimingEvent**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	TimingEvent references the RunnableEntity that need to be started in response to the TimingEvent			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Attribute	**Type**	**Mul.**	**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 B.42: TimingEvent

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

Table B.43: Traceable

Class	VariableAccess			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements			
Note	The presence of a VariableAccess implies that a RunnableEntity needs access to a VariableData Prototype. The kind of access is specified by the role in which the class is used.			
Base	*ARObject*, *AbstractAccessPoint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
accessed Variable	AutosarVariableRef	1	aggr	This denotes the accessed variable.
scope	VariableAccessScope Enum	0..1	attr	This attribute allows for constraining the scope of the corresponding communication. For example, it possible to express whether the communication is intended to cross the boundary of an ECU or whether it is intended not to cross the boundary of a single partition.

Table B.44: VariableAccess

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

Table B.45: VariableDataPrototype

C Splitable Elements in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpSplitable≫ in the scope of this document.

Each entry in Table C.1 consists of the identification of the specific model element itself and the applicable value of the tagged value `atp.Splitkey`.

For more information about the concept of splitable model elements and how these shall be treated please refer to [5].

Name of splitable element	Splitkey
TimingExtension.timingCondition	shortName, variationPoint.shortLabel
TimingExtension.timingDescription	shortName, variationPoint.shortLabel
TimingExtension.timingGuarantee	shortName, variationPoint.shortLabel
TimingExtension.timingRequirement	shortName, variationPoint.shortLabel
TimingExtension.timingResource	shortName, variationPoint.shortLabel
TimingExtensionResource.timingArgument	shortName, variationPoint.shortLabel
TimingExtensionResource.timingMode	shortName, variationPoint.shortLabel

TimingExtensionResource.timingVariable	shortName, variationPoint.shortLabel

Table C.1: Usage of splitable elements

D Variation Points in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpVariation≫ in the scope of this document.

Each entry in Table D.1 consists of the identification of the model element itself and the applicable value of the tagged value vh.latestBindingTime.

For more information about the concept of variation points and how model elements that contain variation points shall be treated please refer to [5].

Variation Point	Latest Binding Time
TimingExtension.timingCondition	postBuild
TimingExtension.timingDescription	postBuild
TimingExtension.timingGuarantee	postBuild
TimingExtension.timingRequirement	postBuild
TimingExtensionResource.timingArgument	postBuild
TimingExtensionResource.timingMode	postBuild
TimingExtensionResource.timingVariable	postBuild

Table D.1: Usage of variation points