

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

Document Version	2.1.1
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History			
Date	Version	Changed by	Description
31.03.2014	2.1.1	AUTOSAR Release Management	<ul style="list-style-type: none"> Revised the entire contents of chapter "Application Notes" Applied editorial changes to section "Repetitive Execution Order Constraint"

<p>16.09.2013</p>	<p>2.1.0</p>	<p>AUTOSAR Release Management</p>	<ul style="list-style-type: none"> • Clarified the semantics of jitter and removed ambiguities in the description of the Periodic Event Triggering Constraint. • Added AUTOSAR constraints in order to ensure specification of consistent 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 <code>ÄRTEÖ</code> <code>ÄIMEX</code> in order to be consistent with the AUTOSAR standard definitions. • Clarified the use of event types in the various timing views defined in the specification.
-------------------	--------------	---	--

<p>06.02.2013</p>	<p>2.0.0</p>	<p>AUTOSAR Release Management</p>	<ul style="list-style-type: none"> • Applied editorial changes in order to improve readability and comprehensibility of the contents of the document • Added VFB event type <code>TDEventTrigger</code> and extended <code>TDEventSwcInternalBehaviourTypeEnum</code> to indicate variable access of runnable entities • Extended the capability of <code>SynchronizationTimingConstraint</code> to reference timing description events • Revised and extended the capabilities of <code>ExecutionOrderConstraint</code> to specify hierarchical and repetitive execution order constraints • Added the capability to specify blueprints of <code>VfbTimings</code> • Added capabilities to reference timing description events in existing timing models and to support reuse of timing models, as well as AUTOSAR methodology
<p>22.09.2011</p>	<p>1.2.0</p>	<p>AUTOSAR Release Management</p>	<ul style="list-style-type: none"> • Added new timing constraint types <code>AgeConstraint</code> and <code>ExecutionTimeConstraint</code> • Added occurrence expression language for <code>TimingDescriptionEvents</code> • Improved <code>TDEventModeDeclaration</code>, <code>BurstPatternEventTriggering</code> and <code>SwcTiming</code>

03.11.2010	1.1.0	AUTOSAR Release Management	<ul style="list-style-type: none"> • Dropped InstanceRefs and replaced with ComponentIn-CompositionInstanceRef • Restricted the semantics of ExecutionOrderConstraint and OffsetConstraint • Parameterize the observable event 'FlexRayClusterCycleStart' by defining the cycle repetition
30.11.2009	1.0.0	AUTOSAR Release Management	Initial Release

Disclaimer

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

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

This specification may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only.

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

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

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

Table of Contents

1	Introduction	10
1.1	Overview	10
1.2	Abbreviations	10
1.3	Glossary of terms	10
1.4	Template implications	13
1.5	Scope	13
1.6	Document conventions	14
1.7	Requirements Traceability	14
2	Timing Extensions Overview	18
3	Timing Views	21
3.1	Timing in Different Phases of the AUTOSAR Methodology	21
3.2	VfbTiming	24
3.3	SwcTiming	26
3.4	SystemTiming	27
3.5	BswModuleTiming	29
3.6	EcuTiming	30
4	Timing Extensions Fundamentals	32
4.1	Formal Specification of Timing Behavior	32
4.2	Timing Extensions and Blueprints	33
4.3	Traceability of Constraints	34
5	Timing Description Events	35
5.1	Timing Events Related to the VFB	36
5.2	Timing Events Related to SwcInternalBehavior	43
5.3	Timing Events Related to Bus Communication	46
5.4	Timing Events Related to the BSW	51
5.5	Complex Timing Event	55
5.6	Occurrence Expression Language for Timing Events	56
5.6.1	Specifying an Occurrence Expression	57
5.6.2	Occurrence Expression Language Syntax	63
5.6.3	Interpreting an Occurrence Expression	64
6	Timing Description Event Chains	66
7	Timing Constraints	68
7.1	EventTriggeringConstraint	69
7.1.1	PeriodicEventTriggering	70
7.1.1.1	Examples	73
7.1.2	SporadicEventTriggering	75
7.1.3	ConcretePatternEventTriggering	77
7.1.4	BurstPatternEventTriggering	79
7.1.5	ArbitraryEventTriggering	83

7.2	LatencyTimingConstraint	85
7.3	AgeConstraint	88
7.4	SynchronizationTimingConstraint	90
7.4.1	SynchronizationTimingConstraint on Event Chains	93
7.4.2	SynchronizationTimingConstraint on Events	95
7.5	OffsetTimingConstraint	97
7.6	ExecutionOrderConstraint	98
7.6.1	Ordinary Execution Order Constraint	103
7.6.2	Hierarchical Execution Order Constraint	103
7.6.3	Repetitive Execution Order Constraint	105
7.7	ExecutionTimeConstraint	107
8	Application Notes	109
8.1	Component integration	109
8.1.1	VFB view	113
8.1.2	ECU view	115
8.2	Engine control	117
8.2.1	Overview	117
8.2.2	Timing Requirements	118
8.2.3	Formal description of timing constraints in VFB View	119
8.2.3.1	Requirement 1	119
8.2.3.2	Requirement 2	124
8.2.3.3	Requirement 3	127
8.2.4	Formal description of timing constraints in ECU View	130
8.2.4.1	Requirement 4	131
8.2.5	Formal description of timing constraints in SW-C View	134
8.2.5.1	Requirement 5	134
8.3	Describing and Constraining Sensor and Actuator Timing	135
8.3.1	External Event of a Sensor accessed via S/R	136
8.3.2	External Event of an Actuator accessed via S/R	136
8.3.3	External Event of a Sensor accessed via C/S	137
8.3.4	External Event of an Actuator accessed via C/S	137
8.3.5	Considering hardware I/O latency of EventChains at VFB-level	137
8.3.6	Constraining Input or Output Latency	138
A	History of Constraints and Specification Items	139
A.1	Constraint History of this Document related to AUTOSAR R4.0.1	139
A.1.1	Changed Constraints in R4.0.1	139
A.1.2	Added Constraints in R4.0.1	139
A.1.3	Deleted Constraints in R4.0.1	139
A.2	Constraint History of this Document related to AUTOSAR R4.0.2	139
A.2.1	Changed Constraints in R4.0.2	139
A.2.2	Added Constraints in R4.0.2	139
A.2.3	Deleted Constraints in R4.0.2	139
A.3	Constraint History of this Document related to AUTOSAR R4.0.3	140
A.3.1	Changed Constraints in R4.0.3	140
A.3.2	Added Constraints in R4.0.3	140

A.3.3 Deleted Constraints in R 4.0.3	140
A.4 Constraint History of this Document related to AUTOSAR R4.1.1	140
A.4.1 Changed Constraints in R4.1.1	140
A.4.2 Added Constraints in R4.1.1	140
A.4.3 Deleted Constraints in R4.1.1	141
A.5 Constraint History of this Document related to AUTOSAR R4.1.2	141
A.5.1 Changed Constraints in R4.1.2	141
A.5.2 Added Constraints in R4.1.2	142
A.5.3 Deleted Constraints in R4.1.2	142
A.6 Added Specification Items in R4.0.3	142
A.7 Added Specification Items in R4.1.1	143
A.8 Added Specification Items in R4.1.2	143
B Mentioned Class Tables	144

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

<i>Abbreviation</i>	<i>Meaning</i>
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

<i>Term</i>	<i>Definition</i>
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 <code>RunnableEntity</code> 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 <code>RunnableEntity</code> 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 `ExecutableEntities` 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 (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]	The AUTOSAR templates shall provide the means to describe the timing properties of a system's dynamics, which are determined by the consumption of computation, communication, and other hardware resources.	[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]

Requirement	Description	Satisfied by
[RS_TIMEX_00002]	The AUTOSAR templates shall provide the means to describe timing constraints, such as software and hardware latency, input/output latency, synchronization, and runnable execution order constraints with clearly defined semantics. Also, the scope and the boundaries of timing constraints shall be explicitly described.	[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]
[RS_TIMEX_00003]	The usage of timing constraints in the AUTOSAR templates shall be optional.	[TPS_TIMEX_00009]
[RS_TIMEX_00004]	The AUTOSAR templates shall provide the means to describe timing specific event chains. An event chain is used as the subject to attach a timing constraint.	[TPS_TIMEX_00002]
[RS_TIMEX_00005]	It shall be possible to organize event chains in hierarchies. That is, event chains can be built up from arbitrary event sub-chains. Leafs of the hierarchy are atomic event chains. Atomic event chains are defined in the sense that stimulus and response are clearly defined by the interaction semantics.	[TPS_TIMEX_00002]
[RS_TIMEX_00006]	The AUTOSAR templates shall provide the means to describe the triggering behavior (e.g. periodic, sporadic, and arbitrary) of event chains.	[TPS_TIMEX_00003] [TPS_TIMEX_00010] [TPS_TIMEX_00011] [TPS_TIMEX_00012] [TPS_TIMEX_00013] [TPS_TIMEX_00014]
[RS_TIMEX_00007]	The AUTOSAR templates shall provide the means to describe timing constraints for the synchronization of multiple event chains with possibly independent stimulus and response events.	[TPS_TIMEX_00006]
[RS_TIMEX_00008]	The AUTOSAR templates shall provide the means to describe multiple asynchronous clocks/time bases and their interrelation.	[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]	It shall be possible to annotate connections among SWCs on VFB, to indicate that a sender-receiver communication needs to be buffered.	[TPS_TIMEX_00002] [TPS_TIMEX_00005]
[RS_TIMEX_00010]	The AUTOSAR templates shall provide the means to describe the validity of timing properties and constraints, e.g. for a certain hardware or software configuration.	[TPS_TIMEX_00037]

Requirement	Description	Satisfied by
[RS_TIMEX_00012]	The AUTOSAR templates shall provide the means to describe the time relation between a physical sensor acquisition (or a physical actuator change) and the availability (or provision) of the corresponding data on the port of a sensor (or actuator) software component on VFB level.	[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]
[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]

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

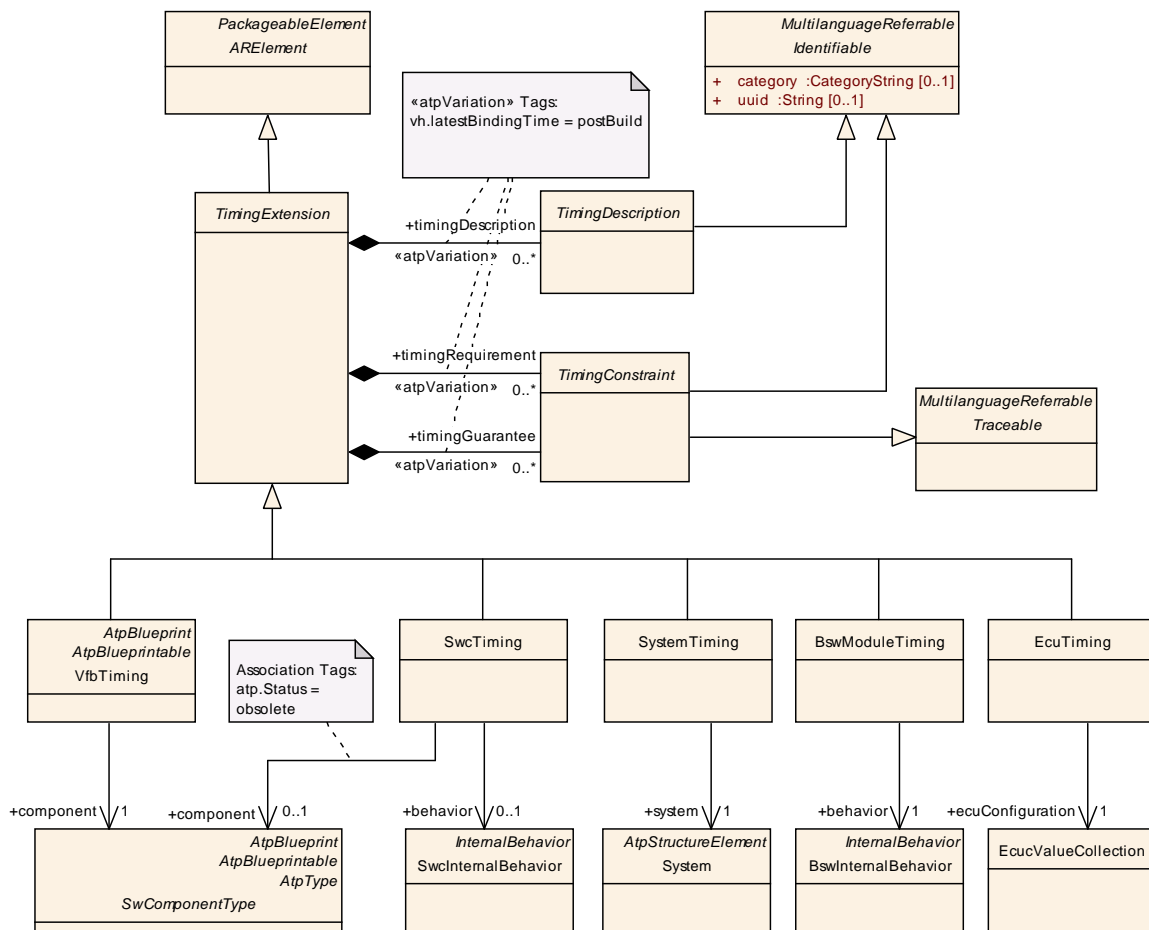


Figure 2.1: Overview of the Timing Extensions including Timing Descriptions, Timing Constraints, and Timing Views.

Class	TimingExtension (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing			
Note	<p>The abstract parent class of the different template specific timing extensions.</p> <p>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.</p>			
Base	ARElement,ARObject,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	Datatype	Mul.	Kind	Note
timingDescription	TimingDescription	*	aggr	<p>The timing descriptions that belong to a specific timing specification.</p> <p>In order to support different timing description variants within a timing specification, the aggregation is marked with the stereotype "atpVariation".</p> <p>Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild</p>
timingGuarantee	TimingConstraint	*	aggr	<p>The timing constraints that belong to a specific timing specification in the role of a timing guarantee.</p> <p>In order to support different timing constraint variants within a timing specification, the aggregation is marked with the stereotype "atpVariation".</p> <p>Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild</p>
timingRequirement	TimingConstraint	*	aggr	<p>The timing constraints that belong to a specific timing specification in the role of a timing requirement.</p> <p>In order to support different timing constraint variants within a timing specification, the aggregation is marked with the stereotype "atpVariation".</p> <p>Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild</p>

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			
Attribute	Datatype	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			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

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 `SwComponentTypes` at VFB level.

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

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

BswModuleTiming This view deals with timing information related to the `BswInternalBehavior` of a single `BswModuleDescription`.

EcuTiming This view deals with timing information related to the `EcucValueCollection`, particularly with the `EcucModuleConfigurationValues`.

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.

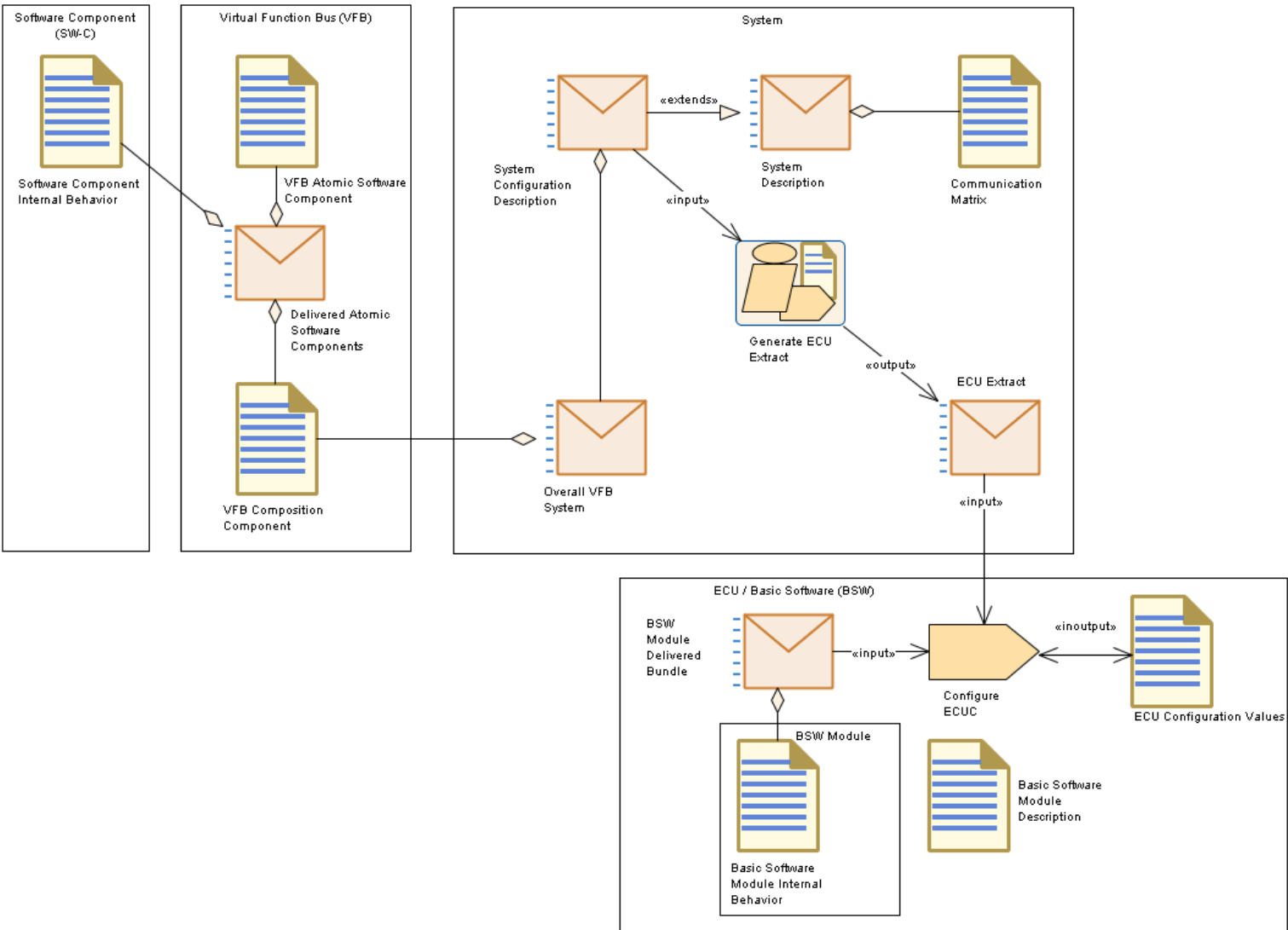


Figure 3.1: Overview of the AUTOSAR methodology and timing specification

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. Typically, 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`.

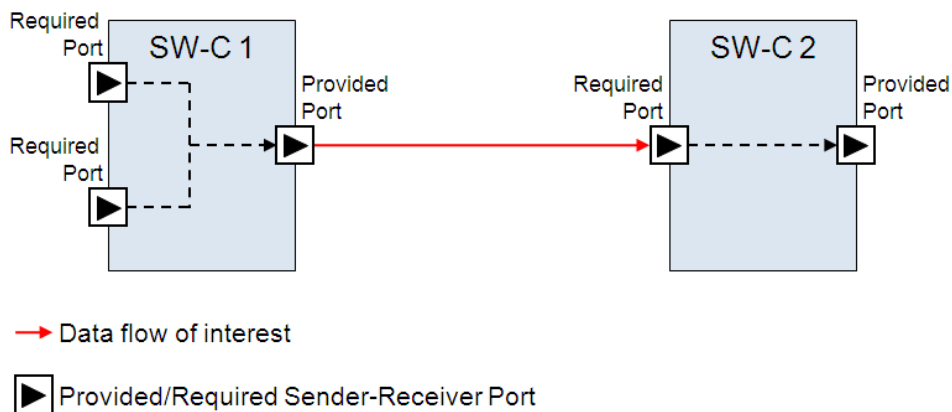


Figure 3.2: Example: Data flow in the scope of the VfbTiming view

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`.

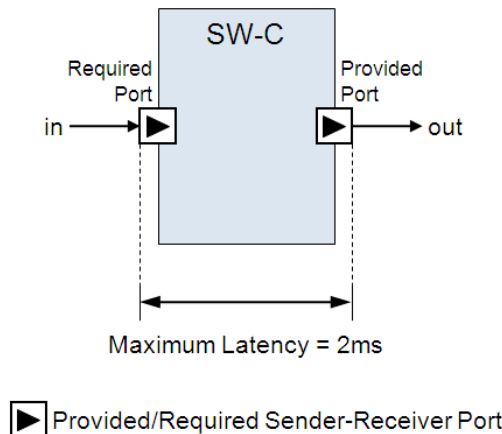


Figure 3.3: Example: Latency requirement

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 Prototypes`, 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	<p>A model element used to define timing descriptions and constraints at VFB level.</p> <p>TimingDescriptions aggregated by VfbTiming are restricted to event chains referring to events which are derived from the class <code>TDEventVfb</code>.</p> <p>Tags: atp.recommendedPackage=TimingExtensions</p>			
Base	<code>ARElement, ARObjct, AtpBlueprint, AtpBlueprintable, Collectable Element, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, Timing Extension</code>			
Attribute	Datatype	Mul.	Kind	Note
component	<code>SwComponentType</code>	1	ref	This defines the scope of a <code>vfbTiming</code> . 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 `AtomicSwComponentTypes` that are represented as black boxes at VFB level. The `SwcInternalBehavior` specifies a component's behavioral decomposition into `RunnableEntities`, which are executed at runtime. Thus, in `SwcTiming` view, a timing description is attached to the `Component Internal Behavior Description` of a `SwComponentType` (see figure 3.1). It can refer to the activation, start, and termination (see section 5.2) of the execution of `RunnableEntities`.

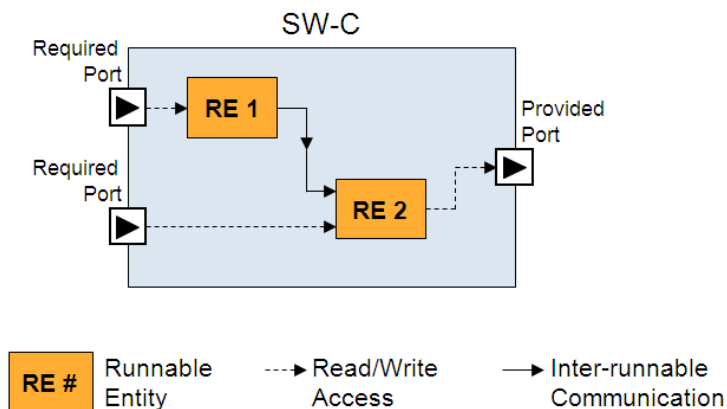


Figure 3.4: Example: Data flow in the scope of the SW-C Timing view

[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	<p>The <code>SwcTiming</code> is used to describe the timing of an atomic software component.</p> <p>TimingDescriptions aggregated by <code>SwcTiming</code> are restricted to event chains referring to events which are derived from the classes <code>TDEventVfb</code> and <code>TDEventSwcInternalBehavior</code>.</p> <p>Tags: <code>atp.recommendedPackage=TimingExtensions</code></p>			
Base	<code>ARElement</code> , <code>ARObject</code> , <code>CollectableElement</code> , <code>Identifiable</code> , <code>Multilanguage</code> , <code>Referrable</code> , <code>PackageableElement</code> , <code>Referrable</code> , TimingExtension			
Attribute	Datatype	Mul.	Kind	Note
behavior	<code>SwcInternalBehavior</code>	0..1	ref	<p>This defines the scope of a <code>SwcTiming</code>. All corresponding timing descriptions and constraints must be defined within this scope.</p> <p>Note! The reason for the cardinality of 0..1 is to ensure backward compatibility.</p>

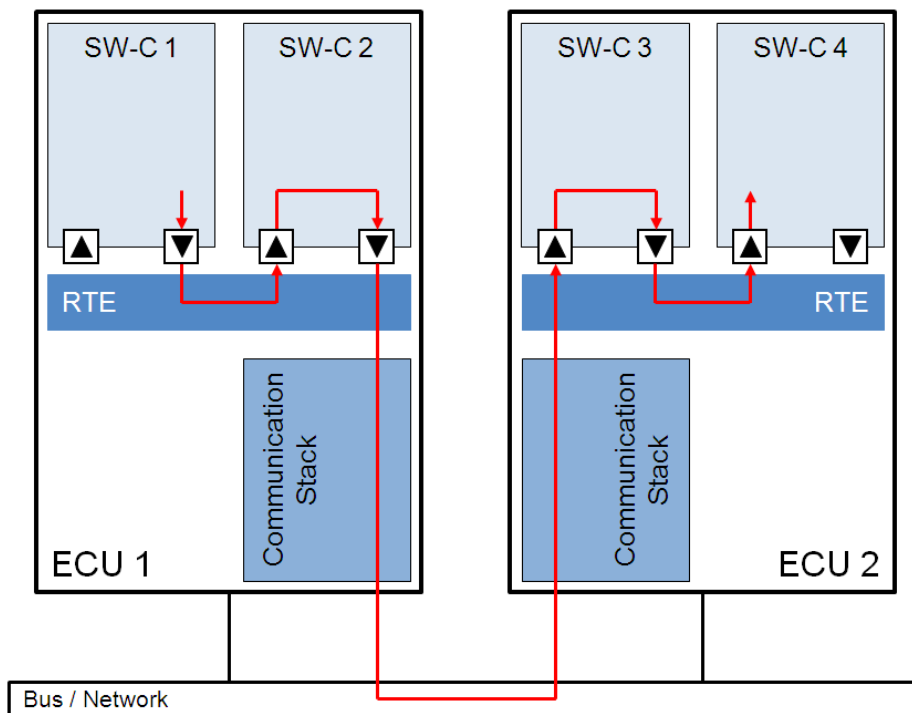
Attribute	Datatype	Mul.	Kind	Note
component	SwComponentType	0..1	ref	Deprecated. The use of this association is deprecated and one is encouraged to use the association between SwcTiming and SwcInternalBehavior. Please note, that the association between SwcTiming and SwComponentType is going to be removed in the future. Tags: atp.Status=obsolete

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 `Deploy Software Component` 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 `SwComponentTypes` and their corresponding `SwComponentInternalBehavior`, 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.



