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

Document Version	1.2.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Description
2011-09-22	1.2.0	AUTOSAR Adiminstration	• Added new timing constraint types `AgeConstraint` and `ExecutionTimeConstraint` • Added occurrence expression language for `TimingDescriptionEvents` • Improved `TDEventModeDeclaration`, `BurstPatternEventTriggering` and `SwcTiming`
2010-11-03	1.1.0	AUTOSAR Adiminstration	• Dropped `InstanceRefs` and replaced with `ComponentInCompositionInstanceRef` • Restricted the semantics of `ExecutionOrderConstraint` and `OffsetConstraint` • Parameterize the observable event ’FlexRayClusterCycleStart’ by defining the cycle repetition
2009-11-30	1.0.0	AUTOSAR Administration	Initial Release

Table of Contents

References

[1] Methodology
AUTOSAR_TR_Methodology.pdf

[2] Requirements on Timing Extensions
AUTOSAR_RS_TimingExtensions.pdf

[3] Virtual Functional Bus
AUTOSAR_EXP_VFB.pdf

[4] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate.pdf

[5] Compositional Scheduling Analysis Using Standard Event Models
http://www.ida.ing.tu-bs.de/forschung/publikationen/
Ric04_CompoSchedAnalyUsing.pdf

[6] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[7] Standardization Template
AUTOSAR_TPS_StandardizationTemplate.pdf

1 Introduction

1.1 Abbreviations

Abbreviation	*Meaning*
BSW	Basic Software
CAN	Controller Area Network
CC	Communication Controller
COM	Communication module
DTD	Document Type Definition
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
SWC	Software Component
TD	Timing Description
UML	Unified Modeling Language
VFB	Virtual Functional Bus
WCRT	Worst Case Response Time

1.2 Glossary of terms

Term	*Definition*
Jitter	For a periodically occurring timing event, the jitter is defined as the maximum variation of its period with respect to a predefined standard period.
Latency	The latency of a timing event chain describes the time duration between the occurrence of the stimulus and the occurrence of the corresponding response.
Maximum interarrival time	Describes the maximum time interval between two consecutive event occurrences. In the more general case, this attribute is an array of the maximum latency between two, three, four, ... event occurrences.
Minimum interarrival time	Describes the minimum time interval between two consecutive event occurrences. In the more general case, this attribute is an array of the minimum latency between two, three, four, ... event occurrences.

Period	Describes the expected time interval between two consecutive event occurrences, neglecting variation (jitter).
Response	End point of an event chain.
Synchronization	Synchronization focuses on the occurrence of different timing events. Synchronization of timing events means that they must occur simultaneously within a certain tolerance interval.
Stimulus	Start point of an event chain.
Timing analysis	Timing analysis is a method of determining the expected 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 describes its expected timing behavior and correlation of timing events.
Timing event	A timing event is the abstract representation of a specific system behavior – that can be observed at runtime – in the AUTOSAR specification. Timing events are used to define the scope for timing constraints. Depending on the specific scope, the view on the system, and the level of abstraction different types of events are defined.
Timing event chain	A timing event chain describes the causal order for a set of functionally dependent timing events. Each event chain has a well defined stimulus and response, which describe its start and end point. Furthermore, it can be hierarchically decomposed into an arbitrary number of sub-chains, so called ”event chain segments”.

Timing event occurrence	A timing event is said to ”occur”, when a specific system behavior – represented by the timing event – can be observed. For example the timing event ”RunnableEntityStarted” occurs, when the associated `RunnableEntity` has entered the state ”started” after its activation.
Timing guarantee	see glossary entry for ”Timing constraint”.
Timing information	Superordinate concept for timing properties and timing constraints.
Timing path	A timing path defines a sequence of communication or computation activities of the system, whose timing behavior shall be examined. Timing paths can be expressed by event chains.
Timing property	A timing property defines the state or value of a timing relevant aspect within the system (e.g. the execution time bounds for a `RunnableEntity` or the priority of a task). Thus, a property does not represent a constraint for the system, but a somehow gathered (e.g. measured, estimated or determined) or defined attribute of the system.
Timing requirement	A timing requirement defines a restriction on timing that must be fulfilled to ensure proper operation of the system. Timing requirements can be expressed by using timing constraints.
Timing validation	Timing validation compares the result of timing analysis (see glossary entry for timing analysis) with the expected behavior defined by timing constraints (see glossary entry for timing constraints).

1.3 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.

1.4 Scope

The primary scope of the AUTOSAR Timing Extensions is to provide a consolidated and consistent representation of relevant timing dependencies and according timing constraints in AUTOSAR.

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.

1.5 Document conventions

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

1.6 Requirements Traceability

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

Requirement	Description	Satisfied by
[RS_TIMEX_0001]	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_0001] [TPS_TIMEX_0002] [TPS_TIMEX_0003] [TPS_TIMEX_0004] [TPS_TIMEX_0005] [TPS_TIMEX_0006] [TPS_TIMEX_0007] [TPS_TIMEX_0008] [TPS_TIMEX_0010] [TPS_TIMEX_0011] [TPS_TIMEX_0012] [TPS_TIMEX_0013] [TPS_TIMEX_0014] [TPS_TIMEX_0015] [TPS_TIMEX_0016] [TPS_TIMEX_0017] [TPS_TIMEX_0018] [TPS_TIMEX_0019] [TPS_TIMEX_0020] [TPS_TIMEX_0021] [TPS_TIMEX_0022] [TPS_TIMEX_0023] [TPS_TIMEX_0024] [TPS_TIMEX_0025] [TPS_TIMEX_0026] [TPS_TIMEX_0027] [TPS_TIMEX_0028] [TPS_TIMEX_0029] [TPS_TIMEX_0030] [TPS_TIMEX_0031] [TPS_TIMEX_0032] [TPS_TIMEX_0033] [TPS_TIMEX_0034] [TPS_TIMEX_0035] [TPS_TIMEX_0036]
[RS_TIMEX_0002]	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_0003] [TPS_TIMEX_0004] [TPS_TIMEX_0006] [TPS_TIMEX_0007] [TPS_TIMEX_0010] [TPS_TIMEX_0011] [TPS_TIMEX_0012] [TPS_TIMEX_0013] [TPS_TIMEX_0014] [TPS_TIMEX_0015]
[RS_TIMEX_0003]	The usage of timing constraints in the AUTOSAR templates shall be optional.	[TPS_TIMEX_0009]
[RS_TIMEX_0004]	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_0002]