→ Data flow of interest

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	<p>A model element used to refine timing descriptions and constraints (from a <code>VfbTiming</code>) at System level, utilizing information about topology, software deployment, and signal mapping described in the System Template.</p> <p>TimingDescriptions aggregated by <code>SystemTiming</code> are restricted to events which are derived from the class <code>TDEventVfb</code>, <code>TDEventSwcInternalBehavior</code> and <code>TDEventCom</code>.</p> <p>Tags: <code>atp.recommendedPackage=TimingExtensions</code></p>			
Base	<code>ARElement</code> , <code>ARObject</code> , <code>CollectableElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>PackageableElement</code> , <code>Referrable</code> , TimingExtension			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	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

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 `BswModuleEntities`. 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 that `BswModuleEntities`. The timing description for each module can be attached to the BSW Module Description.

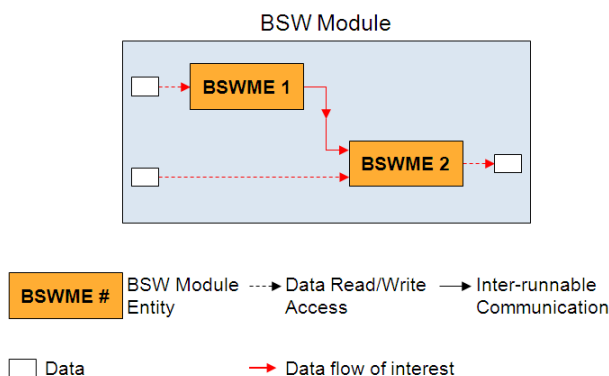


Figure 3.6: Example: Data flow in scope of BSW Module Timing view

[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	<p>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.</p> <p>A constraint defined at this level holds true for all Implementations of that BswInternalBehavior.</p> <p>TimingDescriptions aggregated by BswModuleTiming are restricted to event chains referring to events which are derived from the class TDEventBswInternalBehavior.</p> <p>Tags: atp.recommendedPackage=TimingExtensions</p>			
Base	ARElement,ARObject,CollectableElement,Identifiable,Multilanguage Referrable,PackageableElement,Referrable, TimingExtension			
Attribute	Datatype	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

3.6 EcuTiming

A result of the ECU configuration phase is the complete `EcuConfigurationValues` 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 via **BSW Module Descriptions**, providing for instance information about the interfaces that the modules offer or require.

... check this list and validate against current state of AUTOSAR methodology.

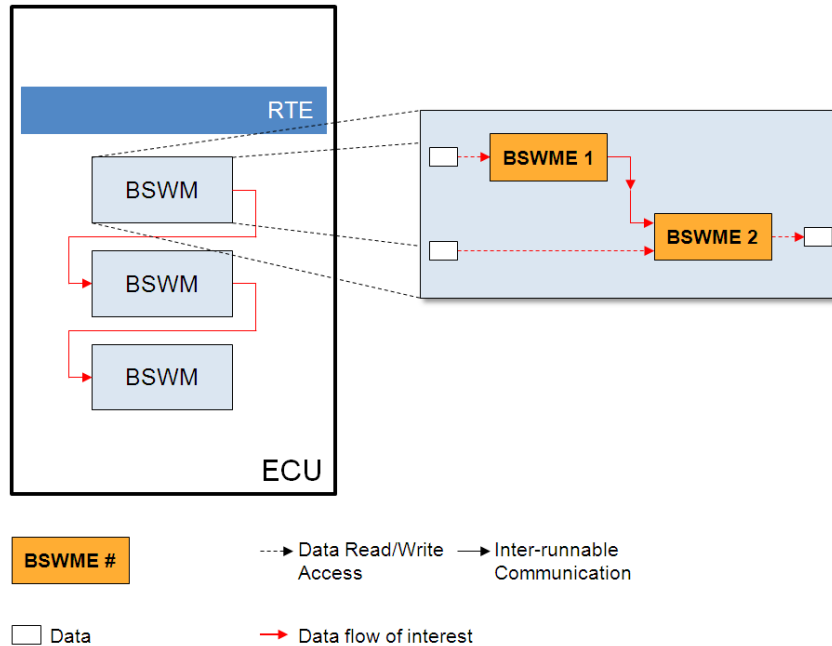


Figure 3.7: An example of data flow, whose timing behavior is in scope of ECU view

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 `EcuValueCollection`.

[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	<p>A model element used to define timing descriptions and constraints within the scope of one ECU configuration.</p> <p>TimingDescriptions aggregated by <code>EcuTiming</code> are allowed to use all events derived from the class <code>TimingDescriptionEvent</code>.</p> <p>Tags: atp.recommendedPackage=TimingExtensions</p>			
Base	ARElement,ARObject,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable, TimingExtension			
Attribute	Datatype	Mul.	Kind	Note
ecuConfiguration	EcuValueCollection	1	ref	This defines the scope of an <code>EcuTiming</code> . All corresponding timing descriptions and constraints must be defined within this scope.

Table 3.5: 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 tim-

ing 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].

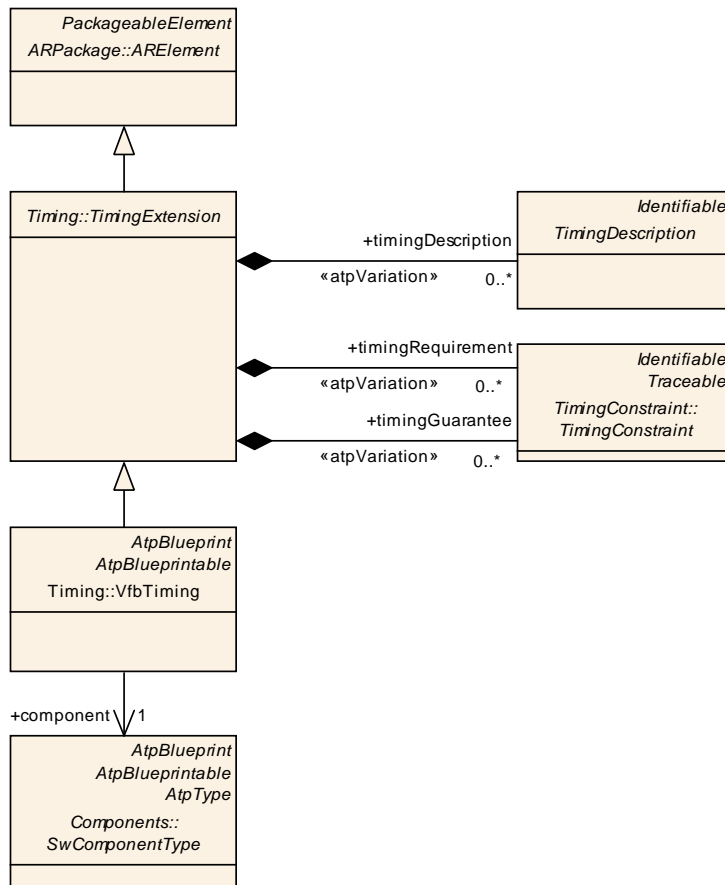


Figure 4.1: VFB Timing Blueprint

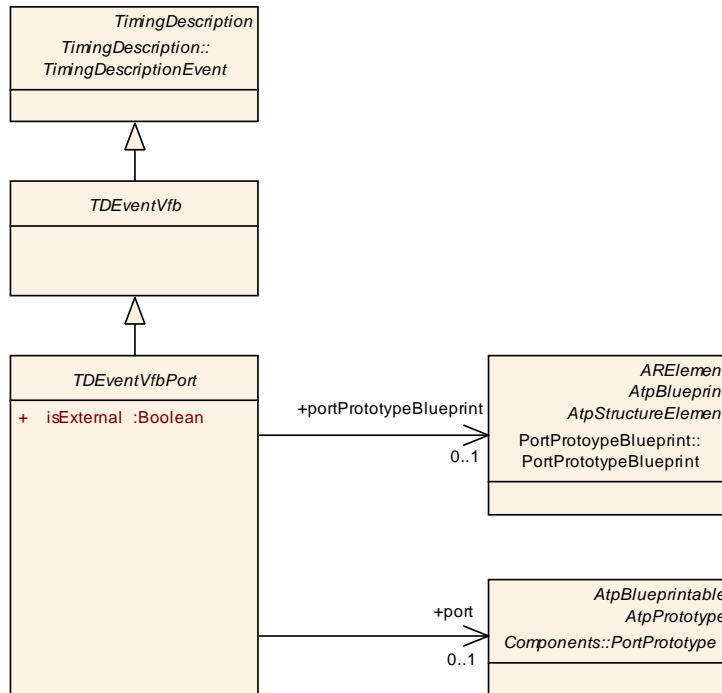


Figure 4.2: TDEventVfb Blueprint

[constr_4508] TDEventVfb shall reference PortPrototypeBlueprint only in Blueprints [An event type TDEventVfb only shall reference PortPrototypeBlueprint 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.

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.

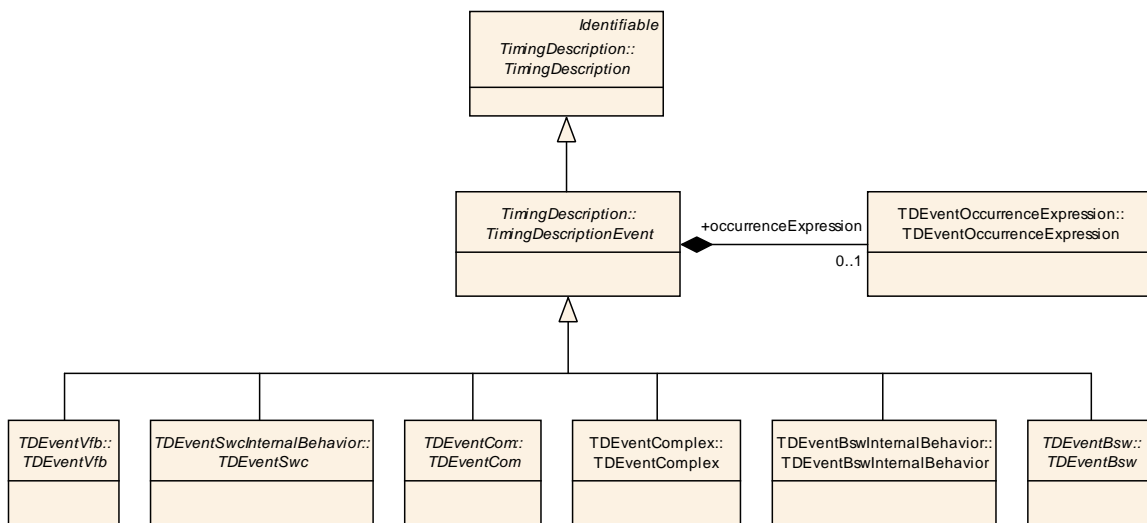


Figure 5.1: Overview of the different types of timing events

Class	TimingDescriptionEvent (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription			
Note	<p>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.</p> <p>In order to avoid confusion with existing event descriptions in the AUTOSAR templates the timing specific event types use the prefix TD.</p>			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, TimingDescription			
Attribute	Datatype	Mul.	Kind	Note
occurrence Expression	TDEventOccurrenceExpression	0..1	aggr	The occurrence expression for this event.

Table 5.1: TimingDescriptionEvent

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

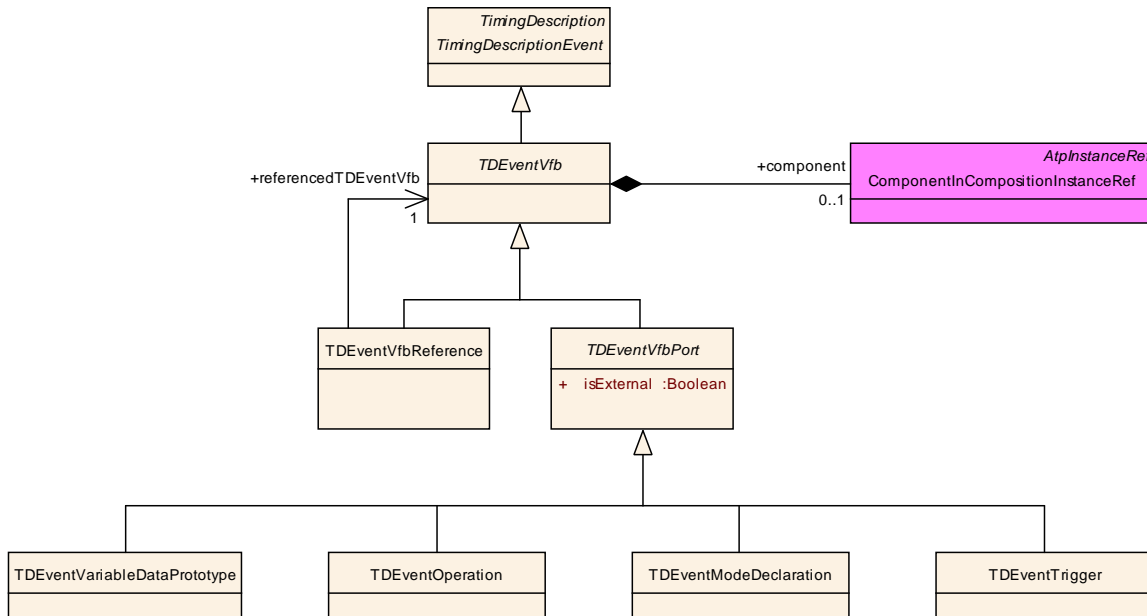


Figure 5.2: VFB events

Class	TDEventVfb (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::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			
Attribute	Datatype	Mul.	Kind	Note
component	SwComponentPrototype	0..1	iref	The context for the scope of this timing event.

Table 5.2: 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::TimingDescriptionEvents::TDEventVfb			
Note	This is the abstract parent class to describe specific timing event types at Virtual Function Bus (VFB) level.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, TDEventVfb , TimingDescription , TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
isExternal	Boolean	1	attr	This attribute is used to refer to external events that are related to hardware I/O, like physical sensors and actuators, at Virtual Function Bus (VFB) level.
port	PortPrototype	0..1	ref	The port scope of the timing event.
portPrototypeBlueprint	PortPrototypeBlueprint	0..1	ref	The PortPrototypeBlueprint is the scope of the timing event.

Table 5.3: 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 `TDEventVfb` the attribute is set to `TRUE`, that 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`, 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::TimingDescriptionEvents::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 , TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
referencedTDEventVfb	TDEventVfb	1	ref	The referenced timing description event.

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

Table 5.4: 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 variable data prototypes, observable at required and provided sender/receiver ports.]([RS_TIMEX_00001](#))

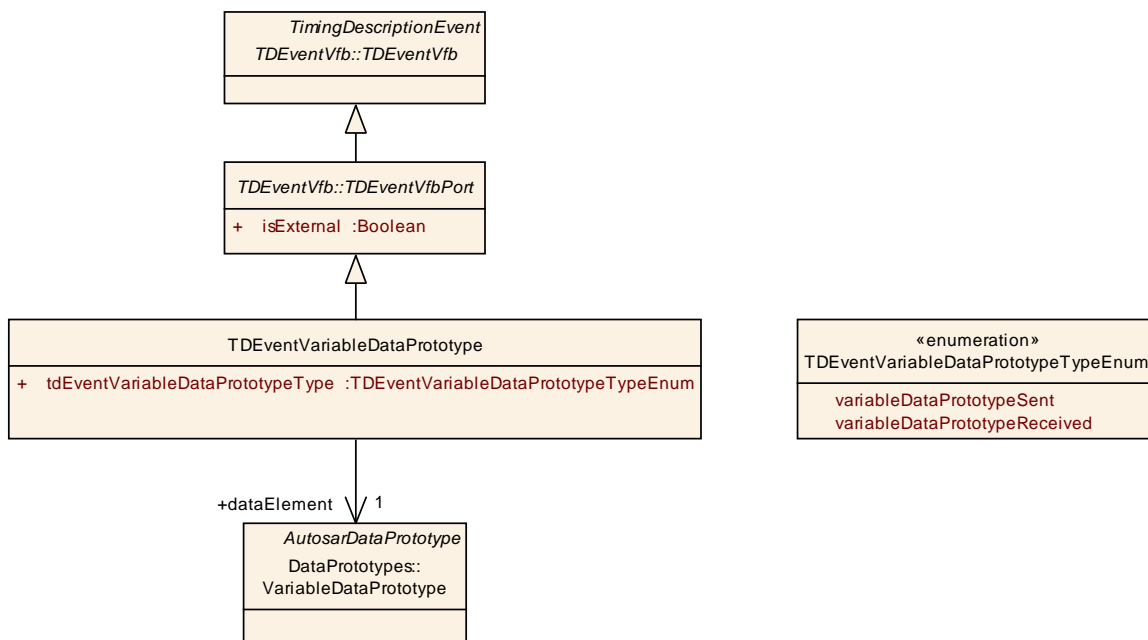


Figure 5.3: Variable Data Prototype

Class	TDEventVariableDataPrototype			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::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 , TimingDescription , TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
dataElement	VariableDataPrototype	1	ref	The referenced VariableDataPrototype
tdEventVariableDataPrototypeType	TDEventVariableDataPrototypeTypeEnum	1	attr	The specific type of this timing event.

Table 5.5: TDEventVariableDataPrototype

Enumeration	TDEventVariableDataPrototypeTypeEnum
--------------------	---

Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventVfb::VariableDataPrototype
Note	This is used to describe the specific event type of a TDEventVariableDataPrototype
Literal	Description
variableData Prototype Received	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.
variableData Prototype Sent	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.

Table 5.6: TDEventVariableDataPrototypeTypeEnum

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

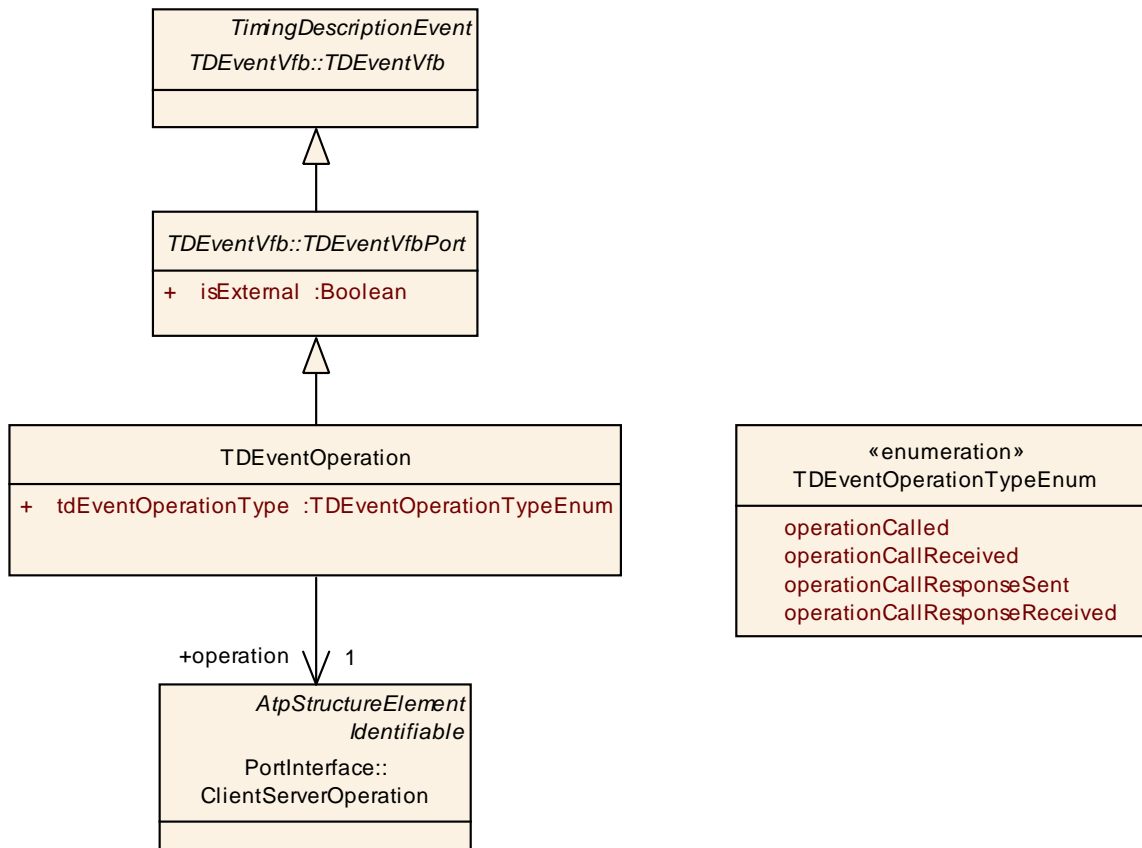


Figure 5.4: Operation

Class	TDEventOperation			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::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 , TimingDescription , TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
operation	ClientServerOperation	1	ref	The referenced operation.
tdEventOperationType	TDEventOperationTypeEnum	1	attr	The specific type of this timing event.

Table 5.7: TDEventOperation

Enumeration	TDEventOperationTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventVfb::Operation
Note	This is used to describe the specific event type of a TDEventOperation.
Literal	Description
operationCallReceived	A point in time where the call of the referenced operation is received by the server SWC.
operationCallResponseReceived	A point in time where the client SWC has received the response of the referenced operation call.
operationCallResponseSent	A point in time where the server SWC has terminated with the execution of the referenced operation, and has sent out a response.
operationCalled	A point in time where the referenced operation is called by the client SWC.

Table 5.8: TDEventOperationTypeEnum

[TPS_TIMEX_00019] TDEventModeDeclaration specifies events observable 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](#))

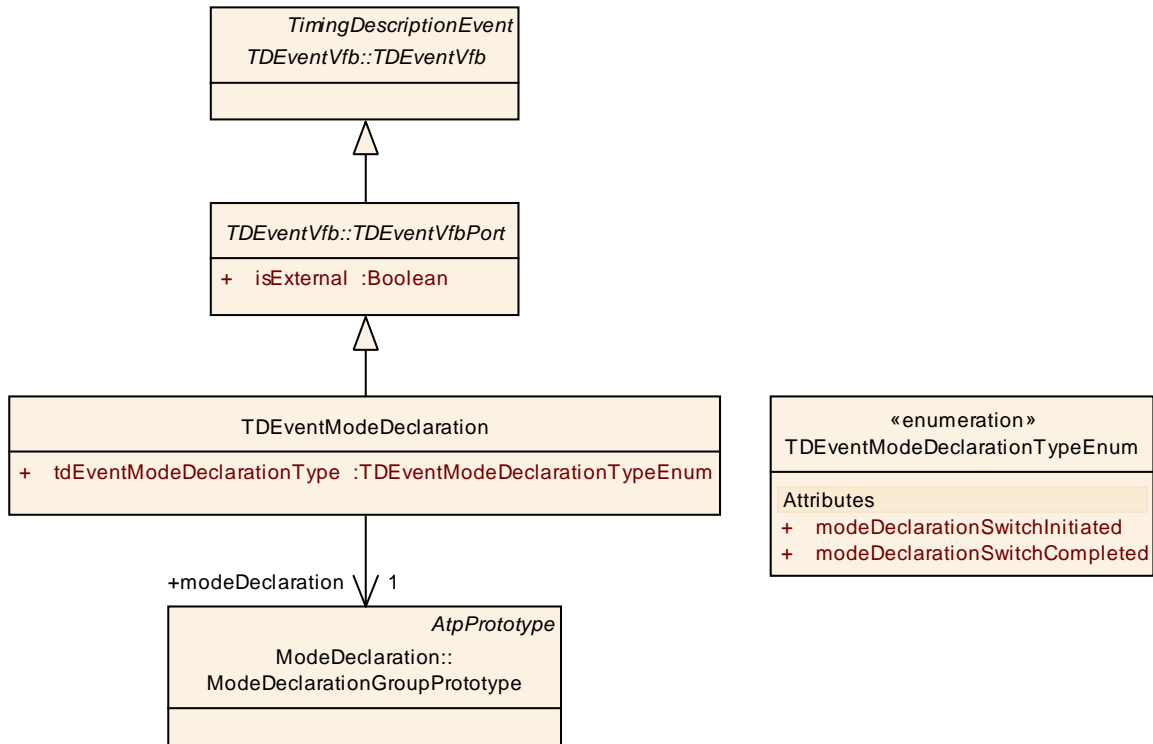


Figure 5.5: Mode Declaration

Class	TDEventModeDeclaration			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::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 , TimingDescription , TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
entryModeDeclaration	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.
exitModeDeclaration	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.
modeDeclaration	ModeDeclarationGroupPrototype	1	ref	The referenced mode declaration group prototype.
tdEventModeDeclarationType	TDEventModeDeclarationTypeEnum	1	attr	The specific type of this timing event.

Table 5.9: TDEventModeDeclaration

Enumeration	TDEventModeDeclarationTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventVfb::ModeDeclaration
Note	This is used to describe the specific event type of a TDEventModeDeclaration
Literal	Description
modeDeclarationSwitchCompleted	A point in time where the switch to the associated ModeDeclarationGroupPrototype has been completed.
modeDeclarationSwitchInitiated	A point in time where the switch to the associated ModeDeclarationGroupPrototype has been initiated.

Table 5.10: 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](#))

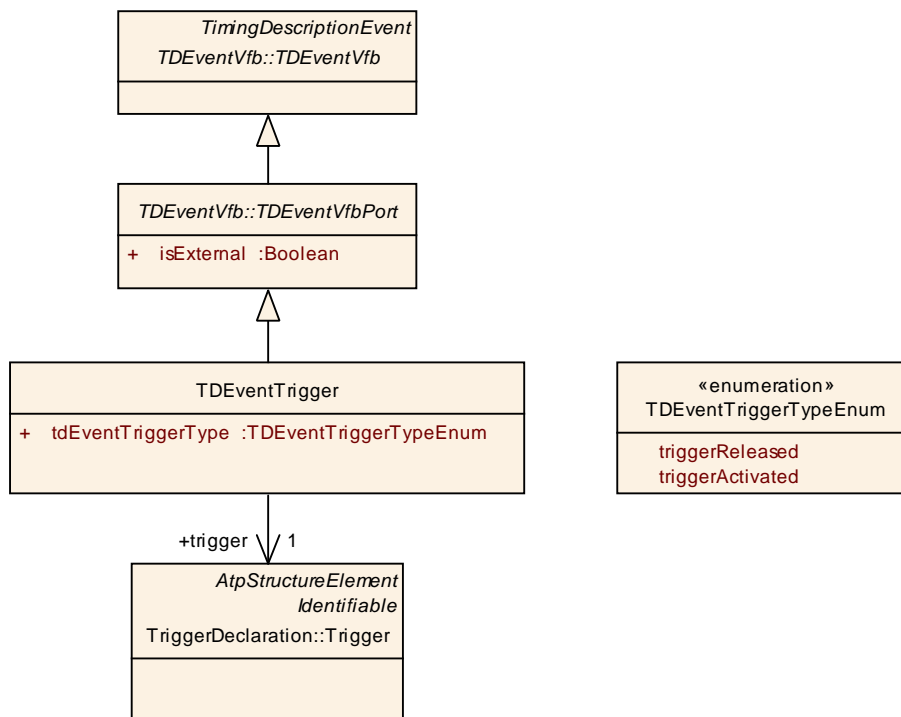


Figure 5.6: Trigger

Class	TDEventTrigger			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventVfb::Trigger			
Note	This is used to describe timing events related to triggers at VFB level.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable, TDEventVfb , TDEventVfbPort , TimingDescription , TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
tdEventTriggerType	TDEventTriggerTypeEnum	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 5.11: TDEventTrigger

Enumeration	TDEventTriggerTypeEnum			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::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.			
triggerReleased	A point in time where the referenced trigger has been successfully released by the emitting SW-C.			

Table 5.12: 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::TimingDescriptionEvents::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			
Attribute	Datatype	Mul.	Kind	Note
component	SwComponentPrototype	0..1	iref	The context for the scope of this timing event.

Table 5.13: 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](#)

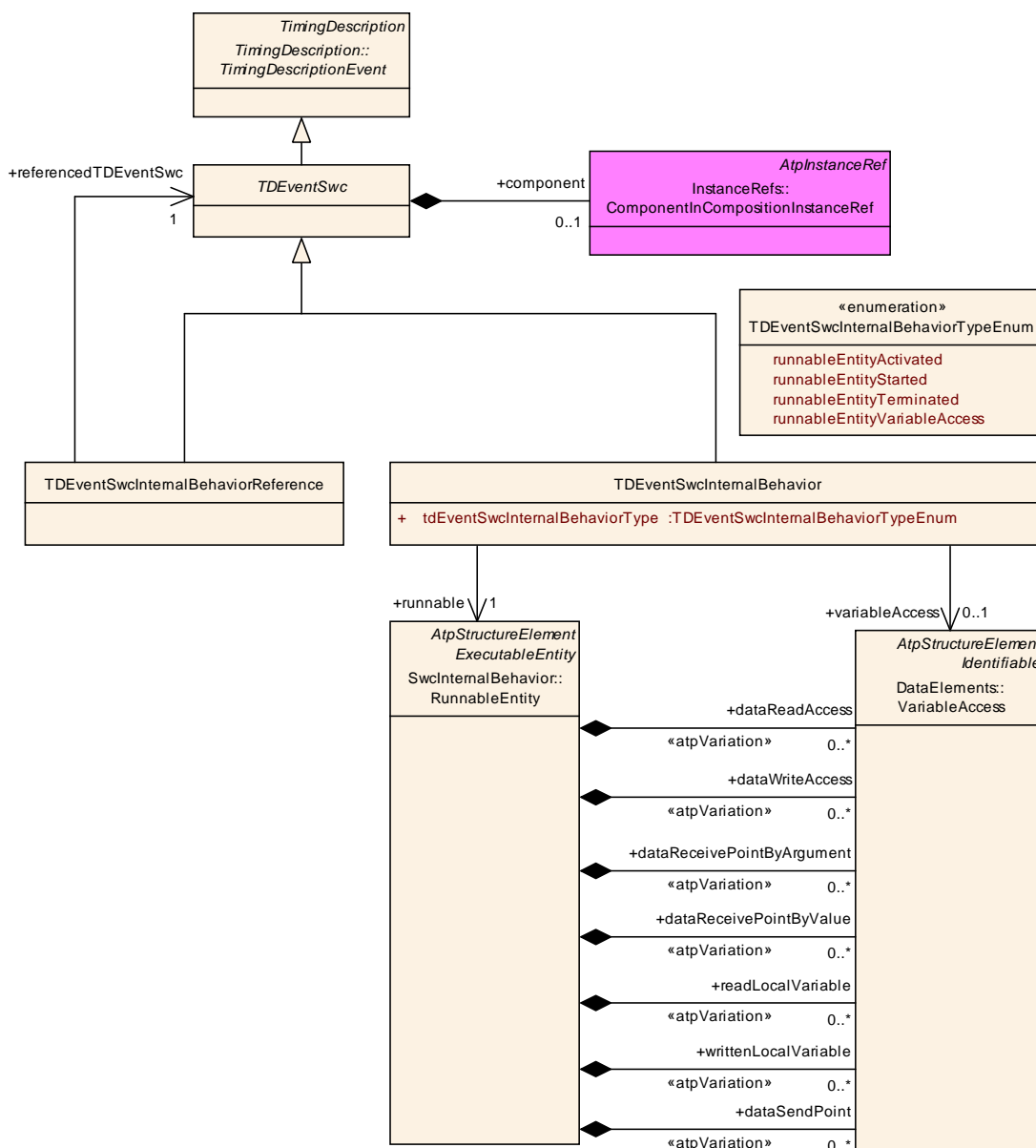


Figure 5.7: Event of type "SwcInternalBehavior"

Class	TDEventSwcInternalBehavior			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventSwcInternalBehavior			
Note	This is used to describe timing events related to the SwcInternalBehavior of an AtomicSwComponentType.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TDEventSwc,TimingDescription,TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
runnable	RunnableEntity	1	ref	The scope of this timing event.
tdEventSwcInternalBehaviorType	TDEventSwcInternalBehaviorTypeEnum	1	attr	The specific type of this timing event.
variableAccess	VariableAccess	0..1	ref	The scope of this timing event.

Table 5.14: TDEventSwcInternalBehavior

Enumeration	TDEventSwcInternalBehaviorTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventSwcInternalBehavior
Note	This is used to describe the specific event type of a TDEventSwcInternalBehavior.
Literal	Description
runnableEntityActivated	A point in time where the associated RunnableEntity has been activated, which means that it has entered the state "to be started".
runnableEntityStarted	A point in time where the associated RunnableEntity has entered the state "started" after its activation.
runnableEntityTerminated	A point in time where the associated RunnableEntity has terminated and entered the state "suspended".
runnableEntityVariableAccess	A point in time where the associated variable is accessed.

Table 5.15: 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 "runnableEntityVariableAccess". 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::TimingDescriptionEvents::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	Datatype	Mul.	Kind	Note
referenced TDEventSwc	TDEventSwc	1	ref	The referenced timing description event.

Table 5.16: 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](#)

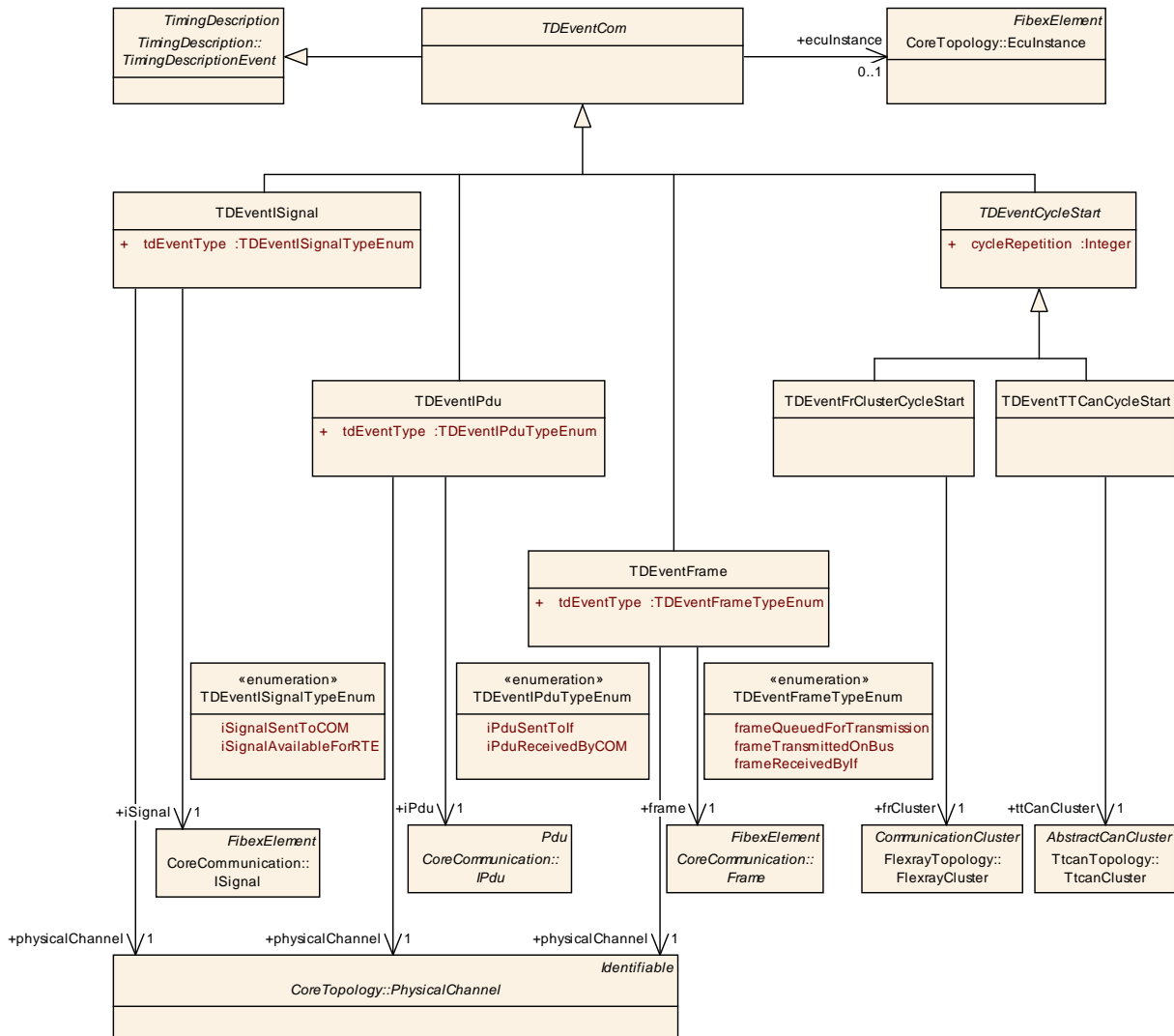


Figure 5.8: Events regarding communication

Class	TDEventCom (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::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			
Attribute	Datatype	Mul.	Kind	Note
eculInstance	EculInstance	0..1	ref	The ECU context for a particular timing event. The link is optional, because the EculInstance can not be defined for events of type TDEventCycleStart.

Table 5.17: 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::TimingDescriptionEvents::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 , TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
iSignal	ISignal	1	ref	The scope of this timing event.
physicalChannel	PhysicalChannel	1	ref	The PhysicalChannel on which the ISignal is transmitted.
tdEventType	TDEventISignalTypeEnum	1	attr	The specific type of this timing event.

Table 5.18: TDEventISignal

Enumeration	TDEventISignalTypeEnum			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventCom			
Note	This is used to describe the specific event type of a TDEventISignal.			
Literal	Description			
iSignalAvailableForRTE	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).			
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.			

Table 5.19: 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::TimingDescriptionEvents::TDEventCom			
Note	This is used to describe timing events related to the exchange of I-PDUs between the bus specific (FlexRay / CAN / LIN) Interface BSW module and COM.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable, TDEventCom , TimingDescription , TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
iPdu	IPdu	1	ref	The scope of this timing event.
physicalChannel	PhysicalChannel	1	ref	The PhysicalChannel on which the IPdu is transmitted.

Attribute	Datatype	Mul.	Kind	Note
tdEventTy pe	TDEventIPduTy peEnum	1	attr	The specific type of this timing event.

Table 5.20: TDEventIPdu

Enumeration	TDEventIPduTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::Timing DescriptionEvents::TDEventCom
Note	This is used to describe the specific event type of a TDEventIPdu.
Literal	Description
iPduReceived ByCOM	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.
iPduSentTolf	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.

Table 5.21: 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::Timing DescriptionEvents::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 , Timing Description , TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
frame	Frame	1	ref	The scope of this timing event.
physicalCh annel	PhysicalChanne l	1	ref	The PhysicalChannel on which the Frame is transmitted.
tdEventTy pe	TDEventFrame TypeEnum	1	attr	The specific type of this timing event.

Table 5.22: TDEventFrame

Enumeration	TDEventFrameTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::Timing DescriptionEvents::TDEventCom
Note	This is used to describe the specific event type of a TDEventFrame.
Literal	Description

frameQueuedForTransmission	A point in time where the frame containing the named signal / I-PDU is queued for transmission within the related Communication Driver.
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.
frameTransmittedOnBus	A point in time where the transmission of the frame completes successfully, and the subscriber's communication controller receives the frame from the bus.

Table 5.23: TDEventFrameTypeEnum

Class	TDEventCycleStart (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::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			
Attribute	Datatype	Mul.	Kind	Note
cycleRepetition	Integer	1	attr	The start of every <cycleRepetition> cycle is targeted by this event.

Table 5.24: 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::TimingDescriptionEvents::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	Datatype	Mul.	Kind	Note
frCluster	FlexrayCluster	1	ref	The scope of this timing event.

Table 5.25: 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::TimingDescriptionEvents::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 , TimingDescription , TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
ttCanCluster	TtcanCluster	1	ref	The scope of this timing event.

Table 5.26: 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](#)
- EcuTiming [3.6](#)

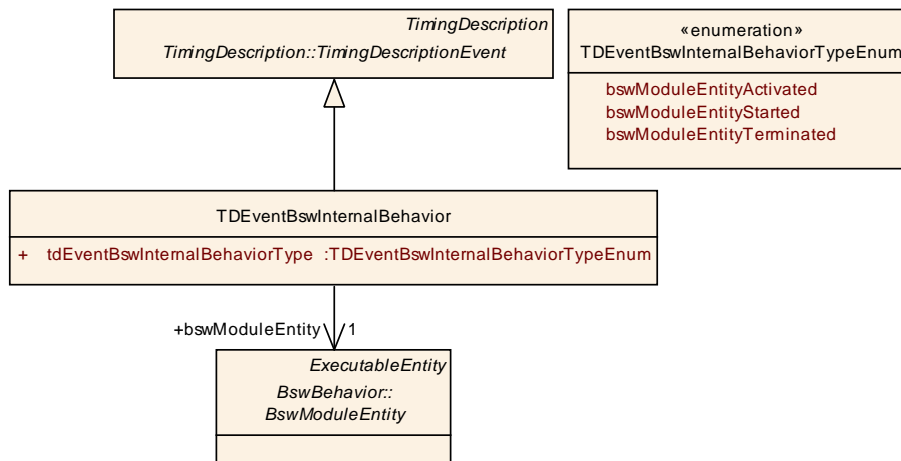


Figure 5.9: Events related to the internal structure of a BSW module

Class	TDEventBswInternalBehavior			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::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	Datatype	Mul.	Kind	Note
bswModuleEntity	BswModuleEntity	1	ref	The scope of this timing event.
tdEventBswInternalBehaviorType	TDEventBswInternalBehaviorTypeEnum	1	attr	The specific type of this timing event.

Table 5.27: 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::TimingDescriptionEvents::TDEventBswInternalBehavior	
Note	This is used to describe the specific event type of a TDEventBswInternalBehavior.	
Literal	Description	
bswModuleEntityActivated	A point in time where the associated BswModuleEntity has been activated, which means that it has entered the state "to be started".	
bswModuleEntityStarted	A point in time where the associated BswModuleEntity has entered the state "started" after its activation.	
bswModuleEntityTerminated	A point in time where the associated BswModuleEntity has terminated and entered the state "suspended"	

Table 5.28: 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](#))

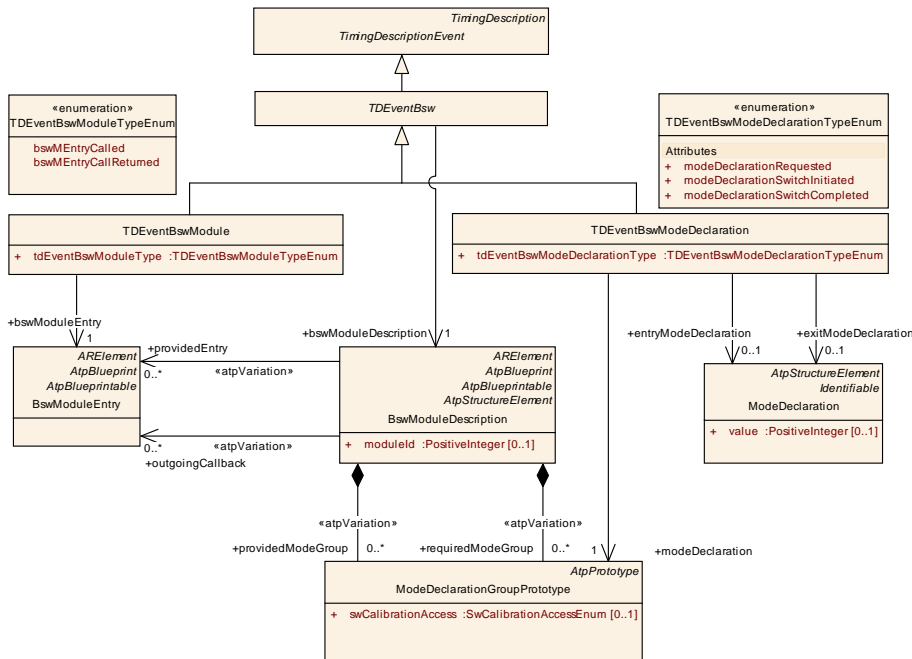


Figure 5.10: Events dealing with inter BSW module relations and mode communications on BSW level

[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	Datatype	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 5.29: 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

bswMEntry CallReturned	A point in time where the call of the associated BswModuleEntry has returned.
bswMEntry Called	A point in time where the associated BswModuleEntry has been called.

Table 5.30: 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::TimingDescriptionEvents::TDEventBsw			
Note	This is used to describe timing events related to the mode communication on BSW level.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, TDEventBsw, TimingDescription , TimingDescriptionEvent			
Attribute	Datatype	Mul.	Kind	Note
entryModeDeclaration	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.
exitModeDeclaration	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.
modeDeclaration	ModeDeclarationGroupPrototype	1	ref	The scope of this timing event.
tdEventBswModeDeclarationType	TDEventBswModeDeclarationTypeEnum	1	attr	The specific type of this timing event.

Table 5.31: TDEventBswModeDeclaration

Enumeration	TDEventBswModeDeclarationTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventBsw
Note	This is used to describe the specific event type of a TDEventBswModeDeclaration.
Literal	Description
modeDeclarationRequested	A point in time where the associated ModeDeclarationGroupPrototype has been requested.

modeDeclarationSwitch Completed	A point in time where the switch to the associated ModeDeclarationGroupPrototype has been completed.
modeDeclarationSwitch Initiated	A point in time where the switch to the associated ModeDeclarationGroupPrototype has been initiated by the BswM.

Table 5.32: 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](#)
- EcuTiming [3.6](#)

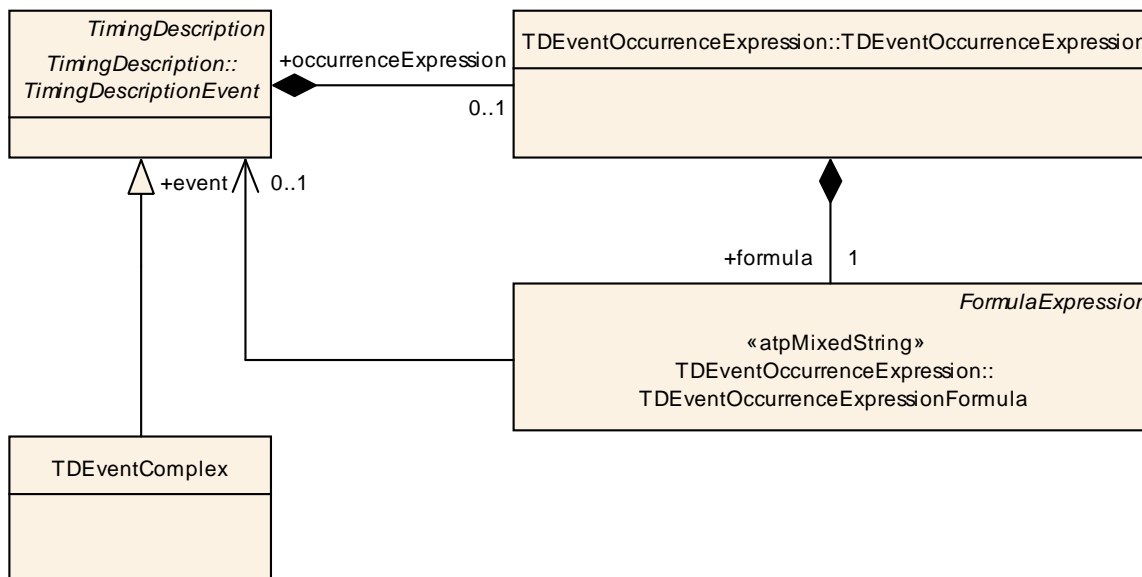


Figure 5.11: Complex timing event

Class	TDEventComplex			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::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	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 5.33: TDEventComplex

A complex timing event is a special observable event. In comparison to the "atomic" events described above (e.g. TDEventVariableDataPrototype), a complex event does not contain information about the context it references (e.g. like the VariableDataPrototype is a mandatory context information for events of type TDEventVariableDataPrototype). Instead, a complex event uses the occurrence expression (see next section 5.6) to specify the context by relating occurrences of usual (also called *atomic*) TimingDescriptionEvents.

5.6 Occurrence Expression Language for Timing Events

The TimingDescriptionEvents mentioned above allow to specify observable events with a well-defined context. However, sometimes the context information of the events is:

- too imprecise, e.g. only in case additional conditions (like a value filter) are valid, the observable event occurs.
- too specific, e.g. the stimulus of an event chain occurs not only when one of the atomic events mentioned above occurs, but also if other conditions are fulfilled.
- or both of these cases.

Thus, the occurrence expression provides means to cope with the limitations of atomic events mentioned above. It is an optional feature provided by the Timing Extensions which can be used in case the atomic events do not offer appropriate means to describe the desired timing behavior.

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

1. **Content Filter:** Filter occurrences of an atomic event based on the *value* of transmitted data or operation arguments.

2. **Complex Event Constructor:** Combine several atomic or complex events to a new complex event. Additionally the content filter can also be applied for complex events.

While the former case of filtering can be used for every `TimingDescriptionEvent`, the latter constructor can only be used for events of type `TDEventComplex` (see [constr_4500]).