Requirement	Description	Satisfied by
[RS_TIMEX_0005]	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_0002]
[RS_TIMEX_0006]	The AUTOSAR templates shall provide the means to describe the triggering behavior (e.g. periodic, sporadic, and arbitrary) of event chains.	[TPS_TIMEX_0003] [TPS_TIMEX_0010] [TPS_TIMEX_0011] [TPS_TIMEX_0012] [TPS_TIMEX_0013] [TPS_TIMEX_0014]
[RS_TIMEX_0007]	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_0006]
[RS_TIMEX_0008]	The AUTOSAR templates shall provide the means to describe multiple asynchronous clocks/time bases and their interrelation.	[TPS_TIMEX_0003] [TPS_TIMEX_0006] [TPS_TIMEX_0010] [TPS_TIMEX_0011] [TPS_TIMEX_0012] [TPS_TIMEX_0013] [TPS_TIMEX_0014] [TPS_TIMEX_0015]
[RS_TIMEX_0009]	It shall be possible to annotate conncetions among SWCs on VFB, to indicate that a sender-receiver communication needs to be buffered.	[TPS_TIMEX_0002] [TPS_TIMEX_0005]
[RS_TIMEX_0010]	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_0037]
[RS_TIMEX_0011]	The AUTOSAR templates shall provide the means to describe the dependency of timing properties and constraints on operation modes defined on system and ECU level.	
[RS_TIMEX_0012]	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_0004]

2 Timing specification overview

The research field of real time systems offers a variety of timing models and specification techniques. The timing extensions described in this document serve as timing specification approach dedicated for AUTOSAR systems. Before going into detail of the solution (see 3), this chapter outlines which timing views can be applied in the different phases of the AUTOSAR methodology.

2.1 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 2.1 provides a dedicated view to the AUTOSAR methodology, focusing on the process elements which are of interest for 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 each 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.

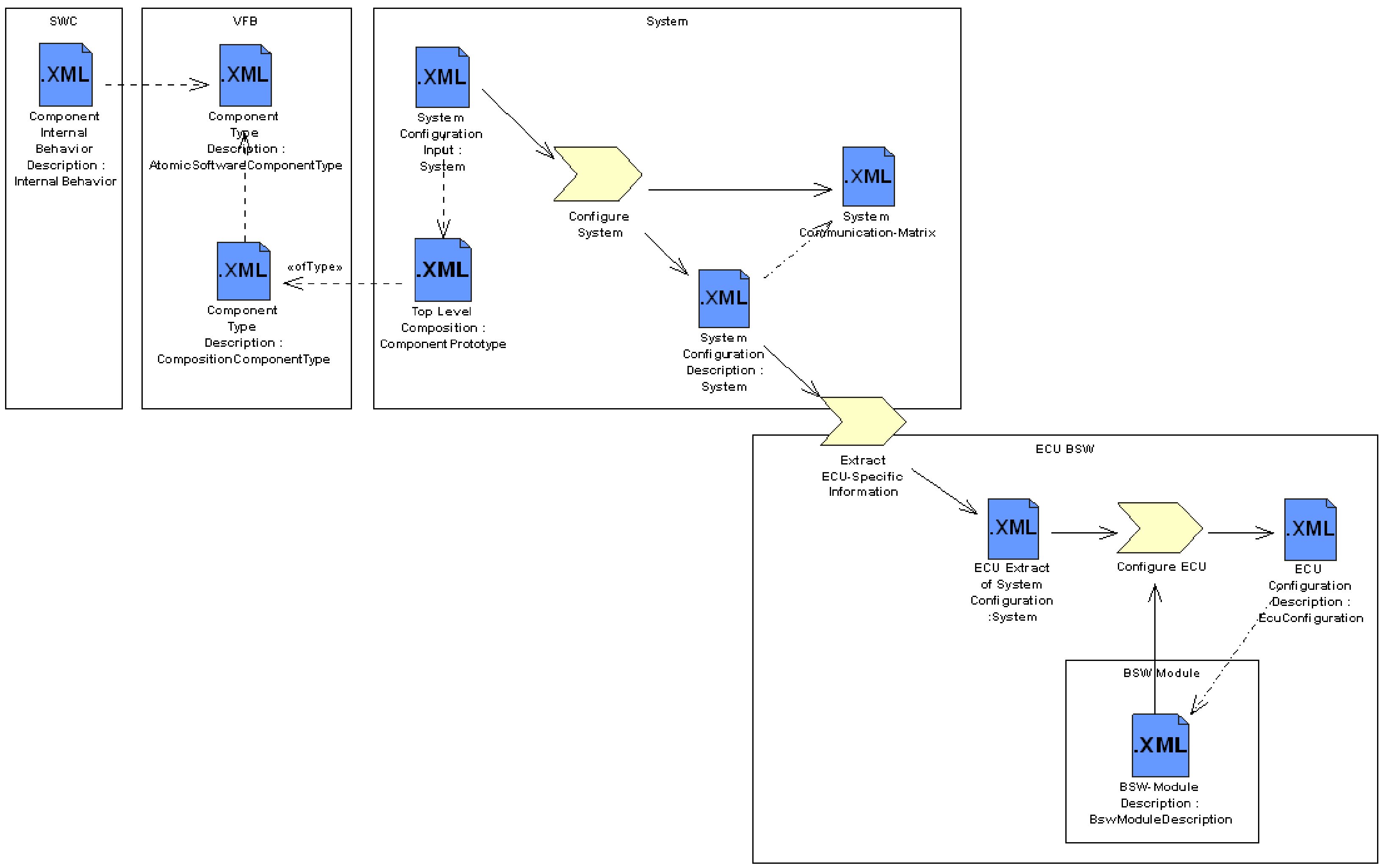

SWC
.XML
Component Internal Behavior Description : Internal Behavior
VFB
.XML
Component Type Description : AtomicSoftwareComponentType
.XML
Component Type Description : CompositionComponentType
«ofType»
System
.XML
System Configuration Input : System
.XML
Top Level Composition : Component Prototype
Configure System
.XML
System Communication-Matrix
.XML
System Configuration Description : System
Extract ECU-Specific Information
ECU BSW
.XML
ECU Extract of System Configuration :System
Configure ECU
.XML
ECU Configuration Description : EcuConfiguration
BSW Module
.XML
BSW-Module Description : BswModuleDescription

Figure 2.1: Overview of the AUTOSAR methodology and timing specification

2.2 VfbTiming

AUTOSAR defines the `Virtual Functional Bus` [3] as a composition of `SwComponentTypes` 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, real end-to-end timing constraints (including sometimes the 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 `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`.

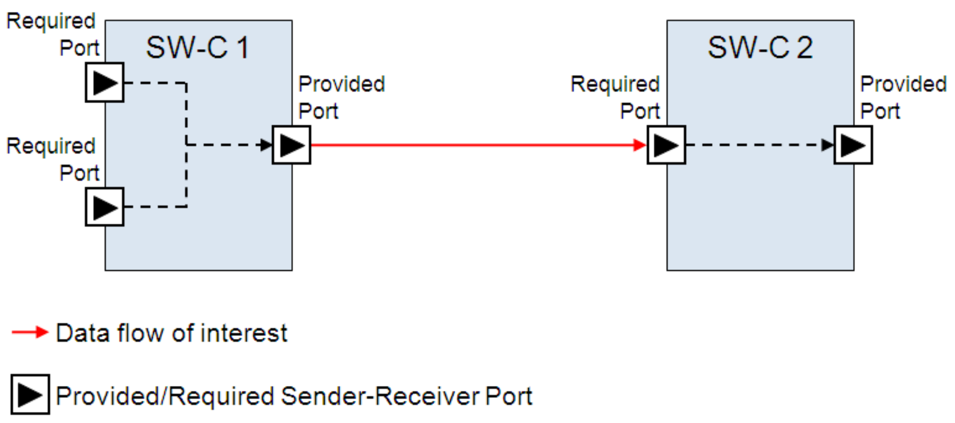


Figure 2.2: Example: data flow in the scope of the VfbTiming view

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`.

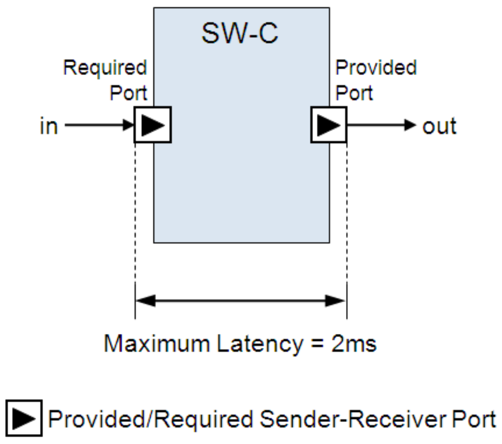


Figure 2.3: Example: latency requirement

As an example, consider the timing constraint illustrated in figure 2.3: ”From the point in time, where the value *in* is received by *SWC*, 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` *SWC* (see figure 2.1).

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

2.3 SwcTiming

In contrast to the `VfbTiming` view, a specification engineer might especially be interested in the `SwcInternalBehavior` of `AtomicSwComponentTypes` that are represented as a 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 2.1). It can refer to the activation, start, and termination (see section 3.2.2) of the execution of `RunnableEntities`.

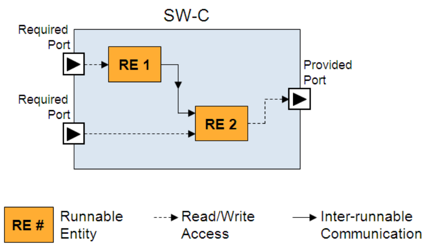


Figure 2.4: Example: data flow in the scope of the SwcTiming view

2.4 SystemTiming

At system level a special prototype of a CompositionSwComponentType – the RootSWComponentPrototype – is instantiated. This prototype, the chosen hardware topology and other artifacts are used as System Configuration Input to configure the system. The main configuration result is the mapping of software components to ECUs and the resulting communication matrix. This information is aggregated in the System Configuration Description document (see figure 2.1).

The SystemTiming view is used to provide timing descriptions at system level. As an extension, it can be attached to the System Configuration Description. As the System Configuration Description aggregates all the information from the Component Type Description and the Component Internal Behavior Description, it is possible to use the same concepts that are available in the views VfbTiming and SwcTiming also here. The difference exists in the special system context that defines the validity of a timing description at system level. Without knowledge of the mapping of components to a concrete target hardware, only a generic platform independent description can be provided.