5.6.1 Specifying an Occurrence Expression

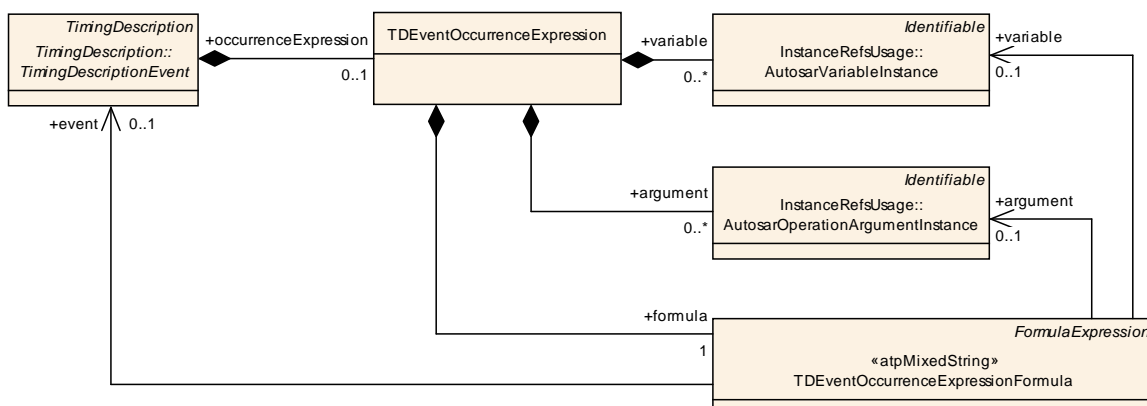


Figure 5.12: The occurrence expression

As shown in figure 5.12, 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`, a reference to an `AutosarOperationArgumentInstance` or a reference to an `ISignal` whose value shall be evaluated. The return type of this function depends on the operand (e.g. 0 or 1 if the referenced `AutosarVariableInstance`, `AutosarOperationArgumentInstance` or `ISignal` is of type boolean)
- 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

been occurred AT LEAST ONCE before (or at the same 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. It returns the time difference between a) the last occurrence of the referenced event and b) the point in time the expression is evaluated. The unit of time is seconds.
- 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. It returns the angle difference of the crank shaft between a) the last occurrence of the referenced event and b) the point in time the expression is evaluated. The unit of angle is degree.

The function *TIMEX_value* is used for specifying the "Content Filter" as specified above, all the other functions are required for the "Complex Event Constructor".

All operands required by the functions are references to model elements. Thus, `TDEventOccurrenceExpressionFormula` requires references to the respective elements of type `TimingDescriptionEvent`, `AutosarVariableInstance`, `AutosarOperationArgumentInstance` and `ISignal`. Due 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* and *TIMEX_angleSinceLastOccurrence* can only be used for occurrence expressions, which are applied to events of type `TDEventComplex`.]

The application of functions that require an event as operand only makes sense for occurrence expressions that are applied to events of type `TDEventComplex`. A complex event describes its occurrence behavior by relating the occurrences of atomic events. This is possible via the functions *TIMEX_occurs*, *TIMEX_hasOccurred*, *TIMEX_timeSinceLastOccurrence* and *TIMEX_angleSinceLastOccurrence*.

[constr_4501] Application rule for the occurrence expression [If the occurrence expression is applied for an event of type `TDEventComplex`, the expression must ensure the following criteria: a complex event can only occur at the occurrence time of one of the referenced `TimingDescriptionEvents` (via the "event" reference). This can e.g. be reached if the expression is defined as sum of products and each product uses the function *TIMEX_occurs* exactly once. Occurrence expressions, which do not satisfy this criteria, are invalid.]

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

In the following, an example is given that combines the functions mentioned above. The occurrence expression for a complex event *EC* shall be described. Figure 5.13 shows the software architecture required for this example.

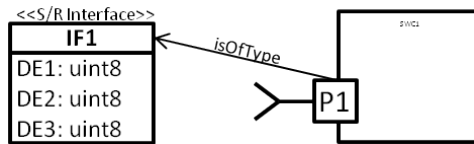


Figure 5.13: The software architecture used by the Occurrence Expression Example

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

Condition1 Either atomic event *E1* or *E2* must occur. In this example, *E1* (or *E2* respectively) is an atomic event which occurs, when the `VariableDataPrototype` *DE1* (or *DE2* respectively) is received on `PortPrototype` *P1* of software component *SWC1*.

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

Condition3 The `VariableDataPrototypes` *DE1* and *DE2* must be received at the same point in time (with a tolerance interval of 0.5 milliseconds). This is the case, when the atomic events *E1* and *E2* occur at the same point in time (with a tolerance interval 0.5 milliseconds).

This complex event *EC* would be described via the occurrence expression as follows:

```

//Condition 1
(TIMEX_occurs(event(/example/expression/E1))
 || TIMEX_occurs(event(/example/expression/E2)))
//Condition 2
&& TIMEX_value(variable(/example/expression/EC/D3))>3
//Condition 3
&& abs(TIMEX_timeSinceLastOccurrence(event(/example/expression/E1)) -
TIMEX_timeSinceLastOccurrence(event(/example/expression/E2))) <= 0.0005
  
```

Due to *Condition1*, the complex event *EC* can only occur when one of the atomic events *E1* or *E2* occurs. Thus, the expression satisfies the semantics constraint defined in [constr_4501].

The corresponding AUTOSAR XML file extract 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 UUID="49213604-5497-36bc-9955-816308da10c8">
      <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 UUID="fbc698cc-ebbd-3b22-a938-1caee6da50a4">
  <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(<EVENT-REF DEST="TD-EVENT-VARIABLE-DATA-
        PROTOTYPE"/>/example/expression/E1</EVENT-REF>))
      || TIMEX_occurs(<EVENT-REF DEST="TD-EVENT-VARIABLE-DATA-
        PROTOTYPE"/>/example/expression/E2</EVENT-REF>))
      && TIMEX_value(<VARIABLE-REF DEST="AUTOSAR-VARIABLE-INSTANCE"
        >/example/expression/EC/D3</VARIABLE-REF>) > 3
      && abs(TIMEX_timeSinceLastOccurrence(<EVENT-REF DEST="TD-
        EVENT-VARIABLE-DATA-PROTOTYPE"/>/example/expression/E1</
        EVENT-REF>))
      - TIMEX_timeSinceLastOccurrence(<EVENT-REF DEST="TD-EVENT-
        VARIABLE-DATA-PROTOTYPE"/>/example/expression/E2) &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::TimingDescriptionEvents::TDEventOccurrenceExpression			
Note	This is used to specify a filter on the occurrences of TimingDescriptionEvents by means of a TDEventOccurrenceExpressionFormula. 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	Datatype	Mul.	Kind	Note
argument	AutosarOperationArgumentInstance	*	aggr	An occurrence expression can reference an arbitrary number of OperationArgumentPrototypes in its expression. This association aggregates instanceRefs to OperationArgumentPrototypes which can be referenced in the expression.
formula	TDEventOccurrenceExpressionFormula	1	aggr	This is the expression formula which is used to describe the occurrence expression.

Attribute	Datatype	Mul.	Kind	Note
variable	AutosarVariableInstance	*	aggr	An occurrence expression can reference an arbitrary number of VariableDataPrototypes in its expression. This association aggregates instanceRefs to VariableDataPrototypes which can be referenced in the expression.

Table 5.34: TDEventOccurrenceExpression

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

Table 5.35: TDEventOccurrenceExpressionFormula

Class	AutosarVariableInstance			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventOccurrenceExpression::InstanceRefsUsage			
Note	<p>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:</p> <ul style="list-style-type: none"> • 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	Datatype	Mul.	Kind	Note
variableInstance	DataPrototype	1	iref	This is the reference to the instanceRef definition.

Table 5.36: AutosarVariableInstance

Class	AutosarOperationArgumentInstance			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventOccurrenceExpression::InstanceRefsUsage			
Note	<p>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:</p> <ul style="list-style-type: none"> • 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 ClientServerInterface 			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable			
Attribute	Datatype	Mul.	Kind	Note
operationArgumentInstance	DataPrototype	1	iref	This is the reference to the instanceRef definition.

Table 5.37: AutosarOperationArgumentInstance

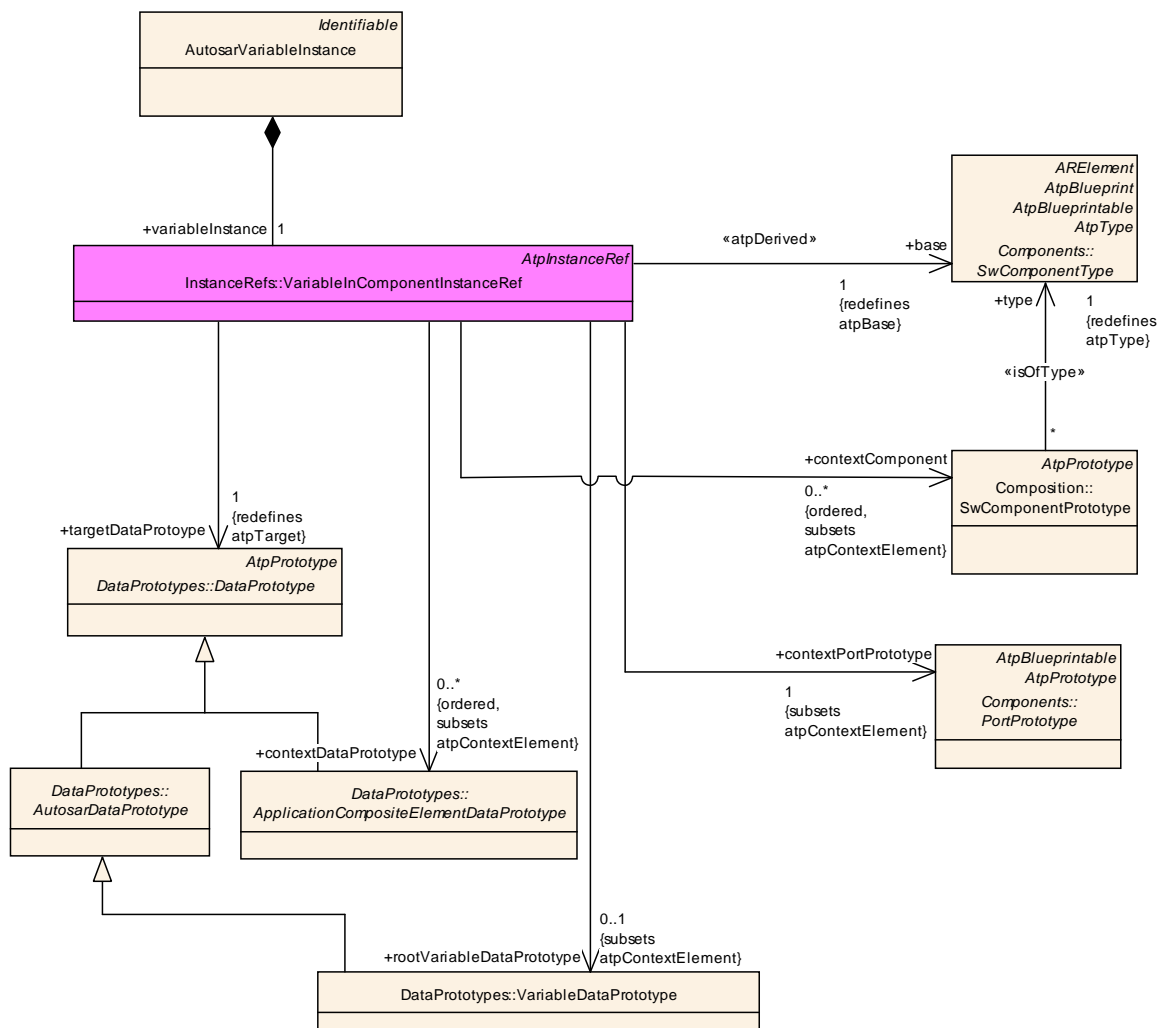


Figure 5.14: The required context information to reference to a variable instance within AUTOSAR.

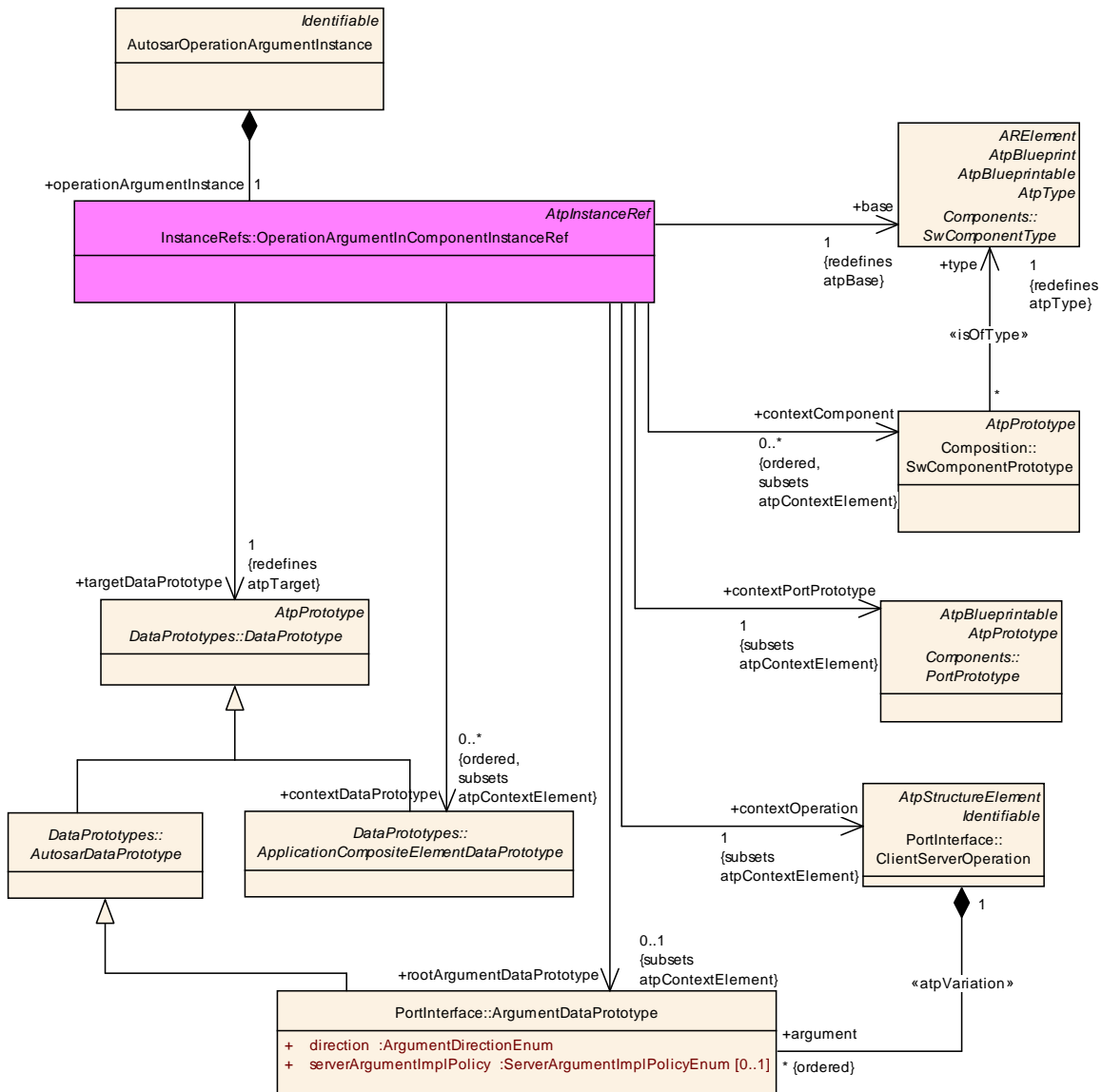


Figure 5.15: The required context information to reference to an operation argument instance within AUTOSAR.

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]. As mentioned above, it extends the language by a) additional functions and b) additional references to model elements. In the following, the implications of the extensions to the syntax is shown based on the grammar definition.

Note: the grammar defined for the formula language is not part of the listing below. It contains only the modifications of the formula language and the newly introduced functions and references.

Listing 5.2: AUTOSAR Occurrence Expression Language

```

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

```

5.6.3 Interpreting an Occurrence Expression

Based on the specification mechanism described above it is possible to use the occurrence expression formula to precise the timing specification. This way the developer is able to concertize the timing behavior he wants to constraint in case the atomic events described in the previous sections do not reflect the interested behavior appropriately.

This section describes how such an occurrence expression can be interpreted. The interpreter has the task to determine the occurrences of the `TimingDescription-Event`, for which the occurrence expression has been defined. This is done in two ways, depending on whether the occurrence expression is used as content filter or as complex event constructor. In the following, both cases are described in a more detailed way.

5.6.3.0.1 Interpreting a Content Filter

In this case, the occurrence expression is defined for an atomic event. From the newly added functions of the expression language, only *TIMEX_value(<reference to argument, variable or isignal>)* is allowed to be used for the content filter. On each occurrence of the atomic event, the interpreter checks whether the content filter defined via 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, when the atomic event occurs.

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

ISignal the value of the referenced isignal is evaluated at the point in time, when the atomic event occurs.

[constr_4503] Restricted usage of AutosarOperationArgumentInstance for Content Filter [If a content filter is defined for an atomic event, 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 `OperationPrototype`. Finally, references to an `AutosarOp-`

erationArgumentInstance 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).]