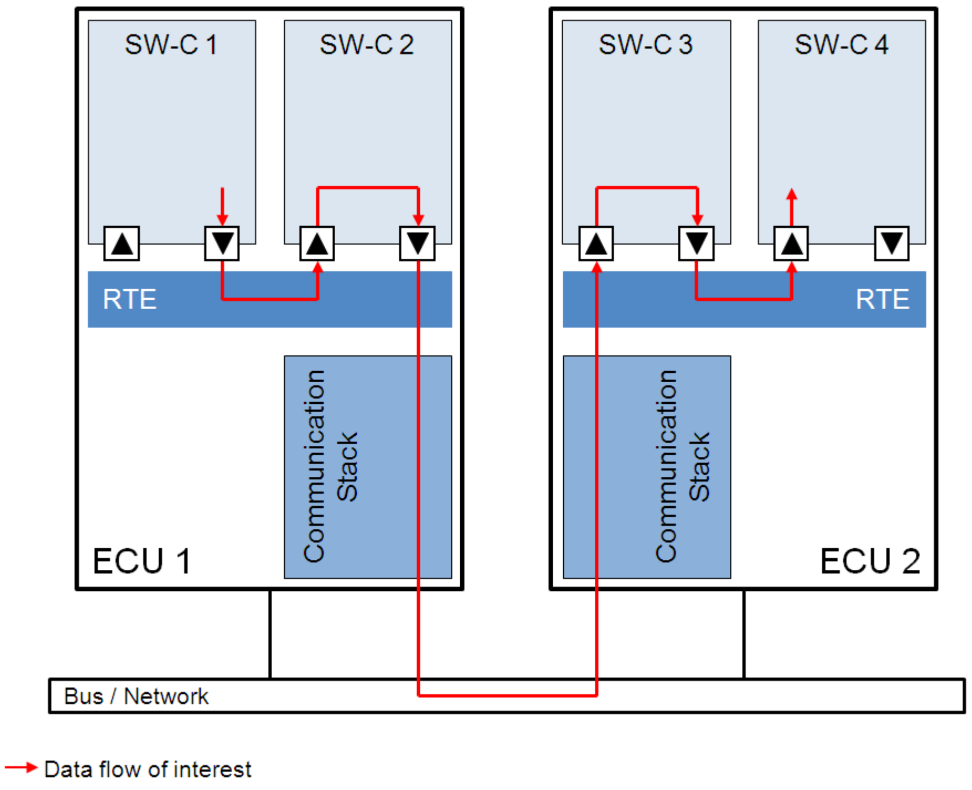


Figure 2.5: An example of data flow, whose timing behavior is in scope of system view

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).

2.5 BswModuleTiming

According to figure 2.1, a `BSW Module Description` is generated for each BSW module as part of the ECU configuration phase. For each module the internal module constitution (the so called `BswInternalBehavior`) must be defined, i.e. structuring `BswModuleEntities`. Similar to the timing view on the `SwcInternalBehavior` of a `AtomicSwComponentType`, BSW module timing 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`.

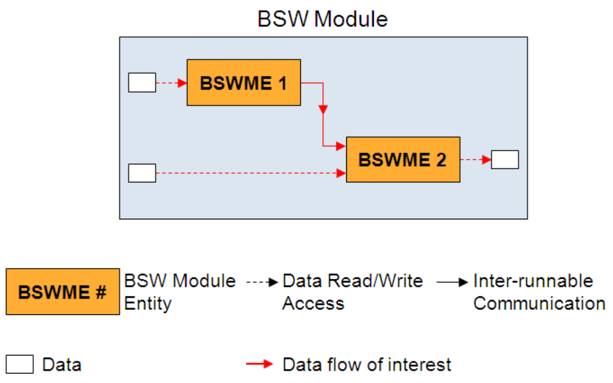


Figure 2.6: An example of data flow, whose timing behavior is in scope of BSW module view

2.6 EcuTiming

A result of the ECU configuration phase is the complete `ECU Configuration Description` (see figure 2.1). During 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.

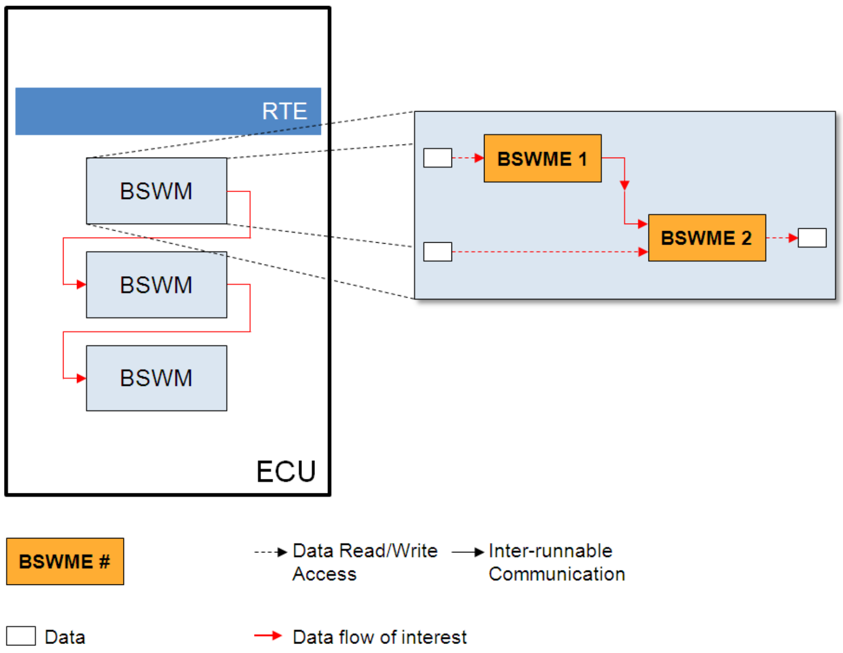


Figure 2.7: An example of data flow, whose timing behavior is in scope of ECU view

In this view, timing can reference all the ECU-relevant information: The deployed software component instances, the ECU related interactions including bus communication, etc. In other words, the `ECUTiming` has the same expressivity as the system timing with the restriction of focusing on one specific ECU. In addition, the complete BSW can be considered in a timely manner, since the complete composition of the BSW modules is known. In this case, isolated BSW modules (as the "BSW module timing" does) as well as inter BSW module relations (e.g. from the BSW module M1 to the BSW module M2) are of interest. The information is attached to the `ECU Configuration Description`.

3 Timing extensions

This chapter describes the framework of timing extensions that provides the means to define timing descriptions and constraints in AUTOSAR, and enable the analysis and validation of timing behavior. Depending on the concrete view (see chapter 2) timing extensions can be used at VFB, System, and ECU level to describe the timing behavior of an AUTOSAR system.

3.1 Fundamentals

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

3.1.1 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. It 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 2), and in order to avoid semantic confusion with existing concepts, a new abstract type `TimingDescriptionEvent` (see section 3.2) is introduced as a formal basis for the timing extensions. Depending on the concrete 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 interest. To relate timing events to each other, a further concept called `TimingDescriptionEventChain` (see section 3.3) 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 2.5 in chapter 2. The 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_0009] Optional use of timing extensions ⌈ The elements `TimingExtension`, `TimingDescription`, and `TimingConstraint` of the timing extensions are derived from the element `ARElement`. This enables one to deliver timing extensions in a separate document. In addition, there are no external references from any template that point to timing extensions elements. ⌋ *(RS_TIMEX_0003)*

3.1.2 Application in different views

The timing specification concept can be applied to the previously described views in a very generic and intuitive way. Depending on the concrete view, different types of timing events and thus event chains, as well as different timing constraints are available to specify the expected timing behavior.

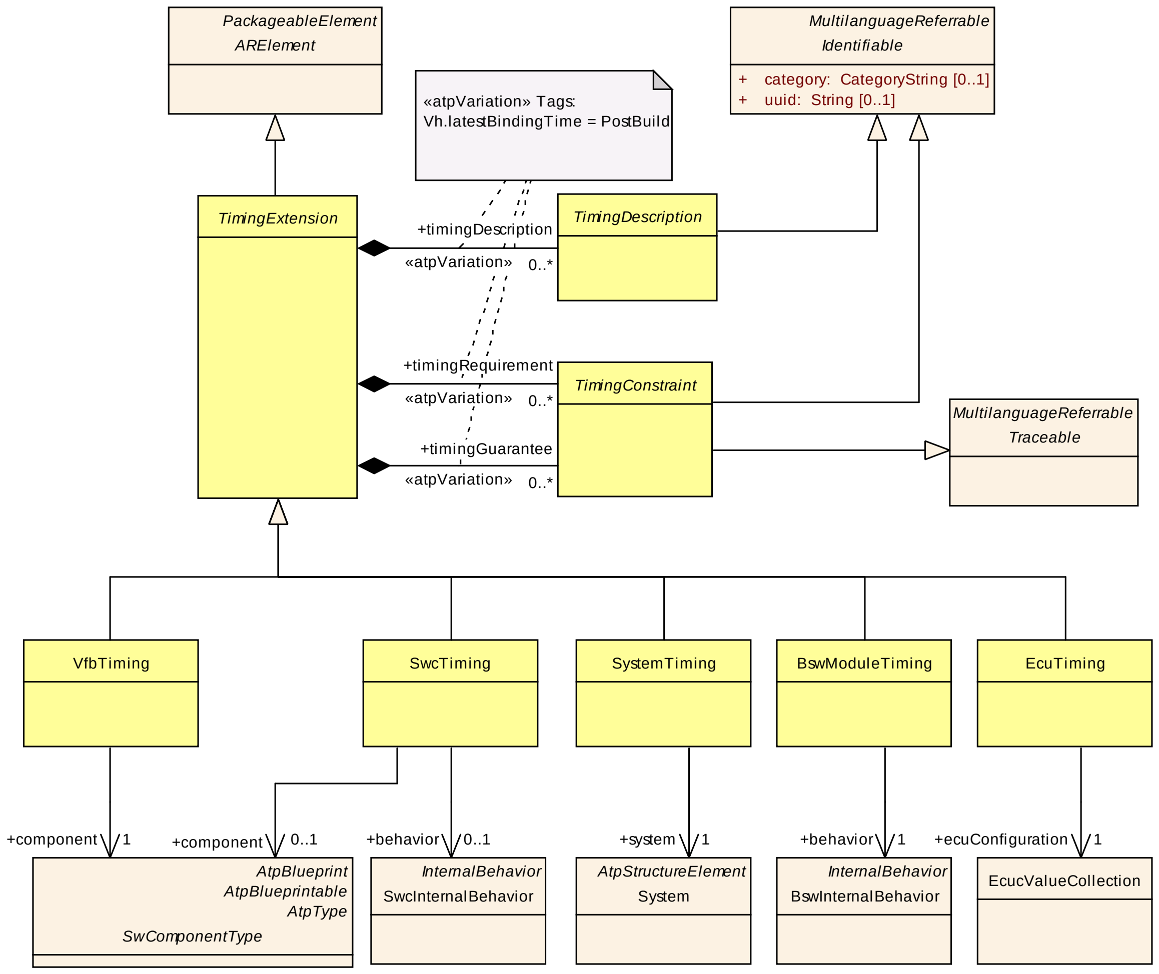


Figure 3.1: Overview of the timing extensions

The association of the described concepts with the different timing views is depicted in figure 3.1.