5.6.3.0.2 Interpreting a Complex Event Constructor

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 above, the interpreter "implements" a function $EC(t)$ which returns true, if the complex event EC occurs at time t :

```
EC(t) =
(TIMEX_occurs(t, event(/example/expression/E1))
 || TIMEX_occurs(t, event(/example/expression/E2)))
&& TIMEX_value(t, variable(/example/expression/EC/D3)) > 3
&& abs(TIMEX_timeSinceLastOccurrence(t, event(/example/expression/E1)) -
TIMEX_timeSinceLastOccurrence(t, event(/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 (or 1).

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 value of a referenced `ISignal` at time t
- the occurrences of a referenced `TimingDescriptionEvent` over time

There are different ways to gather the required information:

- **Model Analysis and Simulation:** In a deterministic system environment, occurrences of `TimingDescriptionEvents` can be determined offline (e.g. 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 e.g. by collecting information about the points in time, when the `TimingDescriptionEvent` has been occurred. Considering an event of type `TDE-`

ventSwcInternalBehavior, an ECU trace could contain a marker for each time the associated `RunnableEntity` has been 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_value(t, <reference to an ISignal>)`: returns the isignal 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 a) t and b) the last occurrence time of the referenced event before (or at) t . The unit of time is seconds.
- `TIMEX_angleSinceLastOccurrence(t, <reference to a TimingDescriptionEvent>)`: returns the angle difference between a) t and b) the last occurrence time of the referenced event before (or at) t . The unit of angle is degree.

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](#))

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, as timing constraints usually only refer to a specific part of the overall system's timing and need clear validity semantics.

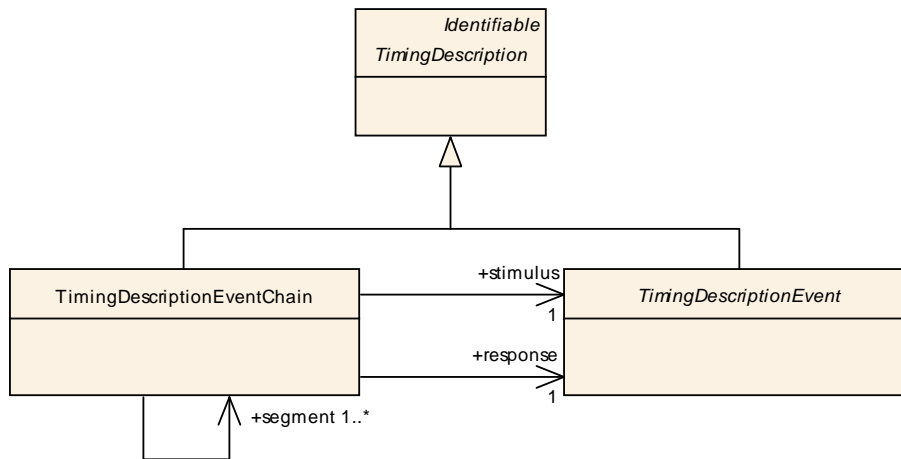


Figure 6.1: TimingDescriptionEventChain

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 <i>event chain segments</i> .			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable, TimingDescription			
Attribute	Datatype	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	TimingDescriptionEventChain	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

An event chain can be hierarchically decomposed into an arbitrary number of sub-chains, so called "event chain segments". It can also contain branch and junction points, so that multiple paths from a stimulus to a response can be described. However, all these paths must correspond to the same stimulus and the same response! Furthermore, loops are not allowed. If paths originating from different stimuli or paths leading to different responses need to be described, separate event chains must be defined.

Obviously, it only makes sense to specify an event chain between two events which have a relation in some sense. In the example shown in figure 6.2, a (indirect) relation

between the timing events "SensorDataSampled" and "ActuatorAccess" exists. In this case, an event chain between these two events is feasible.

In the case the event chain does not contain any event chain segments (sub-chains), it shall point to itself via the segment relation. This implies that the event chain is atomic and cannot be further decomposed.

[constr_4515] Specifying stimulus and response in TimingDescription-EventChain [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 TimingDescription-EventChain consists of further event chain segments then at least one sequence of event chain segments shall exist 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 TimingDescription-EventChain. 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.]

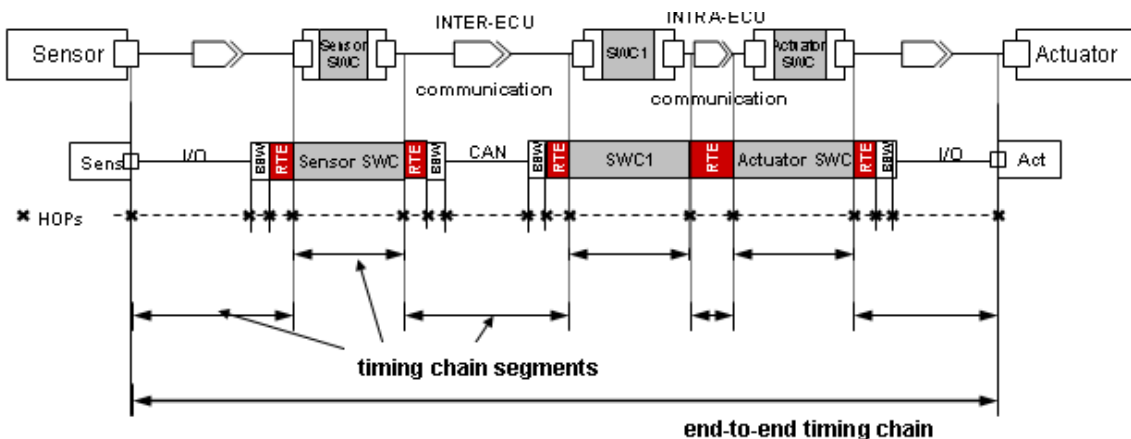


Figure 6.2: Example of an end-to-end event chain between sensor sampling and actuator access

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:

<i>Constraint</i>	<i>Imposed on</i>	<i>Use Case</i>
Event Triggering	TDEvent	Specification of an activation Model
Latency Timing	TDEventChain	End-to-End path latency (in reaction or max age semantics)
Age	TDEvent	Restriction
Synchronization Timing	TDEventChain	Restrictions for forks and joins of event chains
Synchronization Timing	TDEvent	Restriction
Offset Timing	TDEvent	Restriction
Execution Order	ExecutableEntity	Restriction
Execution Time	ExecutableEntity	Restriction

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.

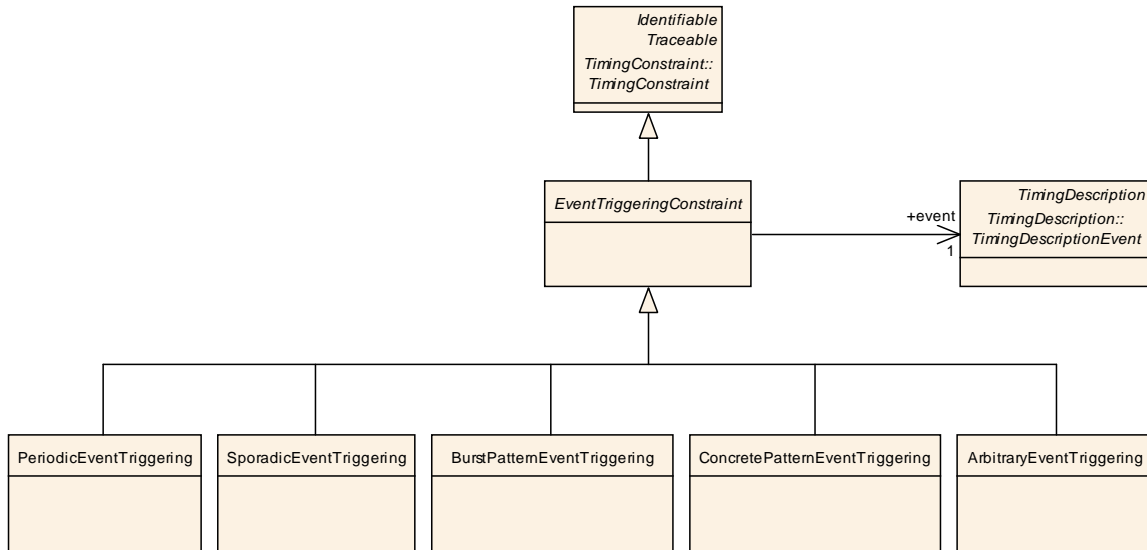


Figure 7.1: The different types of event triggerings

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			
Attribute	Datatype	Mul.	Kind	Note
event	TimingDescriptionEvent	1	ref	The referenced timing event

Table 7.1: 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](#))

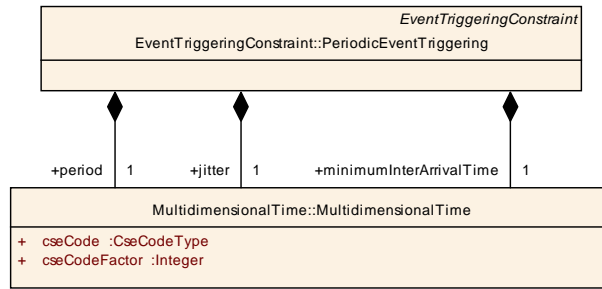


Figure 7.2: PeriodicEventTriggering

Class	PeriodicEventTriggering			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	<p>The PeriodicEventTriggering describes the behavior of an event with a strict periodic occurrence pattern, given by the period attribute.</p> <p>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.</p>			
Base	ARObject, EventTriggeringConstraint , Identifiable, MultilanguageReferrable, Referrable, TimingConstraint , Traceable			
Attribute	Datatype	Mul.	Kind	Note
jitter	MultidimensionalTime	1	aggr	<p>The maximum jitter of the periodic event occurrence.</p> <p>Tags: xml.sequenceOffset=20</p>
minimumInterArrivalTime	MultidimensionalTime	1	aggr	<p>The minimum time distance between two consecutive occurrences of the associated event.</p> <p>Tags: xml.sequenceOffset=10</p>
period	MultidimensionalTime	1	aggr	<p>The period of the event occurrence.</p> <p>Tags: xml.sequenceOffset=30</p>

Table 7.2: 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`.

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

$$AND \quad \forall n : t_{n+1} - t_n \leq minimumInterArrivalTime$$

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`.

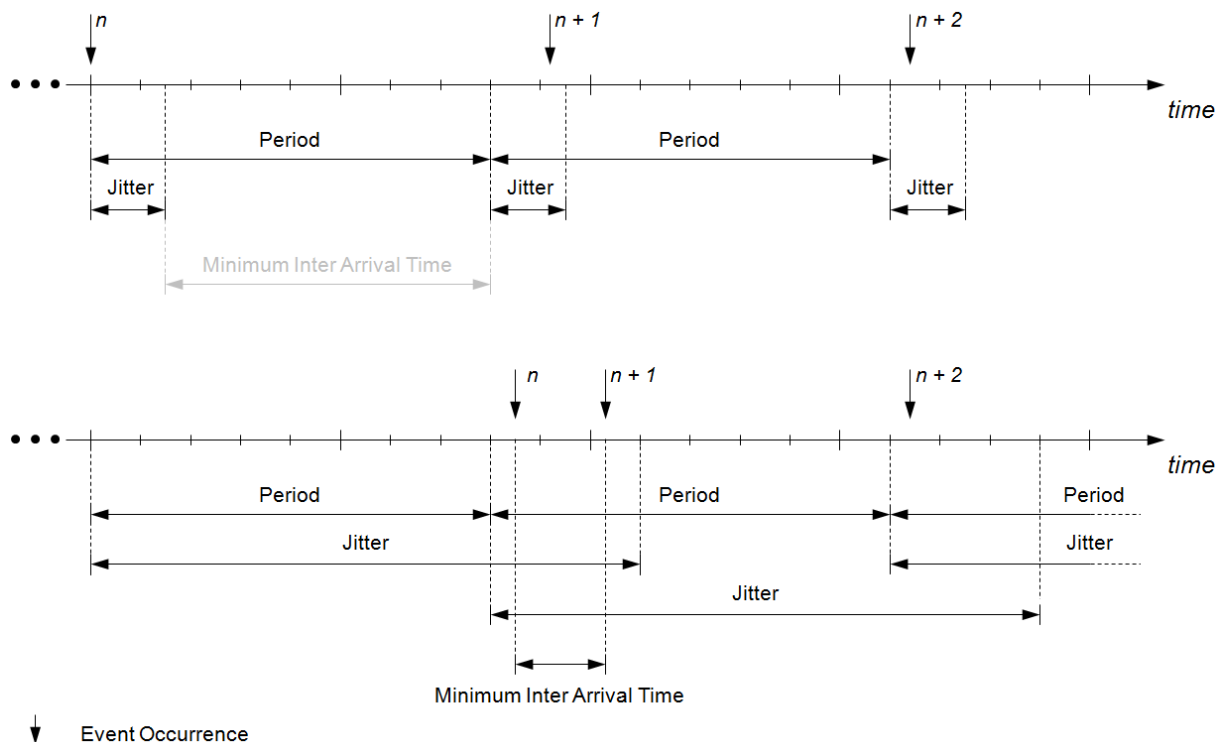


Figure 7.3: Parameters characterizing the Periodic Event Triggering

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.

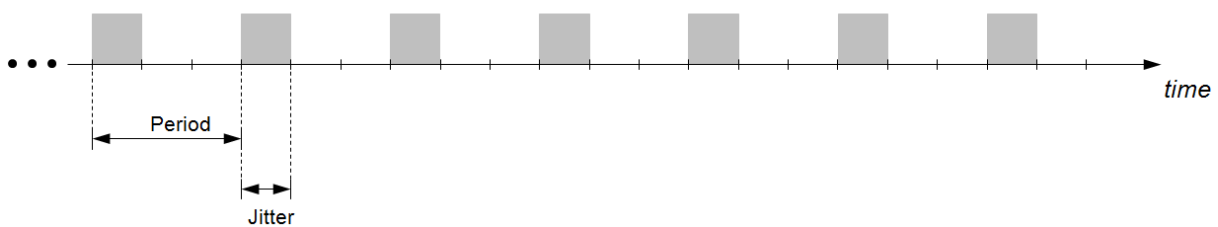


Figure 7.4: Example of a Periodic Event Triggering Constraint

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`.

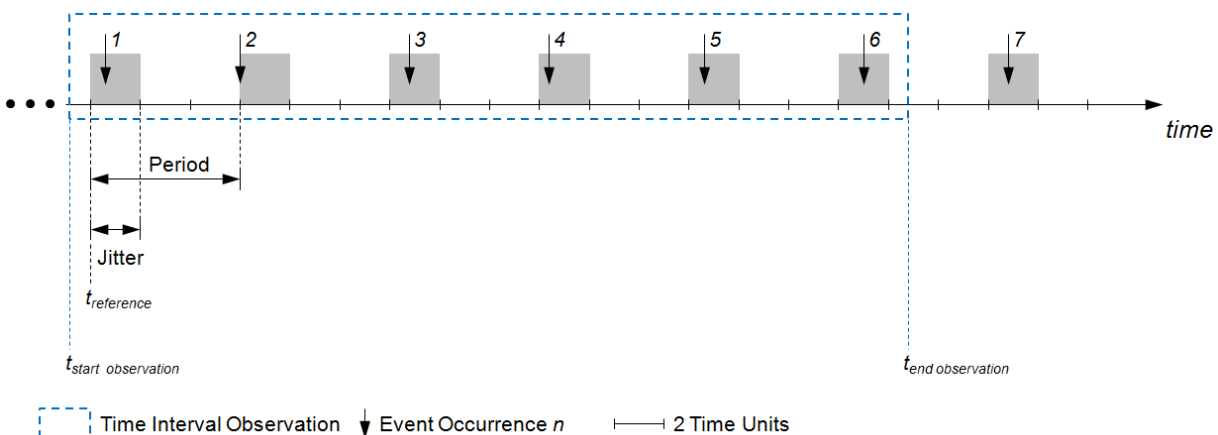


Figure 7.5: Event occurrences satisfying the given Periodic Event Triggering Constraint shown in 7.4

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.

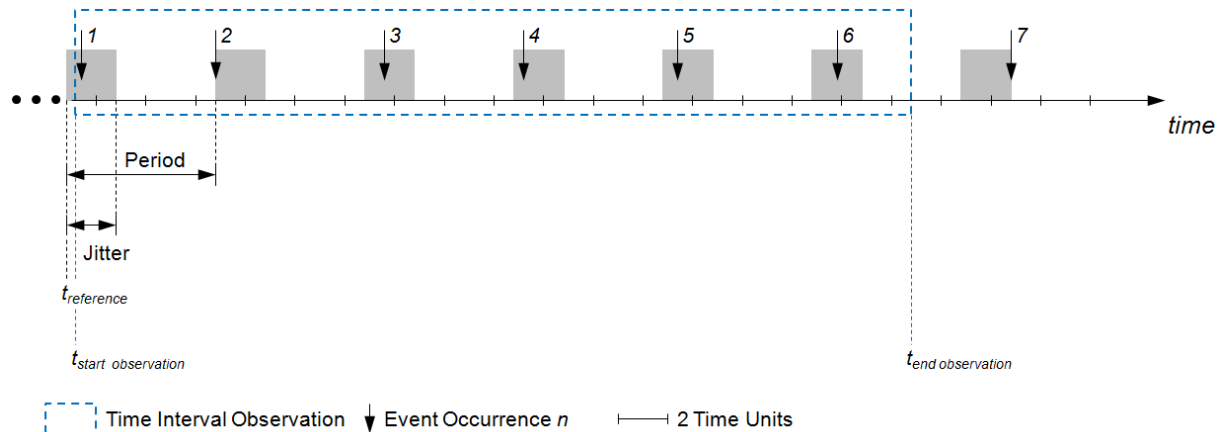


Figure 7.6: Event occurrences satisfying the given Period Event Triggering Constraint shown in 7.4, but with another reference point-in-time $t_{reference}$.

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`.

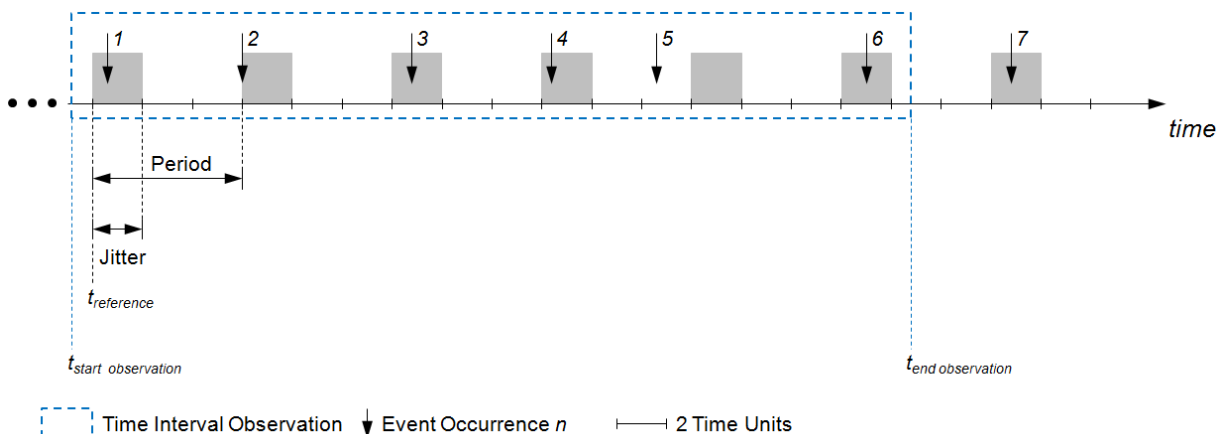


Figure 7.7: Event occurrences violating the given Period Event Triggering Constraint shown in 7.4

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.

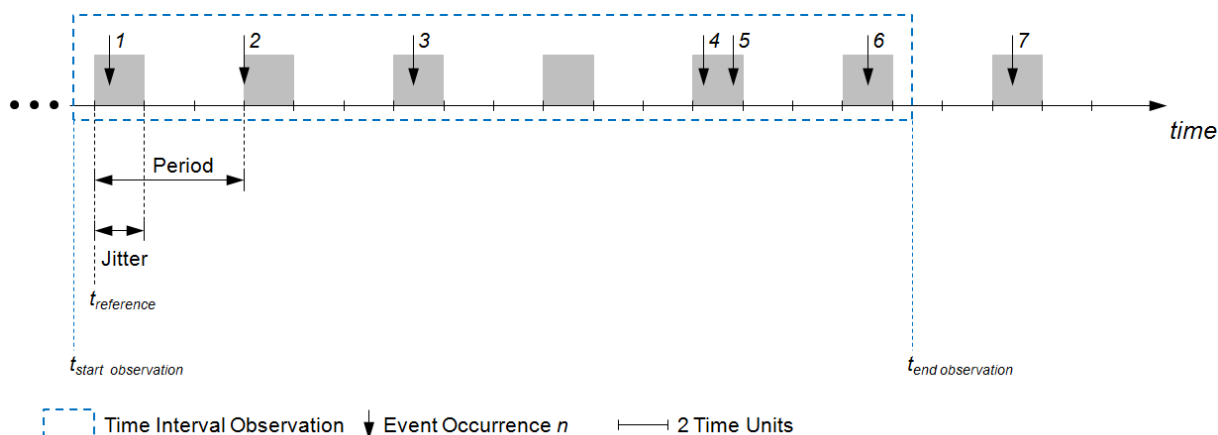


Figure 7.8: Event occurrences satisfying the given Period Event Triggering Constraint shown in 7.4.

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](#))

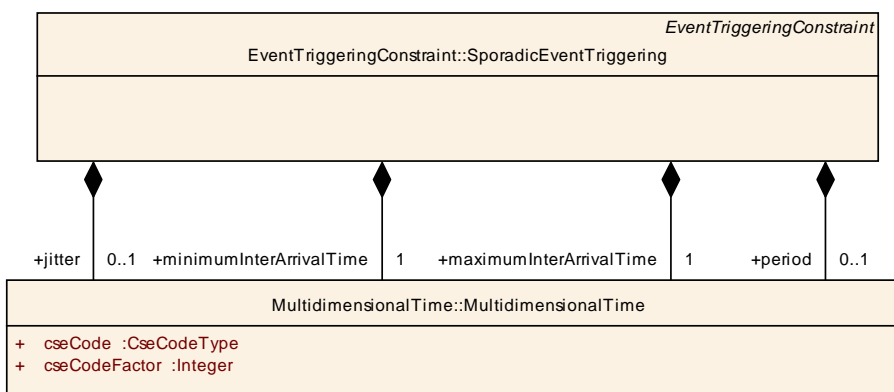


Figure 7.9: SporadicEventTriggering

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, Multilanguage Referrable, Referrable, TimingConstraint , Traceable			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	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 7.3: 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.

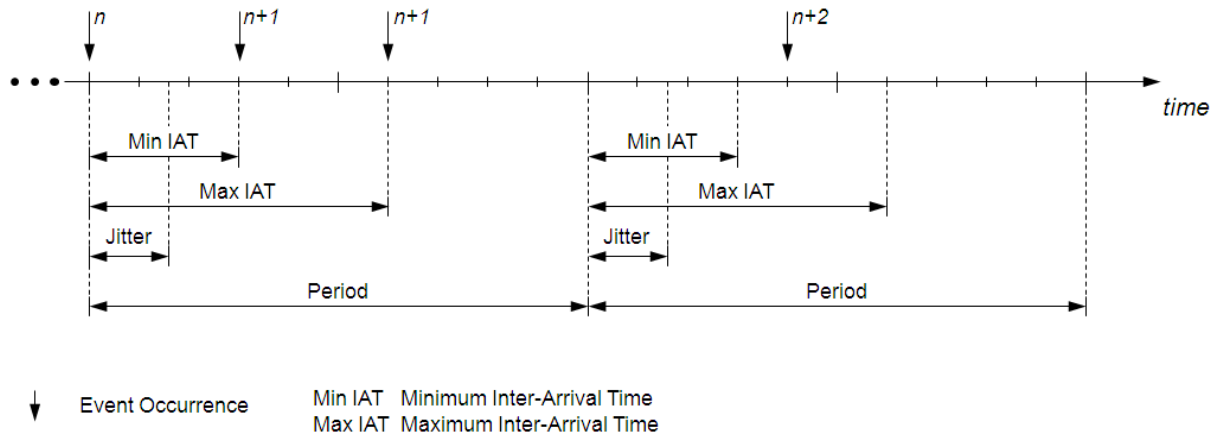


Figure 7.10: Parameters characterizing the Sporadic Event Triggering

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.

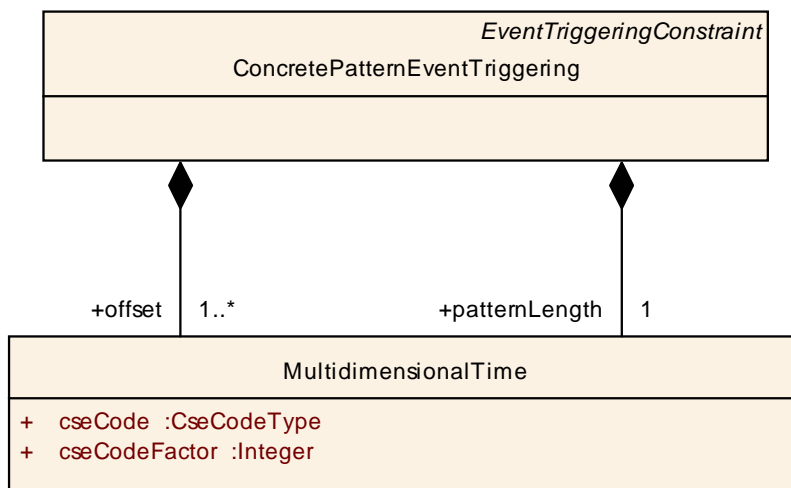


Figure 7.11: ConcretePatternEventTriggering

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	Datatype	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.role Element=true; xml.sequenceOffset=10; xml.type Element=false
patternLength	MultidimensionalTime	1	aggr	The length of the observed time interval. Tags: xml.sequenceOffset=20

Table 7.4: ConcretePatternEventTriggering

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

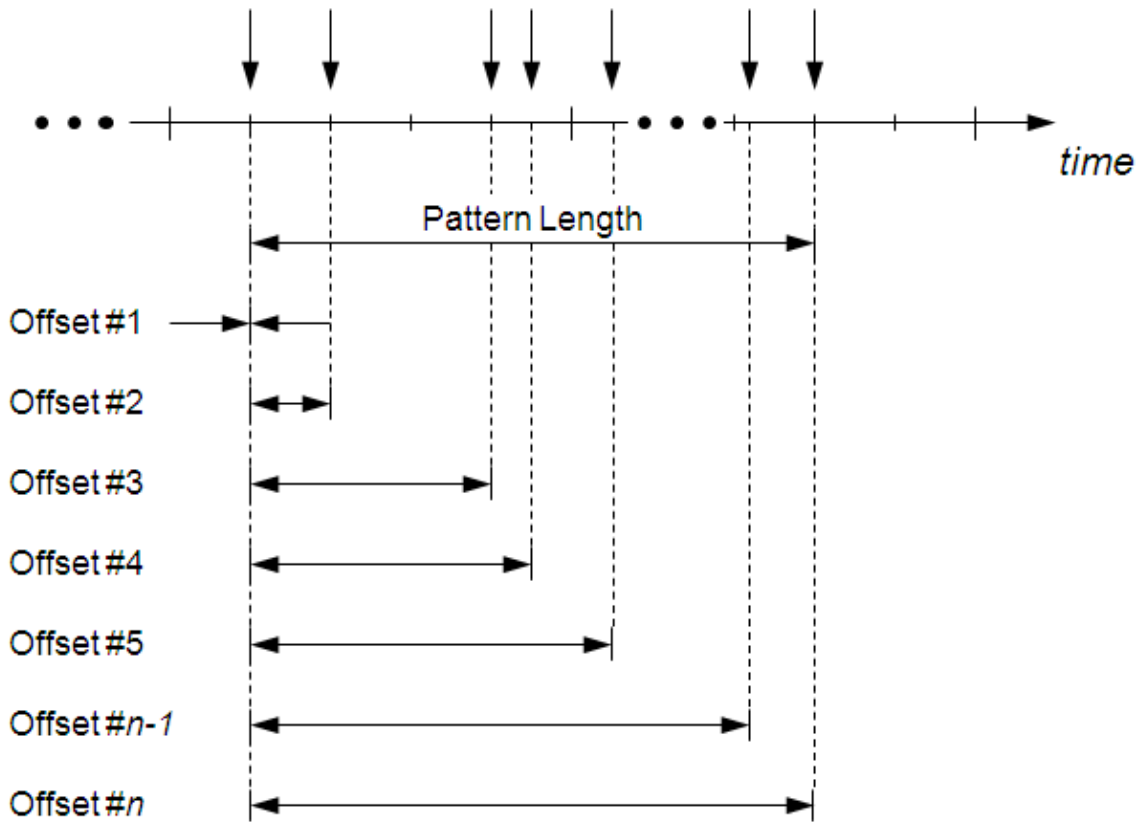
- Pattern Length
- Offset

These parameters are required ones and are described in the following. Figure 7.12 illustrates the parameters of the ConcretePatternEventTriggering.

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.

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



↓ Event Occurrence

Figure 7.12: Parameters characterizing the Concrete Pattern Event Triggering

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.

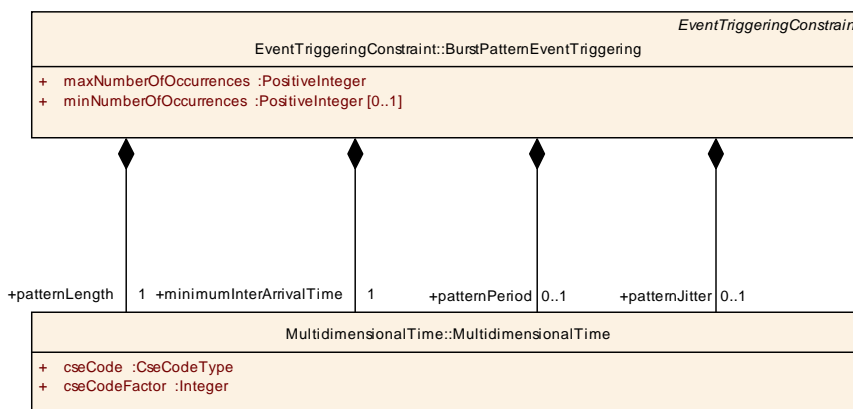


Figure 7.13: BurstPatternEventTriggering

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, Multilanguage Referrable, Referrable, TimingConstraint , Traceable			
Attribute	Datatype	Mul.	Kind	Note
maxNumberOfOccurrences	PositiveInteger	1	attr	The maximum number of event occurrences within the given time interval.
minNumberOfOccurrences	PositiveInteger	0..1	attr	The minimum number of event occurrences within the given time interval. Tags: xml.sequenceOffset=10
minimumInterArrivalTime	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".

<i>Attribute</i>	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>
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 7.5: BurstPatternEventTriggering

The parameters are described in the following and are illustrated in Figure 7.14 and Figure 7.15.

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`.

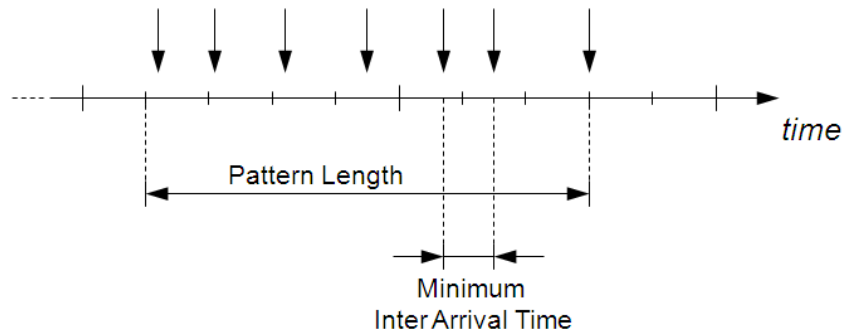
The constraints listed below apply to the `BurstPatternEventTriggering` and shall be considered when using this event triggering constraint.

[constr_4505] Specifying minimum and maximum number of occurrences [The minimum and maximum number of occurrences shall be specified such that the following holds true: $0 \leq \text{minNumberOfOccurrences} \leq \text{maxNumberOfOccurrences}$.]

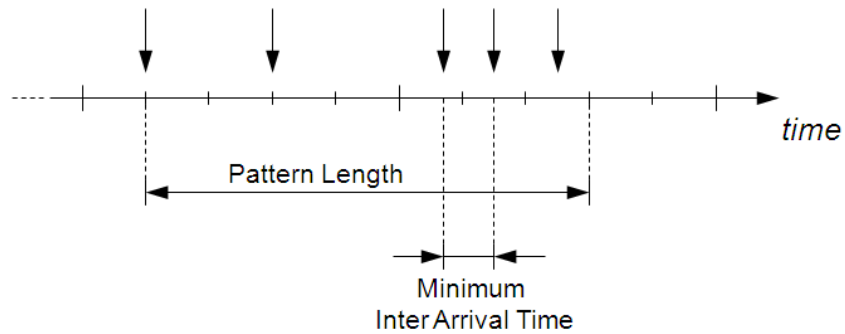
[constr_4506] Specifying minimum inter-arrival time and pattern length [The minimum inter-arrival time and pattern length shall be specified such that the following holds true: $0 < \text{minimumInterArrivalTime} \leq \text{patternLength}$.]

[constr_4507] Specifying pattern length, pattern jitter and pattern period [The pattern length, pattern jitter and pattern period shall be specified such that the following holds true: $\text{patternLength} + \text{patternJitter} < \text{patternPeriod}$.]

Maximum Number of Occurrences = 7



Minimum Number of Occurrences = 5 (optional)



↓ Event Occurrence

Figure 7.14: Parameters characterizing the Burst Pattern Event Triggering

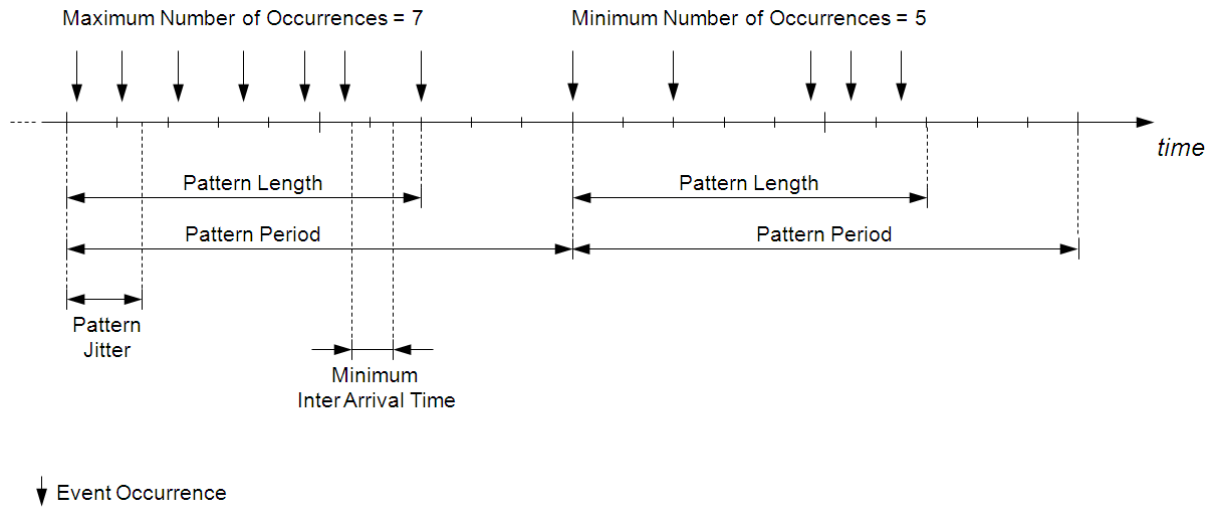


Figure 7.15: Parameters characterizing the Burst Pattern Event Triggering when periodically being repeated

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.

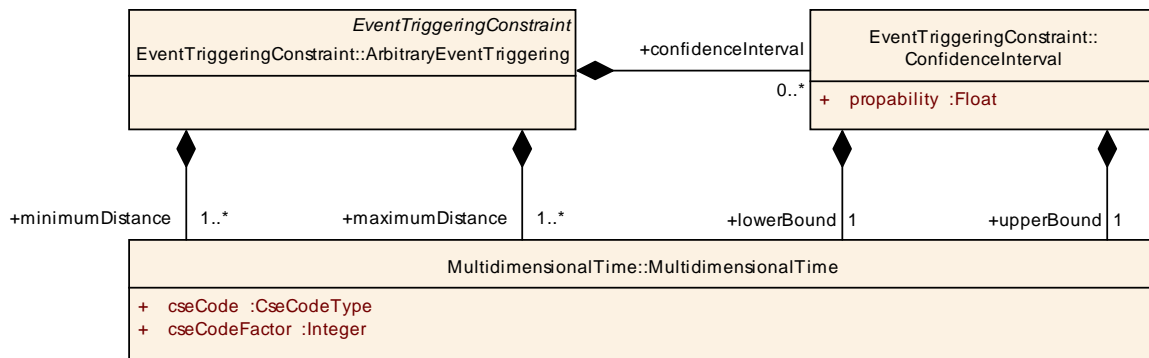


Figure 7.16: ArbitraryEventTriggering

Class	ArbitraryEventTriggering			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	<p>The ArbitraryEventTriggering describes that an event occurs occasionally, singly, irregularly or randomly.</p> <p>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.</p>			
Base	ARObject, EventTriggeringConstraint, Identifiable, MultilanguageReferrable, Referrable, TimingConstraint, Traceable			
Attribute	Datatype	Mul.	Kind	Note
confidenceInterval	ConfidenceInterval	*	aggr	List of confidence intervals. Tags: xml.sequenceOffset=30
maximumDistance	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
minimumDistance	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. Tags: xml.name=TIME-VALUE; xml.roleElement=true; xml.sequenceOffset=10; xml.typeElement=false

Table 7.6: 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	Datatype	Mul.	Kind	Note
lowerBound	MultidimensionalTime	1	aggr	The lower bound of the expected distance of two consecutive event occurrences.
probability	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.

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

Table 7.7: 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 7.17 illustrates the parameters of the `ArbitraryEventTriggering`.

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

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

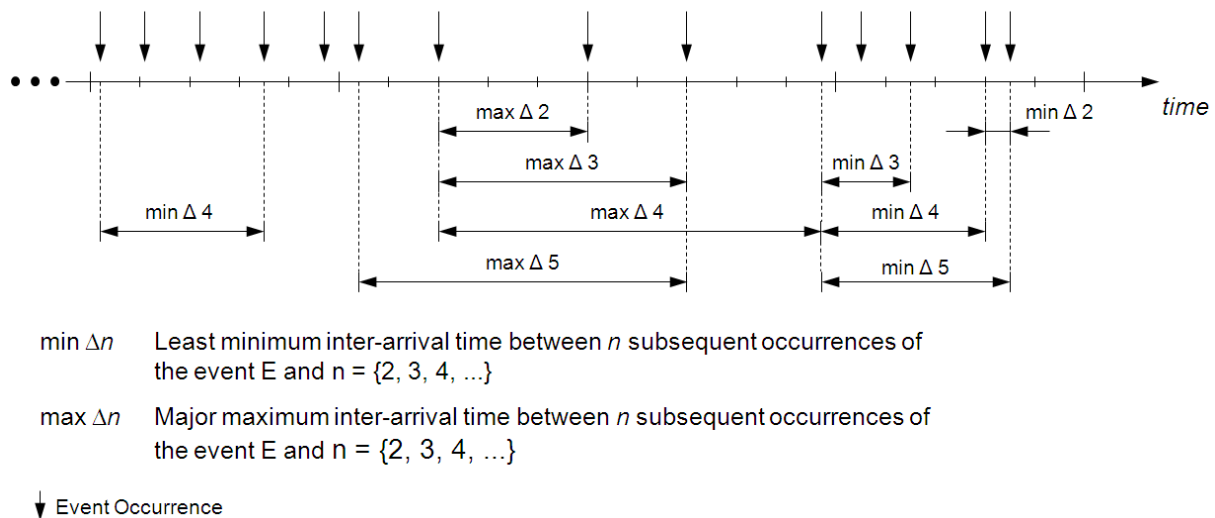


Figure 7.17: Parameters characterizing the Arbitrary Event Triggering

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](#))

¹A synonym for delay

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 7.18.

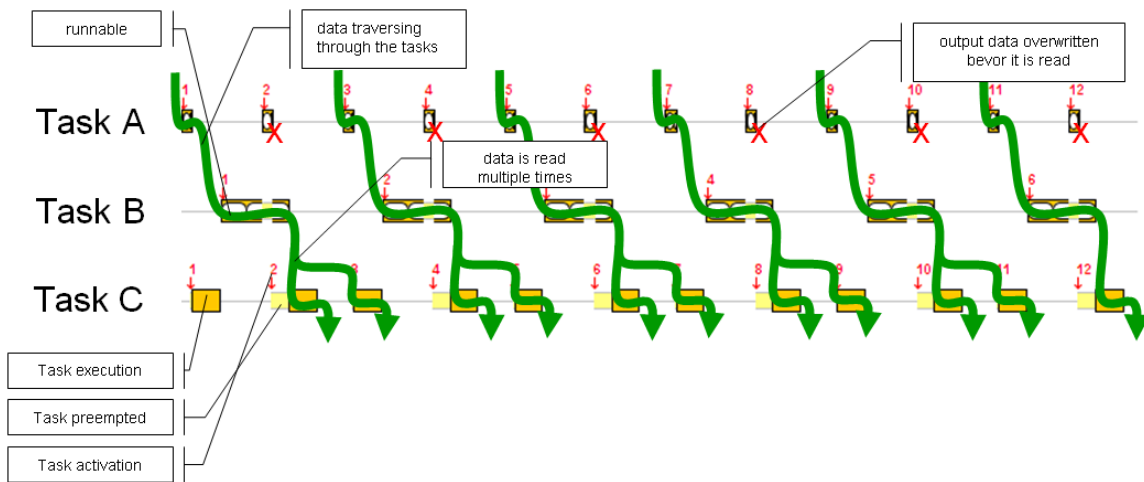


Figure 7.18: Loss and duplication of data due to under- and oversampling.

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.

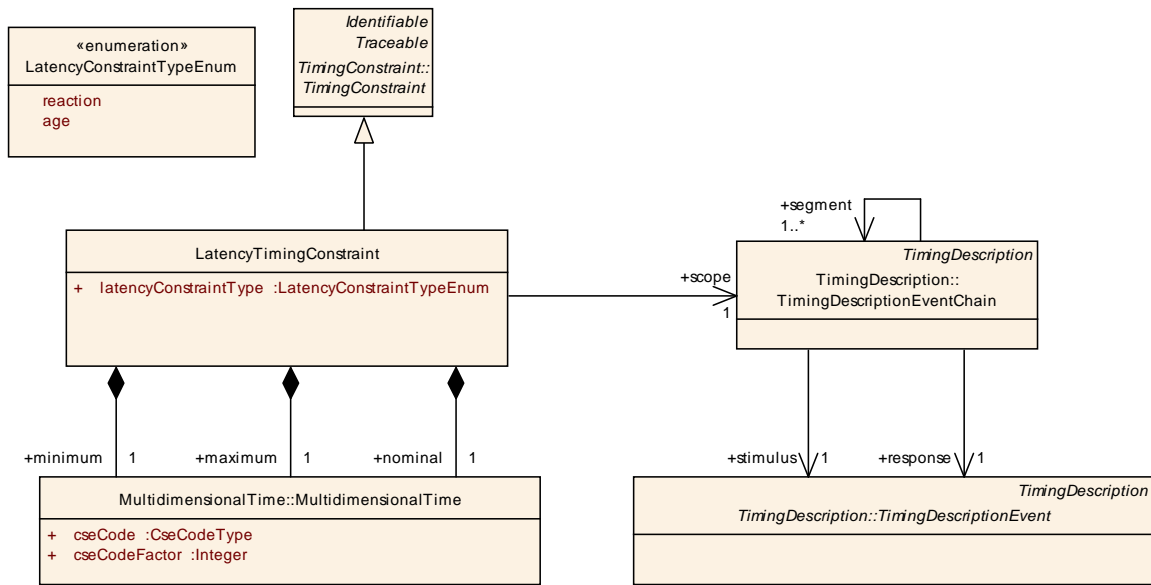


Figure 7.19: Latency constraint

Class	LatencyTimingConstraint			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::LatencyTimingConstraint			
Note	<p>This constraint type restricts the time duration between the occurrence of the stimulus and the occurrence of the corresponding response of that chain.</p> <p>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.</p> <p>In contrast to OffsetTimingConstraint, a causal dependency between the stimulus and response event of the associated event chain is required.</p>			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TimingConstraint,Traceable			
Attribute	Datatype	Mul.	Kind	Note
latencyConstraintType	LatencyConstraintTypeEnum	1	attr	The specific type of this latency constraint
maximum	MultidimensionalTime	1	aggr	<p>The maximum latency between the occurrence of the stimulus and the occurrence of the corresponding response of the associated event chain.</p> <p>Tags: xml.sequenceOffset=20</p>
minimum	MultidimensionalTime	1	aggr	<p>The minimum latency between the occurrence of the stimulus and the occurrence of the corresponding response of the associated event chain.</p> <p>Tags: xml.sequenceOffset=10</p>
nominal	MultidimensionalTime	1	aggr	<p>The nominal latency between the occurrence of the stimulus and the occurrence of the corresponding response of the associated event chain.</p> <p>Tags: xml.sequenceOffset=30</p>

Attribute	Datatype	Mul.	Kind	Note
scope	TimingDescriptionEventChain	1	ref	The event chain that defines the scope of the constraint.

Table 7.8: 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.
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.

Table 7.9: 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 `TDEventChain` 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 `TDEventChain` 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 `TEventVariablePrototype`. 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.

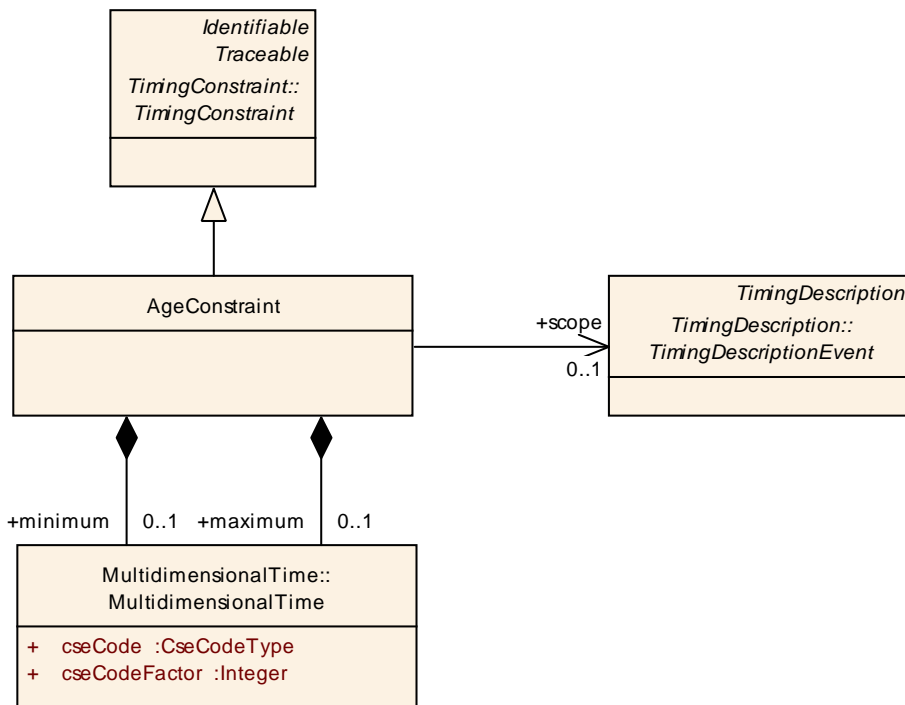


Figure 7.20: Age constraint

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

[constr_4504] Restricted usage of AgeConstraint [An `AgeConstraint` shall only be defined for events of type `TEventDescription` 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	Datatype	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 TimingDescriptionEvent that indicates any receipt of data.

Table 7.10: 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::SynchronizationTimingConstraint			
Note	<p>This constraint is used to restrict the timing behavior of different, but correlated events or event chains, with regard to synchronization.</p> <p>Thereby, in case of imposing a synchronization timing constraint on events or event chains the following two scenarios are supported:</p> <p>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.</p> <p>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.</p> <p>In case of imposing a synchronization timing constraint on events the following two scenarios are supported:</p> <p>1) [eventOccurrenceKind=singleOccurrence] Any of the events shall occur only once in the given time interval.</p> <p>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.</p>			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TimingConstraint,Traceable			
Attribute	Datatype	Mul.	Kind	Note
eventOccurrenceKind	EventOccurrenceKindEnum	0..1	attr	The specific occurrence kind of an event occurring within the given time interval.
scope	TimingDescriptionEventChain	*	ref	The event chain that is in the scope of the constraint.
scopeEvent	TimingDescriptionEvent	*	ref	The event that is in the scope of the constraint.
synchronizationConstraintType	SynchronizationTypeEnum	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 7.11: 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.
singleOccurrence	Specifies that an event shall occur only once in a given time interval.

Table 7.12: EventOccurrenceKindEnum

Enumeration	SynchronizationTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::SynchronizationTimingConstraint
Note	This is used to describe the type of the synchronization timing constraint.
Literal	Description
response Synchronization	<p>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.</p> <p>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.</p>
stimulusSynchronization	<p>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.</p> <p>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.</p>

Table 7.13: 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

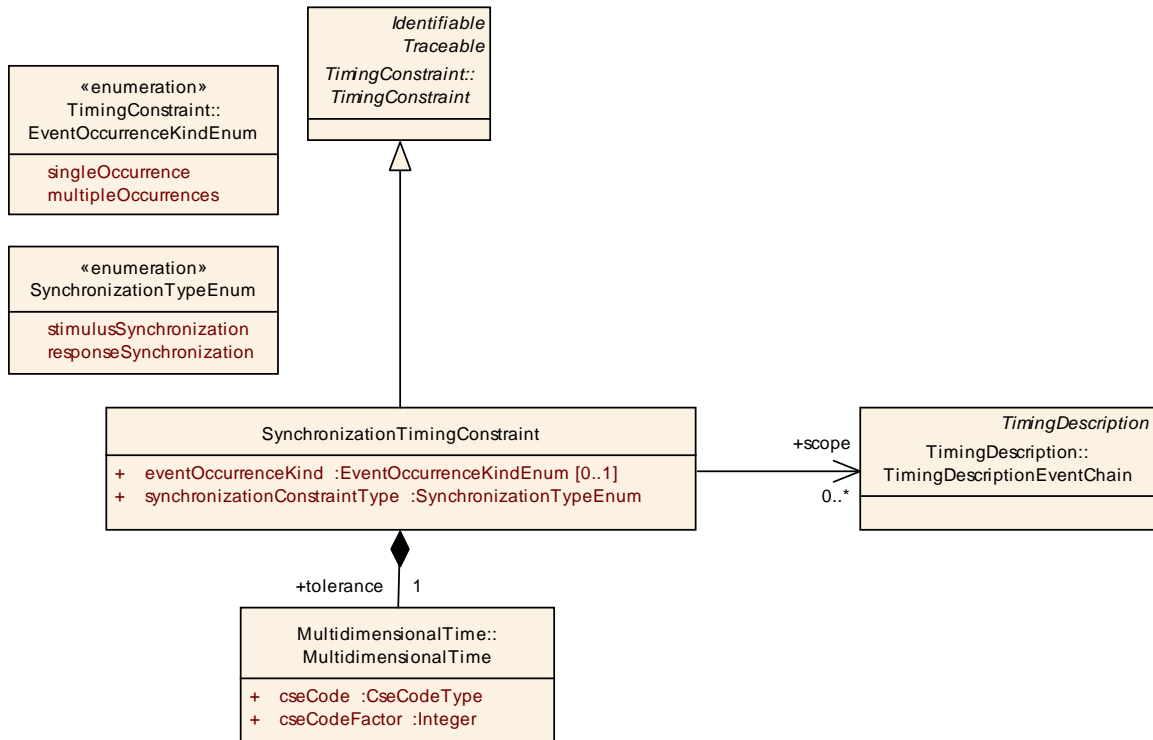


Figure 7.21: Synchronization Timing Constraint on Event Chains

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 7.22 and Figure 7.23.

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.]

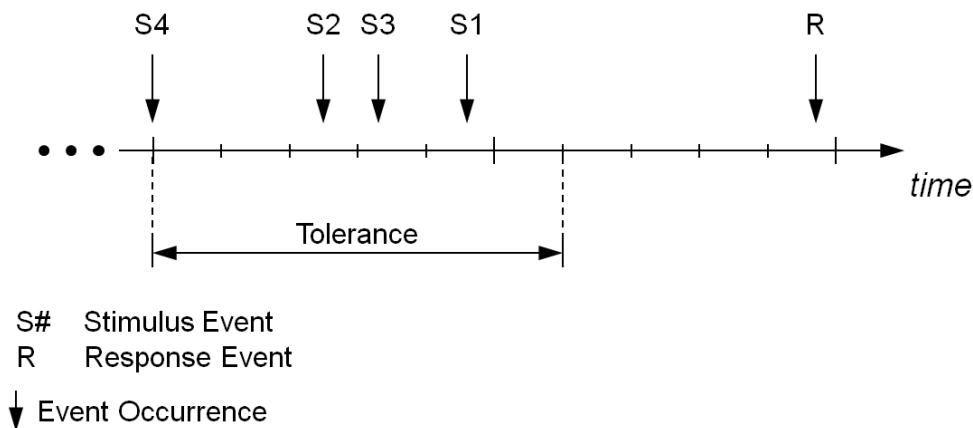


Figure 7.22: Parameters characterizing the Synchronization Timing Constraint imposed on the stimulus events of event chains.

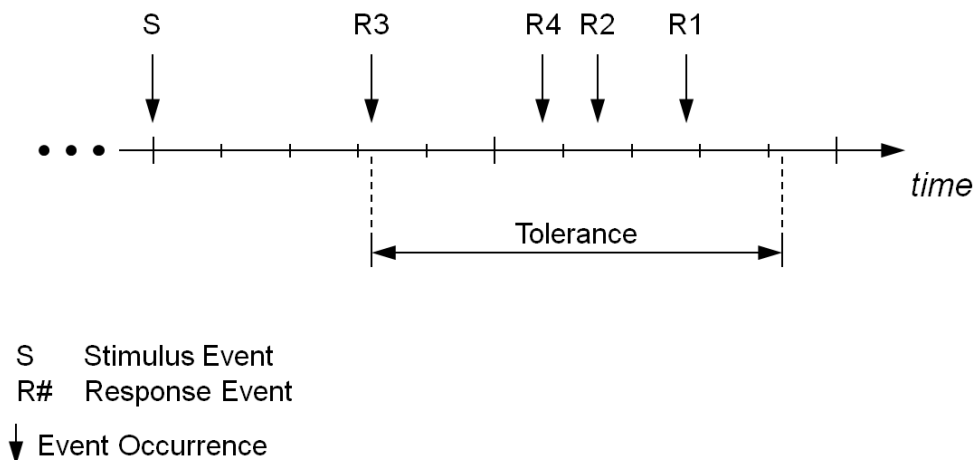


Figure 7.23: Parameters characterizing the Synchronization Timing Constraint imposed on the response events of event chains.

An example for a `stimulusSynchronization` 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 a `responseSynchronization` 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.

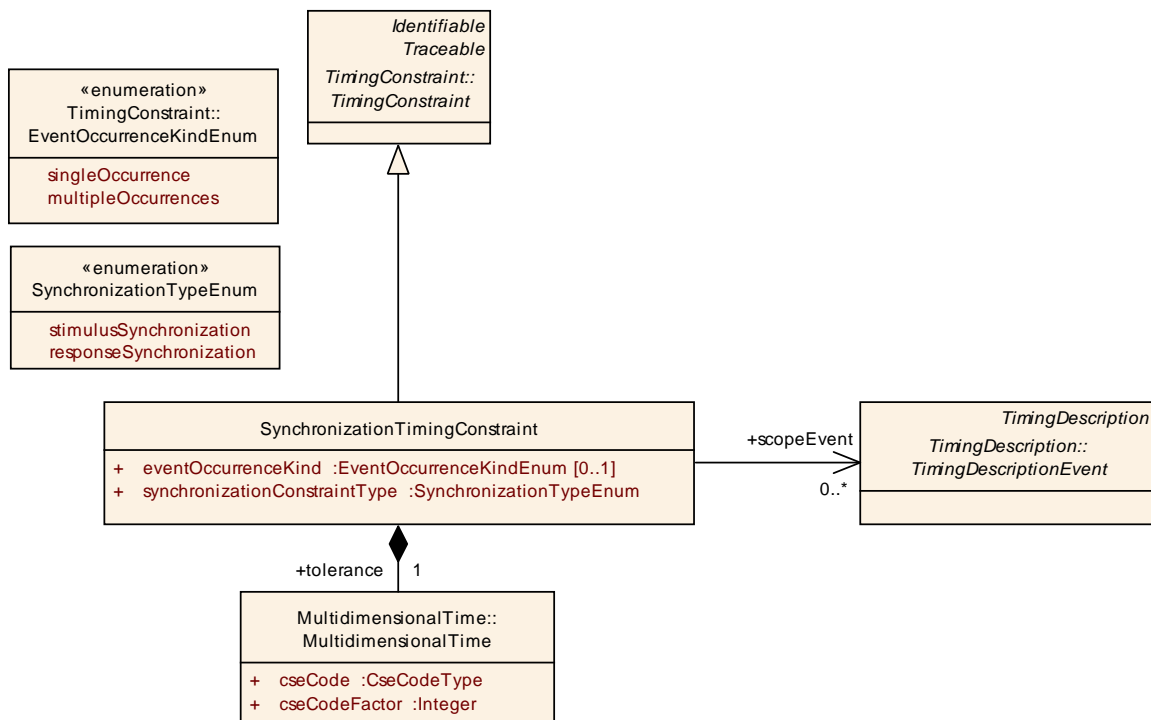


Figure 7.24: Synchronization Timing Constraint on Events

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 7.25.

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.]

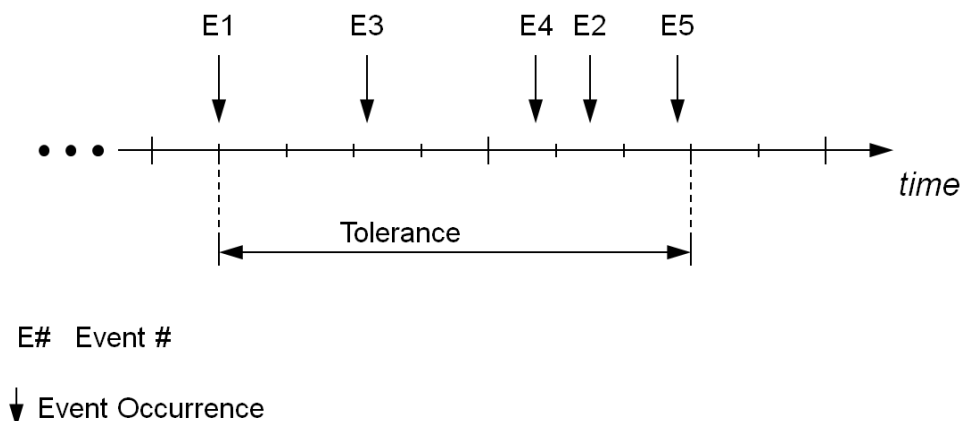


Figure 7.25: Parameter characterizing the Synchronization Constraint

7.5 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.

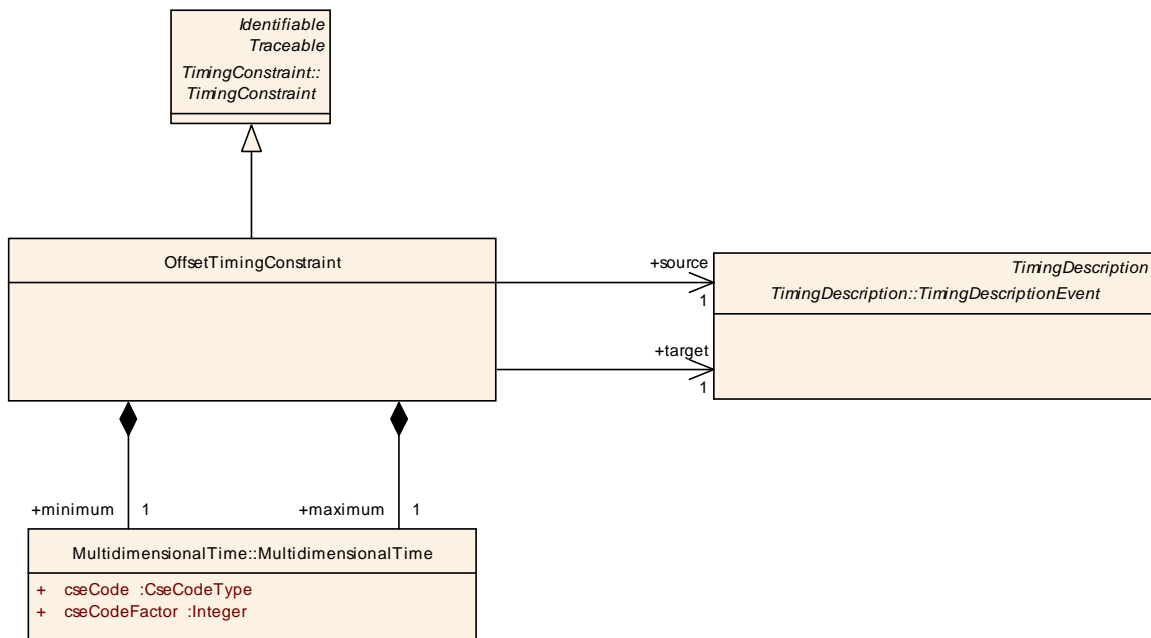


Figure 7.26: Offset Timing Constraint

Class	OffsetTimingConstraint			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::OffsetConstraint			
Note	<p>Bounds the time offset between the occurrence of two timing events, without requiring a direct functional dependency between the source and the target.</p> <p>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.</p> <p>In contrast to LatencyTimingConstraint, there must not necessarily be a causal dependency between the source and target event.</p>			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TimingConstraint,Traceable			
Attribute	Datatype	Mul.	Kind	Note
maximum	MultidimensionalTime	1	aggr	<p>The maximum offset the target event occurs relatively after the occurrence of the source event.</p> <p>Tags: xml.sequenceOffset=20</p>
minimum	MultidimensionalTime	1	aggr	<p>The minimum offset the target event occurs relatively after the occurrence of the source event.</p> <p>Tags: xml.sequenceOffset=10</p>
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 7.14: OffsetTimingConstraint

7.6 ExecutionOrderConstraint

[TPS_TIMEX_00007] ExecutionOrderConstraint specifies sequence of executing executable entities [The element `ExecutionOrderConstraint` is used to specify the order of execution of `ExecutableEntities`.]([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 `ExecutableEntities` to be referenced are available in other AUTOSAR models, namely the Software Component Description and the Basic Software Module Description.

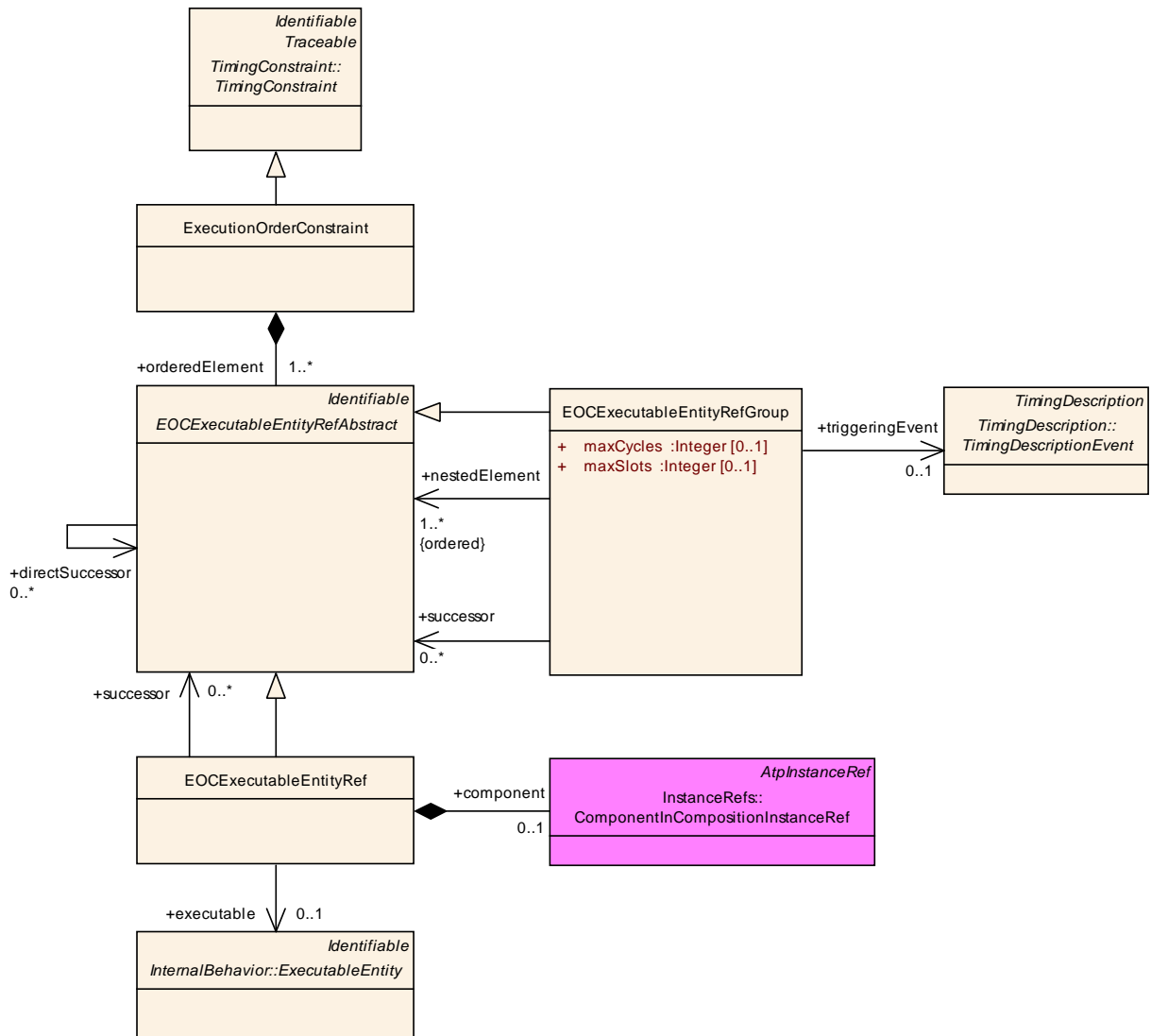


Figure 7.27: Specification of execution order constraints.

Class	ExecutionOrderConstraint			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint			
Note	<p>This constraint is used to restrict the order of execution for a set of ExecutableEntities. The ExecutionOrderConstraint can be used in any timing view.</p> <p>On VFB level an ExecutionOrderConstraint can be specified for RunnableEntities part of an InternalBehavior of AtomicSwComponentTypes which are part of the SwComponentType referenced by the VfbTiming. The ExecutionOrderConstraint is aggregated by the VfbTiming.</p> <p>On SW-C level an ExecutionOrderConstraint can be specified for RunnableEntities part of the InternalBehavior referenced by the SwcTiming. The ExecutionOrderConstraint is aggregated by the SwcTiming.</p> <p>On System level an ExecutionOrderConstraint can be specified for RunnableEntities part of an InternalBehavior of AtomicSwComponentTypes which are part of the RootSwCompositionPrototype of the system referenced by the SystemTiming. The ExecutionOrderConstraint is aggregated by the SystemTiming.</p> <p>On BSW Module level, an ExecutionOrderConstraint can be specified for BswModuleEntities part of an BswInternalBehavior referenced by the BswModuleTiming. The ExecutionOrderConstraint is aggregated by the BswModuleTiming.</p> <p>On ECU level an ExecutionOrderConstraint can be specified for all ExecutableEntities which are mapped to an ECU — RunnableEntities — and/or are part of the ECU's basic software — BswModuleEntities. In the latter case the EcuTiming references the particular EcucValueCollection containing the configuration of all BSW modules. The ExecutionOrderConstraint is aggregated by the EcuTiming.</p>			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TimingConstraint,Traceable			
Attribute	Datatype	Mul.	Kind	Note
orderedElement	EOCExecutableEntityRefAbstract	1..*	aggr	The list of references to ExecutableEntities which shall be ordered.

Table 7.15: ExecutionOrderConstraint

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			
Attribute	Datatype	Mul.	Kind	Note
directSuccessor	EOCExecutableEntityRefAbstract	*	ref	The direct successor of an executable entity or a group of executable entities.

Table 7.16: 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	Datatype	Mul.	Kind	Note
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.
nested Element (ordered)	EOCExecutableEntityRefAbstract	1..*	ref	This association is used to establish hierarchies of EOCEER Groups and References.
successor	EOCExecutableEntityRefAbstract	*	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 7.17: 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 SwComponentPrototype, via the ComponentInCompositionInstanceRef, the referenced ExecutableEntity belongs to, must be provided as context information.			
Base	ARObject,EOCExecutableEntityRefAbstract,Identifiable,MultilanguageReferrable,Referrable			
Attribute	Datatype	Mul.	Kind	Note
component	SwComponentPrototype	0..1	iref	This defines the context of the referenced RunnableEntity on system level.
executable	ExecutableEntity	0..1	ref	The ExecutableEntity whose execution order is restricted by the constraint.
successor	EOCExecutableEntityRefAbstract	*	ref	The logical successor of an executable entity or a group of executable entities.

Table 7.18: EOCExecutableEntityRef

[TPS_TIMEX_00038] Purpose of EOCExecutableEntityRefAbstract [The element EOCExecutableEntityRefAbstract is an abstract class that represents *both*

primitives `EOCExecutableEntityRef` and their composites `EOCExecutableEntityRefGroup`. Essentially, it is used to compose `EOCExecutableEntityRefGroups` and `EOCExecutableEntityRefs` into tree structures to represent part-whole hierarchies.]([RS_TIMEX_00001](#), [RS_TIMEX_00002](#), [RS_TIMEX_00014](#))

[TPS_TIMEX_00041] Purpose of EOCExecutableEntityRefGroup [The element `EOCExecutableEntityRefGroup` is used to define composites of `EOCExecutableEntityRefGroups` and `EOCExecutableEntityRefs`.]([RS_TIMEX_00001](#), [RS_TIMEX_00002](#), [RS_TIMEX_00014](#))

[TPS_TIMEX_00046] Purpose of EOCExecutableEntityRef [The element `EOCExecutableEntityRef` is used to reference executable entities (`EOCExecutableEntity`) which shall be executed in a specific order.]([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 `EOCExecutableEntityRefs`, between two `EOCExecutableEntityRefGroups`, or between an `EOCExecutableEntityRef` and an `EOCExecutableEntityRefGroup` is one (1).]

[constr_4534] Maximum number of directSuccessor relationships [The number of `directSuccessor` relationships of an `EOCExecutableEntityRef` or an `EOCExecutableEntityRefGroup` shall not exceed the number of independent execution units available in a system.]

[constr_4535] Same Mode of ExecutableEntities [In an `ExecutionOrderConstraint` the `ExecutableEntities` referenced by all `EOCExecutableEntityRefs` shall be active in the same mode.]

[constr_4536] Compatible recurrence of ExecutableEntities [In an `ExecutionOrderConstraint` the `ExecutableEntities` referenced by all `EOCExecutableEntityRefs` shall be compatible with regard to their recurrence.]

[constr_4537] References among elements in an ExecutionOrderConstraint [An `EOCExecutableEntityRef` or an `EOCExecutableEntityRefGroup` shall ref-

erence only `EOCExecutableEntityRefs` or `EOCExecutableEntityRefGroups` which are part of the same `ExecutionOrderConstraint`.]

7.6.1 Ordinary Execution Order Constraint

The *Ordinary* Execution Order Constraint is used to specify an order of execution of executable entities. As sketched in Figure 7.28 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.

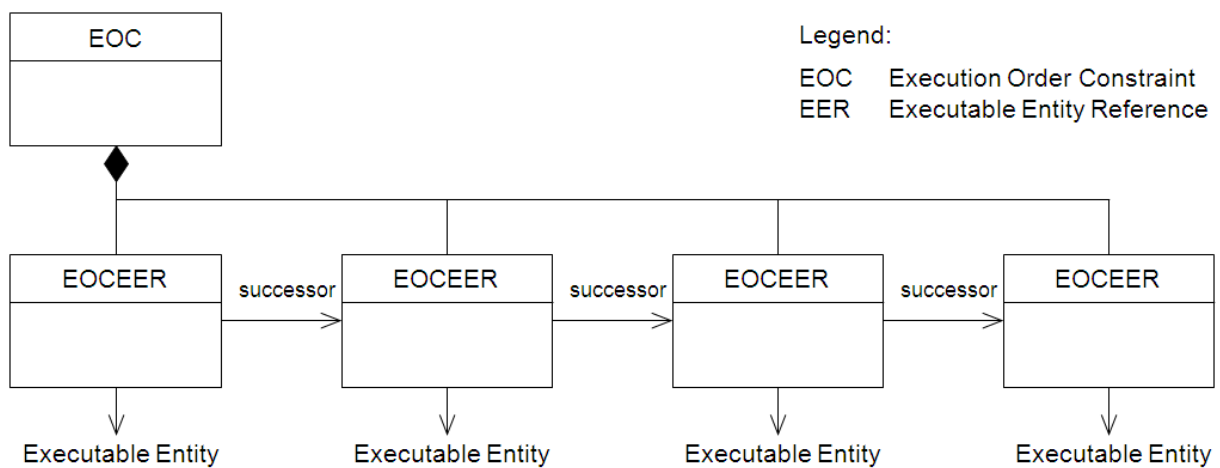


Figure 7.28: Example of an Ordinary Execution Order Constraint

[constr_4541] `EOCExecutableEntityRef` shall reference `ExecutableEntity` in Ordinary Execution Order Constraint [In an Ordinary Execution Order Constraint all `EOCExecutableEntityRefs` shall reference `ExecutableEntities`.]

7.6.2 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 one to specify tree-like structures of executable entity references and executable entity reference groups. As sketched in Figure 7.29 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.

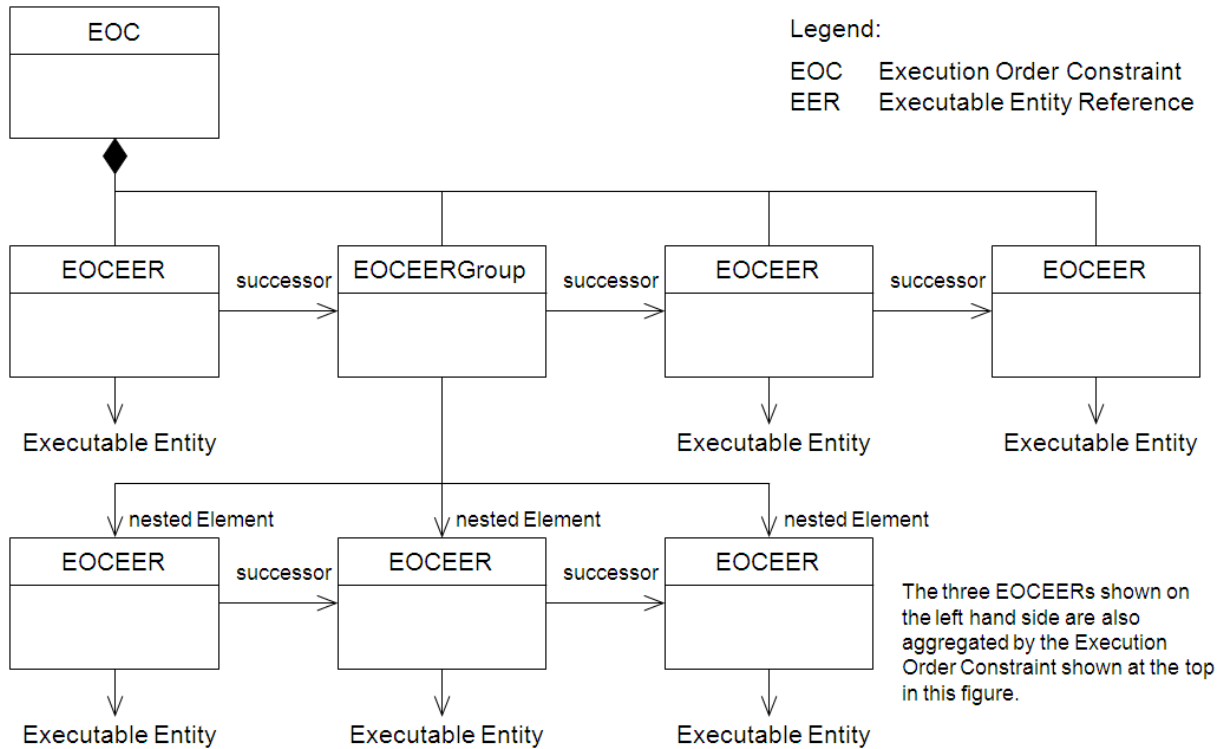


Figure 7.29: Example of a Hierarchical Execution Order Constraint

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 `EOCEexecutableEntityRefGroup` that is part of a hierarchical execution order constraint.]

[constr_4524] Referencing `TimingDescriptionEvent` [Any element `EOCEexecutableEntityRefGroup` that is part of a hierarchical execution order constraint shall not reference any timing description event `TimingDescriptionEvent`.]

[constr_4538] Hierarchical Execution Order Constraint: `EOCEexecutableEntityRef` and `EOCEexecutableEntityRefGroup` shall be target or source of a successor relationship [In a given Hierarchical Execution Order Constraint, each `EOCEexecutableEntityRef` and `EOCEexecutableEntityRefGroup` which is not part of an `EOCEexecutableEntityRefGroup` shall be target or source of at least one successor relationship. The *root* `EOCEexecutableEntityRefGroup` is excluded from this constraint.]

[constr_4542] `EOCEexecutableEntityRef` shall reference `ExecutableEntity` in Hierarchical Execution Order Constraint [In an Hierarchical Execution Order Constraint all `EOCEexecutableEntityRefs` shall reference `ExecutableEntities`.]

7.6.3 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 7.30 the Repetitive Execution Order Constraints follows a specific pattern.

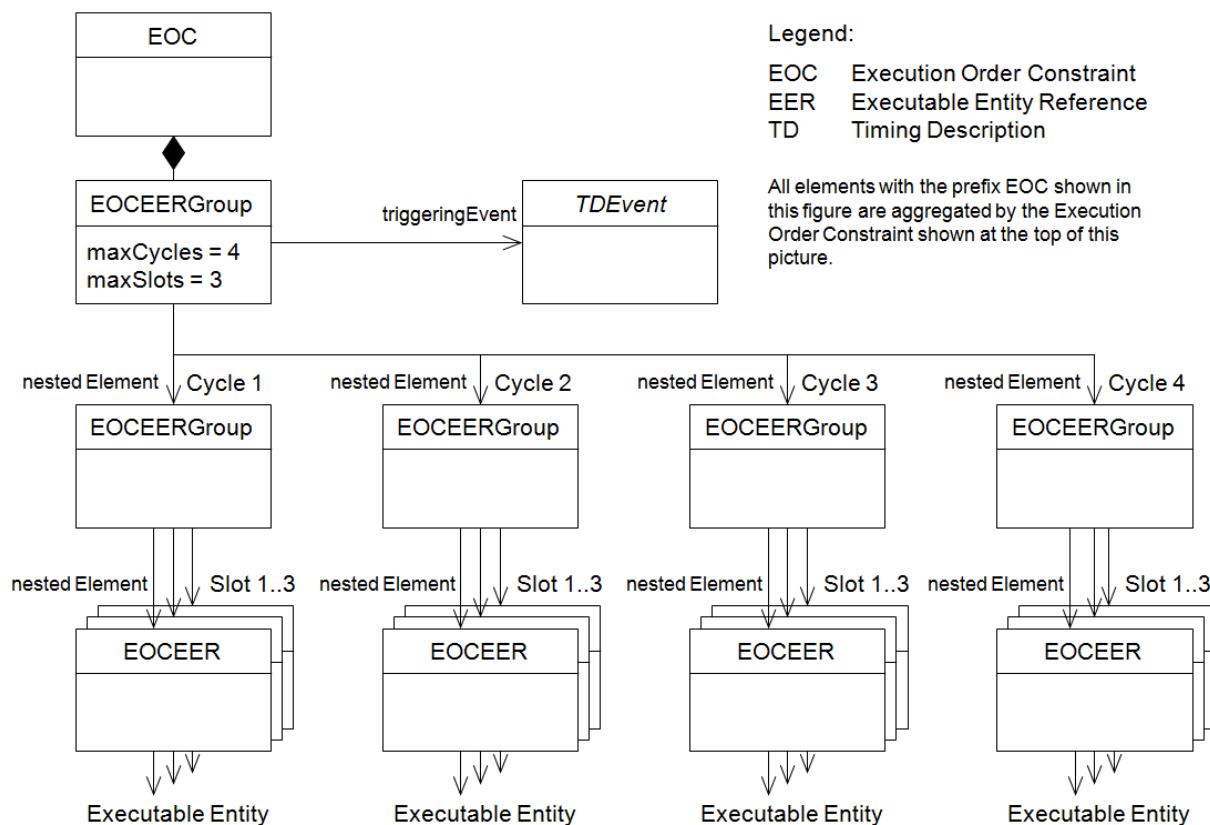


Figure 7.30: Example of a Repetitive Execution Order Constraint

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 which eventually reference the specific executable entities. 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 7.30 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

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 `EOCExecutableEntityRefGroups` [The *root* `EOCExecutableEntityRefGroup` shall reference only groups of executable entity references `EOCExecutableEntityRefGroups`.]

[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 `EOCExecutableEntityRefs` [The `EOCExecutableEntityRefGroup` representing a cycle shall reference only executable entity references `EOCExecutableEntityRefs`.]

[constr_4531] Number of nested elements referenced by `EOCExecutableEntityRefGroup` representing a cycle [The number of nested elements referenced by a

EOExecutableEntityRefGroup 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).]

7.7 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](#))

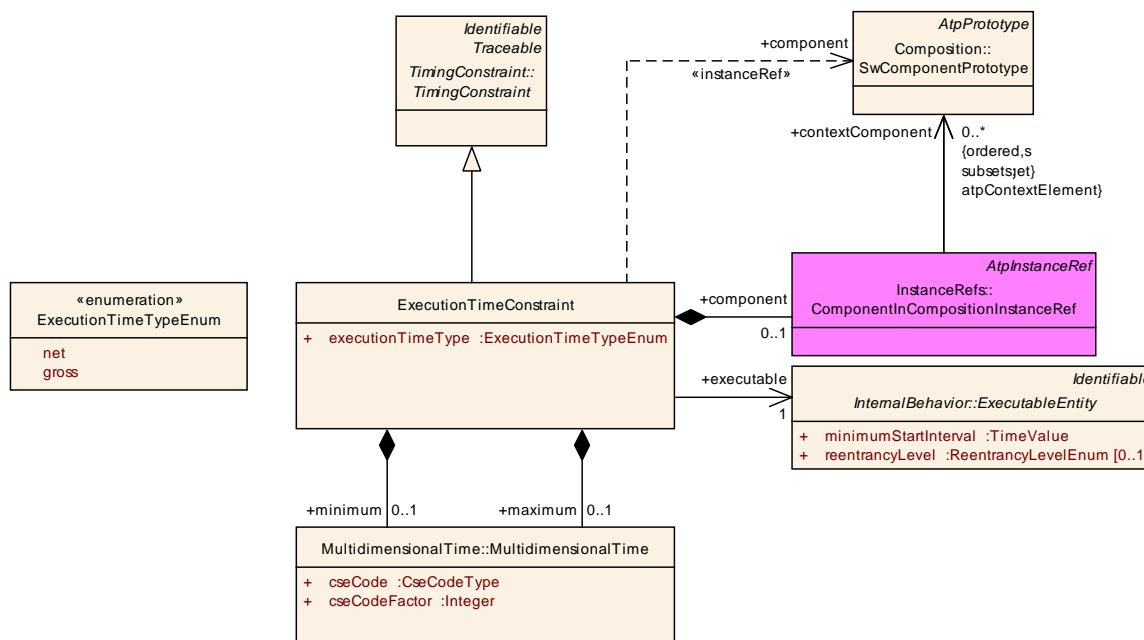


Figure 7.31: Execution time constraint

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	<p>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.</p> <p>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.</p> <p>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.</p>			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TimingConstraint,Traceable			
Attribute	Datatype	Mul.	Kind	Note
component	SwComponentPrototype	0..1	iref	The component that contains the referenced ExecutableEntity 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 ExecutionTimeConstraint.
executionTimeType	ExecutionTimeTypeEnum	1	attr	
maximum	MultidimensionalTime	0..1	aggr	The maximum execution time.
minimum	MultidimensionalTime	0..1	aggr	The minimum execution time.

Table 7.19: ExecutionTimeConstraint

Enumeration	ExecutionTimeTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionTimeConstraint
Note	
Literal	Description
gross	Indicates that the given execution time is the time used to execute the ExecutableEntity without any interruption and including external calls.
net	Indicates that the given execution time is the time used to execute the ExecutableEntity without any interruption and without any external calls.

Table 7.20: 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.

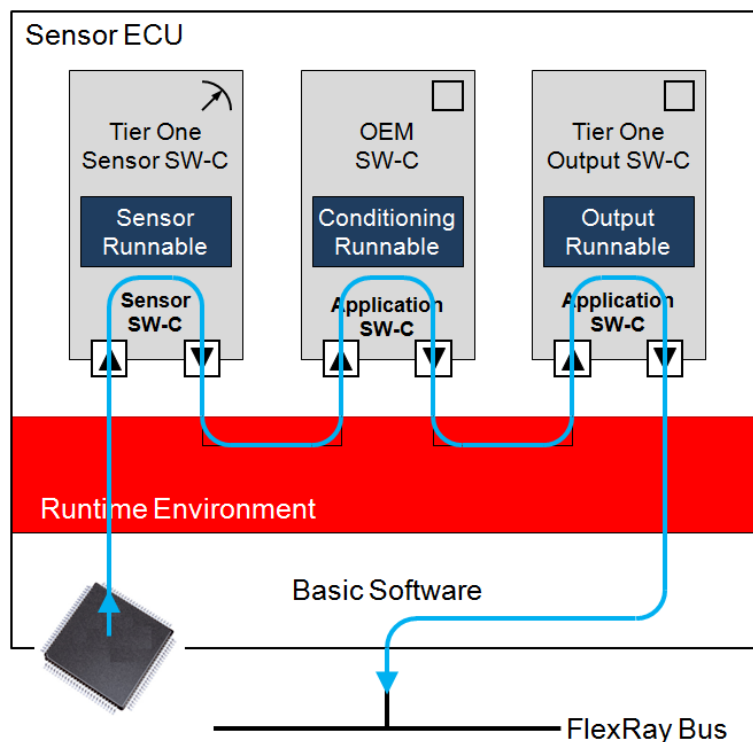


Figure 8.1: A Sensor ECU connected with a FlexRay Network and three software components

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>
  </AR-PACKAGE>