Class	**TimingExtension (abstract)**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing			
Note	The abstract parent class of the different template specific timing extensions. Depending on the specific timing extension (VfbTiming, SwcTiming, SystemTiming, BswModuleTiming, EcuTiming) the timing descriptions and timing constraints, that can be used to specify the timing behavior, are restricted.			
Base	ARElement,ARObject,CollectableElement,Identifiable,Multilanguage Referrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
timingDescription	TimingDescription	*	aggr	The timing descriptions that belong to a specific timing specification. In order to support different timing description variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=PostBuild
timingGuarantee	TimingConstraint	*	aggr	The timing constraints that belong to a specific timing specification in the role of a timing guarantee. In order to support different timing constraint variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=PostBuild
timingRequirement	TimingConstraint	*	aggr	The timing constraints that belong to a specific timing specification in the role of a timing requirement. In order to support different timing constraint variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=PostBuild

Table 3.1: 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 3.2: 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 3.3: TimingConstraint

Each view is represented by a specific timing extension (`VfbTiming`, `SwcTiming`, `SystemTiming`, `BswModuleTiming`, `EcuTiming`), that is associated with an AUTOSAR model entity which defines the outer scope of the timing descriptions (events and event chains) and the timing constraints for that extension.

[TPS_TIMEX_0032] Purpose of `VfbTiming` ⌈ The element `VfbTiming` aggregates all timing information, timing descriptions and timing constraints, that is related to the VFB View. ⌋*(RS_TIMEX_0001)*

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

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

Table 3.4: VfbTiming

[TPS_TIMEX_0033] 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_0001)*

Class	**SwcTiming**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing			
Note	The SwcTiming is used to describe the timing of a software component. A software component can either be of type AtomicSwComponentType or CompositionSwComponentType. In the former case, the SwcTiming allows to describe timing description and constraints for the SwcInternalBehavior of the AtomicSwComponentType. In the latter case, timing descriptions and constraints can be defined for all AtomicSwComponentType's within the CompositionSwComponentType. Unlike the VfbTiming, TimingDescriptions aggregated by SwcTiming are restricted to event chains referring to events which are derived from the classes TDEventVfb and TDEventInternalBehavior. **Tags:** atp.recommendedPackage=TimingExtensions			
Base	ARElement,ARObject,CollectableElement,Identifiable,Multilanguage Referrable,PackageableElement,Referrable,TimingExtension			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
behavior	SwcInternalBeh avior	0..1	ref	This defines the scope of a SwcTiming. All corresponding timing descriptions and constraints must be defined within this scope. Note! The reason for the cardinality of 0..1 is to ensure backward compatibility.
component	SwComponentT ype	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 3.5: SwcTiming

[TPS_TIMEX_0034] Purpose of `SystemTiming` ⌈ The element `SystemTiming` aggregates all timing information, timing descriptions and timing constraints, that is related to the System View. ⌋*(RS_TIMEX_0001)*

Class	**SystemTiming**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing			
Note	A model element used to refine timing descriptions and constraints (from a VfbTiming) at System level, utilizing information about topology, software deployment, and signal mapping described in the System Template. TimingDescriptions aggregated by SystemTiming are restricted to events which are derived from the class TDEventVfb, TDEventSwcInternalBehavior and TDEventCom. **Tags:** atp.recommendedPackage=TimingExtensions			
Base	ARElement,ARObject,CollectableElement,Identifiable,Multilanguage Referrable,PackageableElement,Referrable,TimingExtension			
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 3.6: SystemTiming

[TPS_TIMEX_0035] Purpose of `BswModuleTiming` ⌈ The element `BswModule-Timing` aggregates all timing information, timing descriptions and timing constraints, that is related to the Basic Software Module View. ⌋*(RS_TIMEX_0001)*

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

Table 3.7: BswModuleTiming

[TPS_TIMEX_0036] Purpose of `EcuTiming` ⌈ The element `EcuTiming` aggregates all timing information, timing descriptions and timing constraints, that is related to the ECU View. ⌋*(RS_TIMEX_0001)*

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

Table 3.8: EcuTiming

Details of the different event types and the limitations to their usage are described in section 3.2.

3.2 TimingDescriptionEvent

[TPS_TIMEX_0001] 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_0001)*

For example, this can be the start of a `RunnableEntity` or the hand-over of a frame to the hardware buffer of a communication controller.

An overview of the different event types is given in figure 3.2. These are described in more detail in the following.