```

```

        </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>
    </SW-COMPONENT-PROTOTYPE>
    <SW-COMPONENT-PROTOTYPE>
      <SHORT-NAME>conditioningComponent</SHORT-NAME>
    </SW-COMPONENT-PROTOTYPE>
    <SW-COMPONENT-PROTOTYPE>
      <SHORT-NAME>outputComponent</SHORT-NAME>
    </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 have 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>
          <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>
            <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>
  <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>
      <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>
    <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>
  <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>
  </LATENCY-TIMING-CONSTRAINT>
</TIMING-REQUIREMENTS>
    
```

```
<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 "IgnitionActuator".

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.

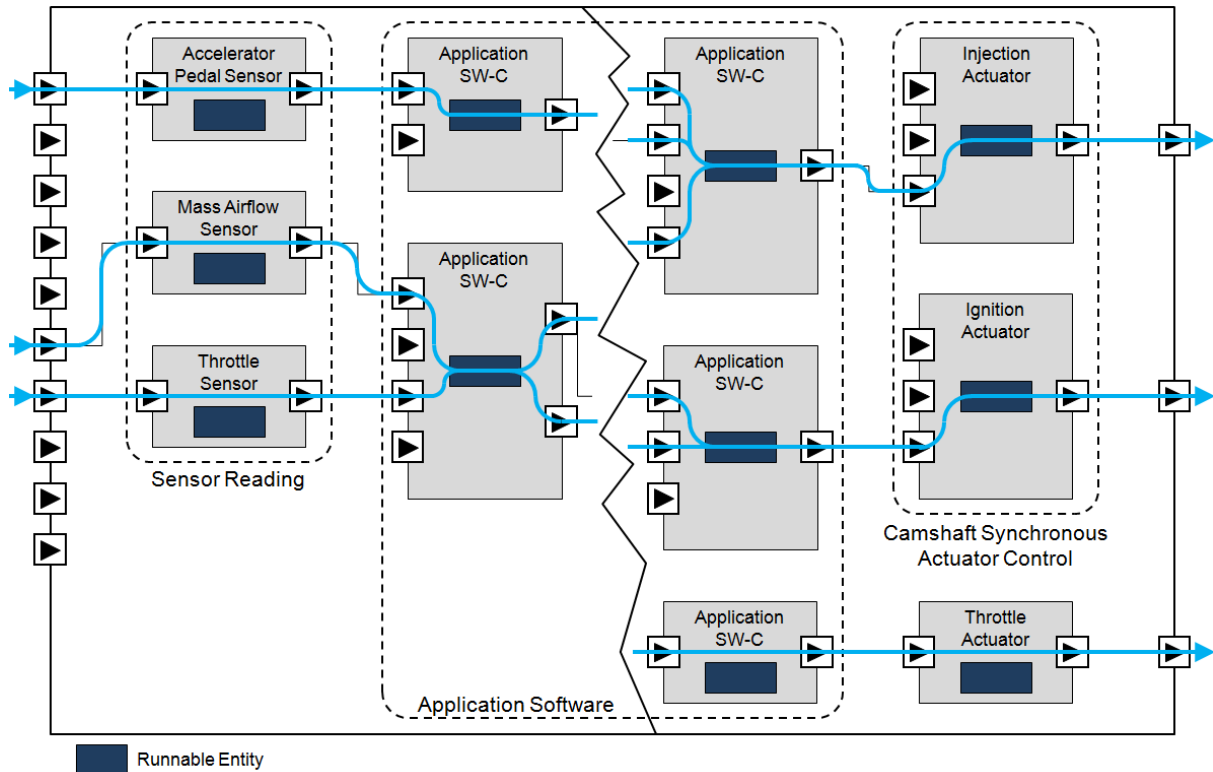


Figure 8.2: Rough sketch of an internal combustion engine control system including important signal flow paths.

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 "AcceleratorPedal" is responsible for reading in the signal and passes it to the application software component "AcceleratorPedalVoter". 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.

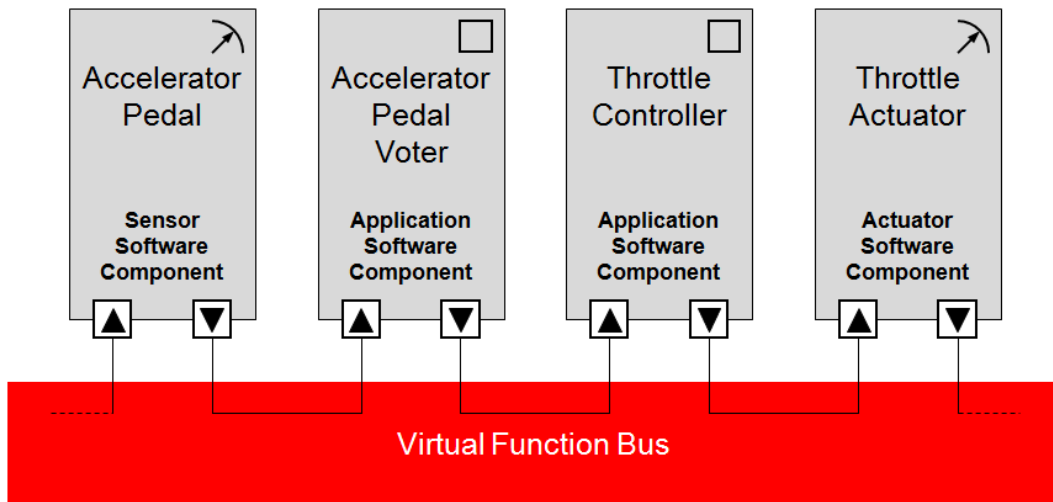


Figure 8.3: Involved components for signal flow from "AcceleratorPedal" to "ThrottleActuator" for timing requirement 1.

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.

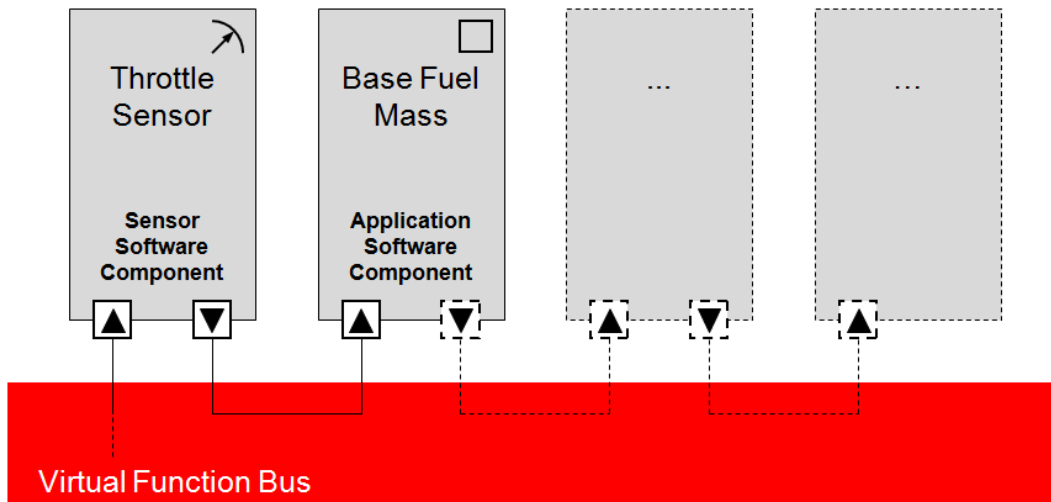


Figure 8.4: Involved components for signal flow from "ThrottleSensor" to application component "BaseFuelMass" for timing requirement 2.

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>
  </TIMING-DESCRIPTIONS>