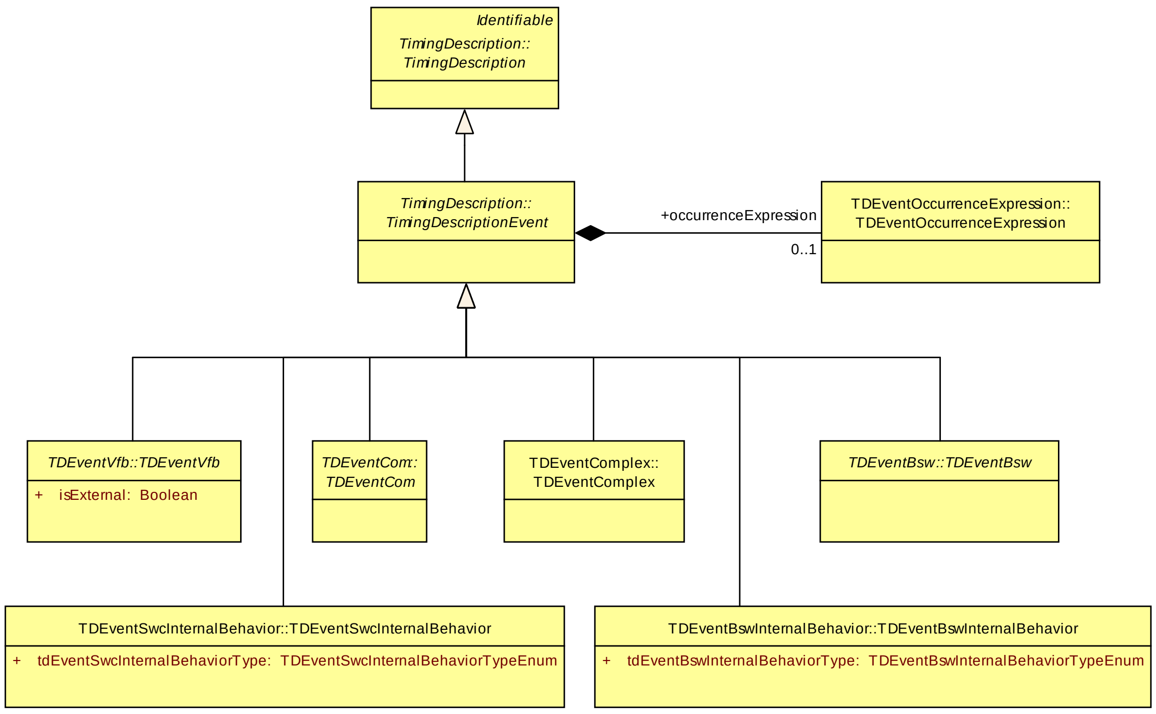

Identifiable
TimingDescription::
TimingDescription
TimingDescription::
TimingDescriptionEvent
+occurrenceExpression
0..1
TDEventOccurrenceExpression::
TDEventOccurrenceExpression
TDEventVfb::TDEventVfb
+ isExternal: Boolean
TDEventCom::
TDEventCom
TDEventComplex::
TDEventComplex
TDEventBsw::TDEventBsw
TDEventSwcInternalBehavior::TDEventSwcInternalBehavior
+ tdEventSwcInternalBehaviorType: TDEventSwcInternalBehaviorTypeEnum
TDEventBswInternalBehavior::TDEventBswInternalBehavior
+ tdEventBswInternalBehaviorType: TDEventBswInternalBehaviorTypeEnum

Figure 3.2: Overview of the different types of timing events

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

Table 3.9: TimingDescriptionEvent

Also note that information regarding the occurrence of a `TimingDescriptionEvent` is described separately in section 3.4.