```

```

<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>

```

```

    </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.

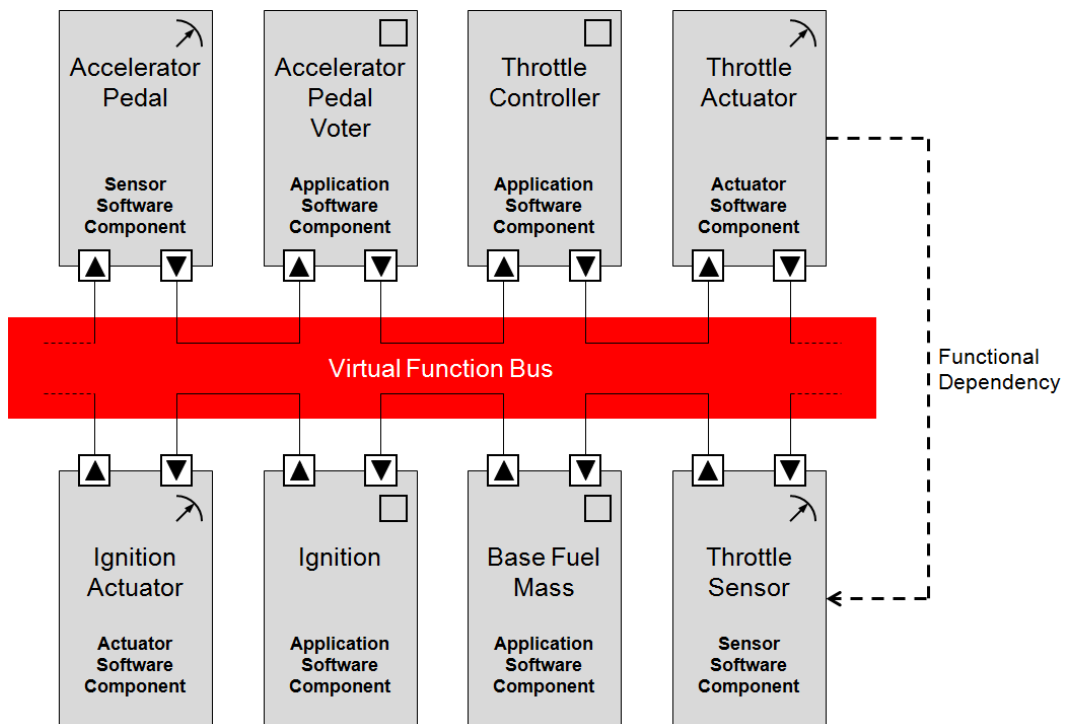


Figure 8.5: Involved components for signal flow from sensor software component "AcceleratorPedal" to actuator software component "IgnitionActuator" for timing requirement 3.

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/
    IgnitionActuator/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/RMaRateOutReceived</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/RMaRateOutReceived</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-REFS>

```

```

        <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.

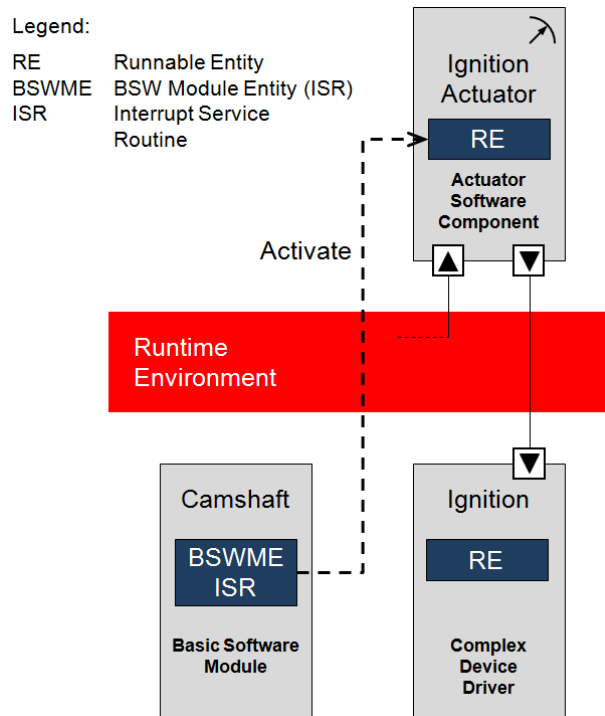


Figure 8.6: Involved components and control flow for timing requirement 4.

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/
    IgnitionActuator/IgnitionActuatorBehavior/
    IgnitionActuatorCalculation</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/
    IgnitionActuator/IgnitionActuatorBehavior/
    IgnitionActuatorCalculation</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/
    IgnitionActuator/IgnitionActuatorBehavior/
    IgnitionActuatorCalculation</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.

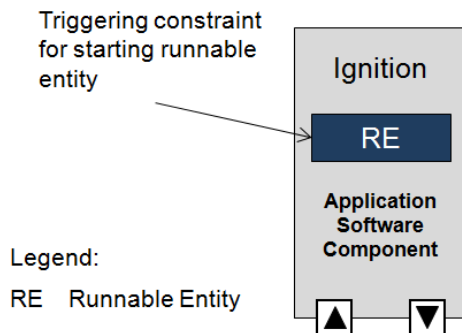


Figure 8.7: Involved component and control flow for timing requirement 5.

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/
        IgnitionBehavior/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/
        IgnitionBehavior/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".

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.

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 <code>AutosarOperationArgumentInstance</code> for Content Filter
[constr_4504]	Restricted usage of <code>AgeConstraint</code>
[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 <code>AgeConstraint</code> is no longer restricted to events of type <code>TDEventVariable-DataPrototype</code> only. All events indicating the receipt and reading of data can be referenced by the <code>AgeConstraint</code> .

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 EOCExecutableEntityRefGroup
[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 <code>directSuccessor</code> relationships
[constr_4535]	Same Mode of <code>ExecutableEntities</code>
[constr_4536]	Compatible recurrence of <code>ExecutableEntities</code>
[constr_4537]	References among elements in an <code>ExecutionOrderConstraint</code>
[constr_4538]	Hierarchical Execution Order Constraint: <code>EOCExecutableEntityRef</code> and <code>EOCExecutableEntityRef</code> shall be target or source of a successor relationship
[constr_4539]	The successor relationships <code>successor</code> and <code>directSuccessor</code> shall not be used
[constr_4540]	<code>maxCycles</code> and <code>maxSlots</code> shall not be zero
[constr_4541]	<code>EOCExecutableEntityRef</code> shall reference <code>ExecutableEntity</code> in Ordinary Execution Order Constraint
[constr_4542]	<code>EOCExecutableEntityRef</code> shall reference <code>ExecutableEntity</code> in Hierarchical Execution Order Constraint
[constr_4543]	Maximum value of the parameter <code>minimumInterArrivalTime</code>

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 Added Specification Items in R4.0.3

Number	Heading
[TPS_TIMEX_00001]	Purpose of <code>TimingDescriptionEvent</code>
[TPS_TIMEX_00002]	Purpose of <code>TimingDescriptionEventChain</code>
[TPS_TIMEX_00003]	<code>EventTriggeringConstraint</code> specifies occurrence behavior respectively model
[TPS_TIMEX_00004]	<code>LatencyTimingConstraint</code> specifies latency constraints
[TPS_TIMEX_00005]	<code>AgeConstraint</code> to specify age constraints
[TPS_TIMEX_00006]	<code>SynchronizationTimingConstraint</code> specifies synchronicity constraints
[TPS_TIMEX_00007]	<code>ExecutionOrderConstraint</code> specifies sequence of executing executable entities
[TPS_TIMEX_00008]	<code>ExecutionTimeConstraint</code> to specify execution time constraints
[TPS_TIMEX_00009]	Optional use of timing extensions
[TPS_TIMEX_00010]	<code>PeriodicEventTriggering</code> specifies periodic occurrences of events
[TPS_TIMEX_00011]	<code>SporadicEventTriggering</code> specifies sporadic occurrences of events
[TPS_TIMEX_00012]	<code>ConcretePatternEventTriggering</code> specifies concrete pattern of occurrences of events
[TPS_TIMEX_00013]	<code>BurstPatternEventTriggering</code> specifies burst of occurrences of events
[TPS_TIMEX_00014]	<code>ArbitraryEventTriggering</code> specifies arbitrary occurrences of an event
[TPS_TIMEX_00015]	<code>OffsetTimingConstraint</code> specifies offset between occurrences of events
[TPS_TIMEX_00016]	Purpose of <code>TDEventVfb</code>

[TPS_TIMEX_00017]	TDEventVariableDataPrototype specifies events observable at sender/receiver ports
[TPS_TIMEX_00018]	TDEventOperation specifies events observable at client/server ports.
[TPS_TIMEX_00019]	TDEventModeDeclaration specifies events observable 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.5: Added Specification Items in 4.0.3

A.7 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.6: Added Specification Items in 4.1.1

A.8 Added Specification Items in R4.1.2

No specification items were added 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	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	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table B.1: 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			
Attribute	Datatype	Mul.	Kind	Note
internalBehavior	SwcInternalBehavior	0..1	aggr	The SwcInternalBehaviors owned by an AtomicSwComponentType can be located in a different physical file. Therefore the aggregation is «atpSplitable». Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=internalBehavior, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the AtomicSwComponentType. Stereotypes: atpSplitable Tags: atp.Splitkey=shortName

Table B.2: AtomicSwComponentType

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, Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
interruptCategory	BswInterruptCategory	1	attr	Category of the interrupt
interruptSource	String	1	attr	Allows a textual documentation of the intended interrupt source.

Table B.3: BswInterruptEntity

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, Multilanguage Referrable, Referrable			
Attribute	Datatype	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 be raised by the referring operation.

Table B.4: 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, Multilanguage Referrable, PackageableElement, Referrable, SwComponentType			
Attribute	Datatype	Mul.	Kind	Note
hardwareElement	HwDescriptionEntity	*	ref	Reference from the ComplexDeviceDriverSwComponentType to the description of the used HwElements.

Table B.5: ComplexDeviceDriverSwComponentType

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

Table B.6: EcuAbstractionSwComponentType

Class	RunnableEntity			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
Note	<p>A RunnableEntity represents the smallest code-fragment that is provided by an AtomicSwComponentType and are executed under control of the RTE. RunnableEntities are for instance set up to respond to data reception or operation invocation on a server.</p>			
Base	ARObject,AtpClassifier,AtpFeature,AtpStructureElement,ExecutableEntity,Identifiable,MultilanguageReferrable,Referrable			
Attribute	Datatype	Mul.	Kind	Note
argument (ordered)	RunnableEntityArgument	*	aggr	This represents the formal definition of a an argument to a RunnableEntity.
asynchronousServerCallResultPoint	AsynchronousServerCallResultPoint	*	aggr	<p>The server call result point admits a runnable to fetch the result of an asynchronous server call.</p> <p>The aggregation of AsynchronousServerCallResultPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes and the variant existence of server call result points in the implementation.</p> <p>Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime</p>
canBeInvokedConcurrently	Boolean	1	attr	<p>If the value of this attribute is set to "true" the enclosing RunnableEntity can be invoked concurrently (even for one instance of the corresponding AtomicSwComponentType). This implies that it is the responsibility of the implementation of the RunnableEntity to take care of this form of concurrency. Note that the default value of this attribute is set to "false".</p>

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

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

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

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

Table B.7: RunnableEntity

Class	SensorActuatorSwComponentType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	<p>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.</p> <p>Tags: atp.recommendedPackage=SwComponentTypes</p>			
Base	ARElement,ARObject, AtomicSwComponentType ,AtpBlueprint,AtpBlueprintable,AtpClassifier,AtpType,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable, SwComponentType			
Attribute	Datatype	Mul.	Kind	Note
sensorActuator	HwDescriptionEntity	1	ref	Reference from the Sensor Actuator Software Component Type to the description of the actual hardware.

Table B.8: SensorActuatorSwComponentType

Class	SwComponentType (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for AUTOSAR software components.			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,AtpClassifier,AtpType,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
consistencyNeeds	ConsistencyNeeds	*	aggr	This represents the collection of ConsistencyNeeds owned by the enclosing SwComponentType. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
port	PortPrototype	*	aggr	The ports through which this component can communicate. The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
swComponentDocumentation	SwComponentDocumentation	0..1	aggr	This adds a documentation to the SwComponentType. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=swComponentDocumentation, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10
unitGroup	UnitGroup	*	ref	This allows for the specification of which UnitGroups are relevant in the context of referencing SwComponentType.

Table B.9: SwComponentType

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	Datatype	Mul.	Kind	Note
ecuExtractVersion	RevisionLabelString	0..1	attr	Version number of the Ecu Extract.

Attribute	Datatype	Mul.	Kind	Note
fibexElement	FibexElement	*	ref	<p>Reference to ASAM FIBEX elements specifying Communication and Topology.</p> <p>All Fibex Elements used within a System Description shall be referenced from the System Element.</p> <p>atpVariation: In order to describe a product-line, all FibexElements can be optional.</p> <p>Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild</p>
mapping	SystemMapping	*	aggr	<p>Aggregation of all mapping aspects (mapping of SW components to ECUs, mapping of data elements to signals, and mapping constraints).</p> <p>In order to support OEM / Tier 1 interaction and shared development for one common System this aggregation is atpSplittable and atpVariation. The content of SystemMapping can be provided by several parties using different names for the SystemMapping.</p> <p>This element is not required when the System description is used for a network-only use-case.</p> <p>Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=postBuild</p>
pncVectorLength	PositiveInteger	0..1	attr	<p>Length of the partial networking request release information vector (in bytes).</p>
pncVectorOffset	PositiveInteger	0..1	attr	<p>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.</p>
rootSoftwareComposition	RootSwCompositionPrototype	0..1	aggr	<p>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.</p> <p>atpVariation: The RootSwCompositionPrototype can vary.</p> <p>Stereotypes: atpVariation Tags: vh.latestBindingTime=systemDesignTime</p>

Attribute	Datatype	Mul.	Kind	Note
systemDocumentation	Chapter	*	aggr	Possibility to provide additional documentation while defining the System. The System documentation can be composed of several chapters. Stereotypes: atpSplittable; 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.10: System

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

Table B.11: VariableDataPrototype