3.2.1 Timing events related to the VFB

[TPS_TIMEX_0016] 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_0001)*

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

- `VfbTiming` 2.2
- `SwcTiming` 2.3
- `SystemTiming` 2.4
- `EcuTiming` 2.6

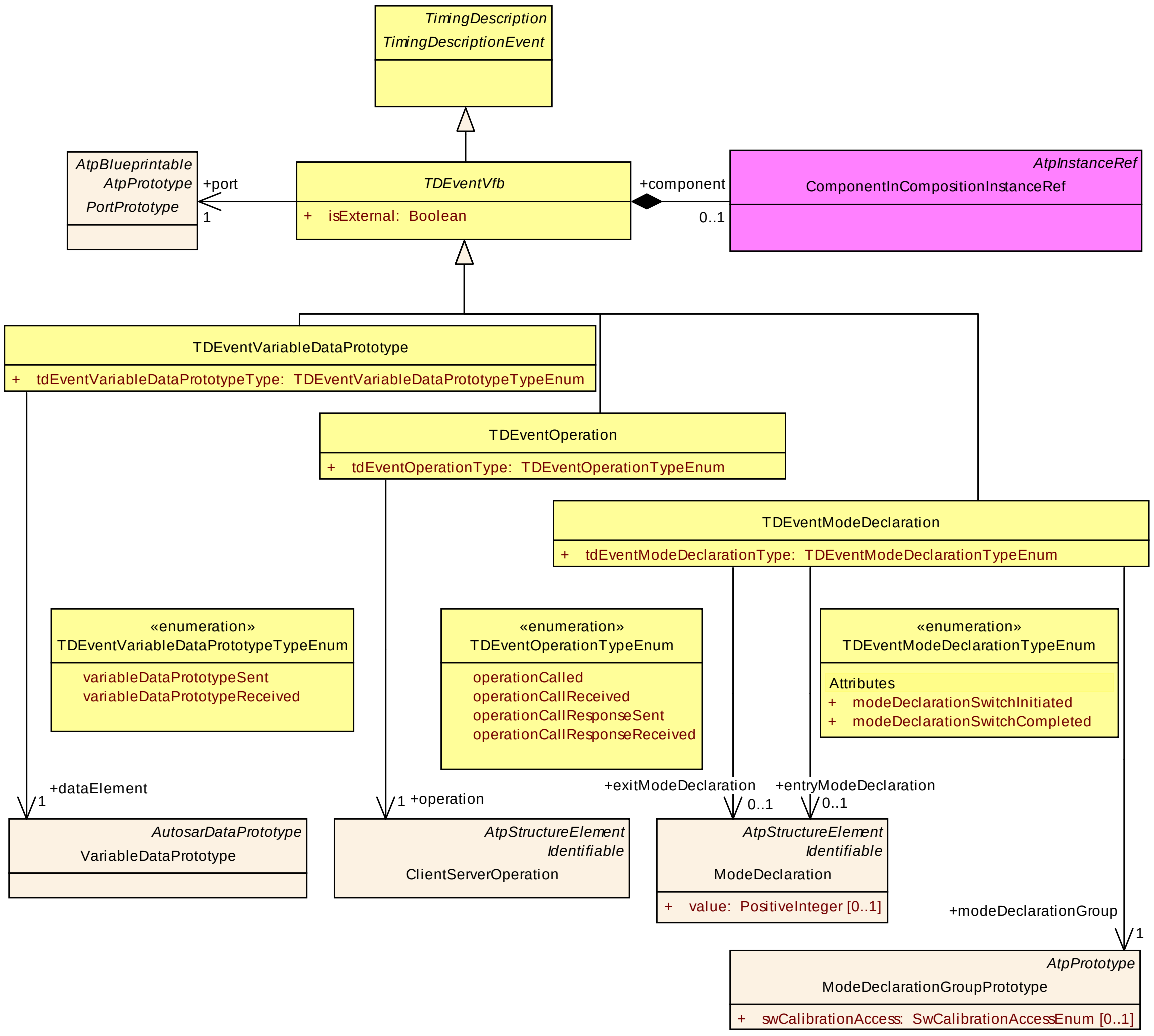

TimingDescription
TimingDescriptionEvent
AtpBlueprintable
AtpPrototype
PortPrototype
+port
1
TDEventVfb
+ isExternal: Boolean
+component
0..1
AtpInstanceRef
ComponentInCompositionInstanceRef
TDEventVariableDataPrototype
+ tdEventVariableDataPrototypeType: TDEventVariableDataPrototypeTypeEnum
TDEventOperation
+ tdEventOperationType: TDEventOperationTypeEnum
TDEventModeDeclaration
+ tdEventModeDeclarationType: TDEventModeDeclarationTypeEnum
«enumeration»
TDEventVariableDataPrototypeTypeEnum
variableDataPrototypeSent
variableDataPrototypeReceived
«enumeration»
TDEventOperationTypeEnum
operationCalled
operationCallReceived
operationCallResponseSent
operationCallResponseReceived
«enumeration»
TDEventModeDeclarationTypeEnum
Attributes
+ modeDeclarationSwitchInitiated
+ modeDeclarationSwitchCompleted
+dataElement
1
+operation
1
+exitModeDeclaration
0..1
+entryModeDeclaration
0..1
AutosarDataPrototype
VariableDataPrototype
AtpStructureElement
Identifiable
ClientServerOperation
AtpStructureElement
Identifiable
ModeDeclaration
+ value: PositiveInteger [0..1]
+modeDeclarationGroup
1
AtpPrototype
ModeDeclarationGroupPrototype
+ swCalibrationAccess: SwCalibrationAccessEnum [0..1]

Figure 3.3: 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 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.

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 (e.g. physical sensors / actuators) already at VFB level. If set to TRUE this timing event refers to the point in time, where the associated data is generated/processed by hardware I/O. Furthermore, in this case the associated variable data prototype respectively operation must belong to a component of type SensorActuatorSwComponentType or ComplexDeviceDriverSwComponentType.
port	PortPrototype	1	ref	The port scope of the timing event

Table 3.10: TDEventVfb

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_0017] `TDEventVariableDataPrototype` specifies events observable at sender/receiver ports ⌈ The element `TDEventVariableDataPrototype` is used to specify events, namely the receipt and sending of veriable data prototypes, observable at required and provided sender/receiver ports. ⌋*(RS_TIMEX_0001)*

Class	**TDEventVariableDataPrototype**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventVfb			
Note	This is used to describe timing events related to sender-receiver communication at VFB level.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TDEventVfb,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 3.11: TDEventVariableDataPrototype

Enumeration	**TDEventVariableDataPrototypeTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::Timing DescriptionEvents::TDEventVfb
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 3.12: TDEventVariableDataPrototypeTypeEnum

[TPS_TIMEX_0018] `TDEventOperation` specifies events obeservable at client/server ports. ⌈The element `TDEventOperation` is used to specify events, namely the invokation of operations and their completion, observable at required and provided client/server ports. ⌋*(RS_TIMEX_0001)*

Class	**TDEventOperation**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::Timing DescriptionEvents::TDEventVfb			
Note	This is used to describe timing events related to client-server communication at VFB level.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TDEventVfb,Timing Description,TimingDescriptionEvent			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
operation	ClientServerOp eration	1	ref	The referenced operation.
tdEventOp erationTyp e	TDEventOperati onTypeEnum	1	attr	The specific type of this timing event.

Table 3.13: TDEventOperation

Enumeration	**TDEventOperationTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::Timing DescriptionEvents::TDEventVfb
Note	This is used to describe the specific event type of a TDEventOperation.
Literal	***Description***
operationCall Received	A point in time where the call of the referenced operation is received by the server SWC.
operationCall Response Received	A point in time where the client SWC has received the response of the referenced operation call.
operationCall Response Sent	A point in time where the server SWC has terminated with the execution of the referenced operation, and has sent out a response.

operation Called	A point in time where the referenced operation is called by the client SWC.

Table 3.14: TDEventOperationTypeEnum

[TPS_TIMEX_0019] `TDEventModeDeclaration` specifies events obeservable at mode ports. ⌈ The element `TDEventModeDeclaration` is used to specify events, namely initiation and propagation of mode changes, observable at required and provided mode ports. ⌋*(RS_TIMEX_0001)*

Class	**TDEventModeDeclaration**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventVfb			
Note	This is used to describe timing events related to mode switch communication at VFB level.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TDEventVfb,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.
modeDeclarationGroup	ModeDeclarationGroupPrototype	1	ref	The referenced mode declaration group prototype.
tdEventModeDeclarationType	TDEventModeDeclarationTypeEnum	1	attr	The specific type of this timing event.

Table 3.15: TDEventModeDeclaration

Enumeration	**TDEventModeDeclarationTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventVfb
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 3.16: TDEventModeDeclarationTypeEnum

3.2.2 Timing events related to SwcInternalBehavior

[TPS_TIMEX_0020] `TDEventSwcInternalBehavior` specifies observable events of runnable entities ⌈ The element `TDEventSwcInternalBehavior` is used to specify events, namely the activation, start and termination of runnable entities, which are observable in the Software Component view. ⌋*(RS_TIMEX_0001)*

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

- `SwcTiming` 2.3
- `SystemTiming` 2.4
- `EcuTiming` 2.6

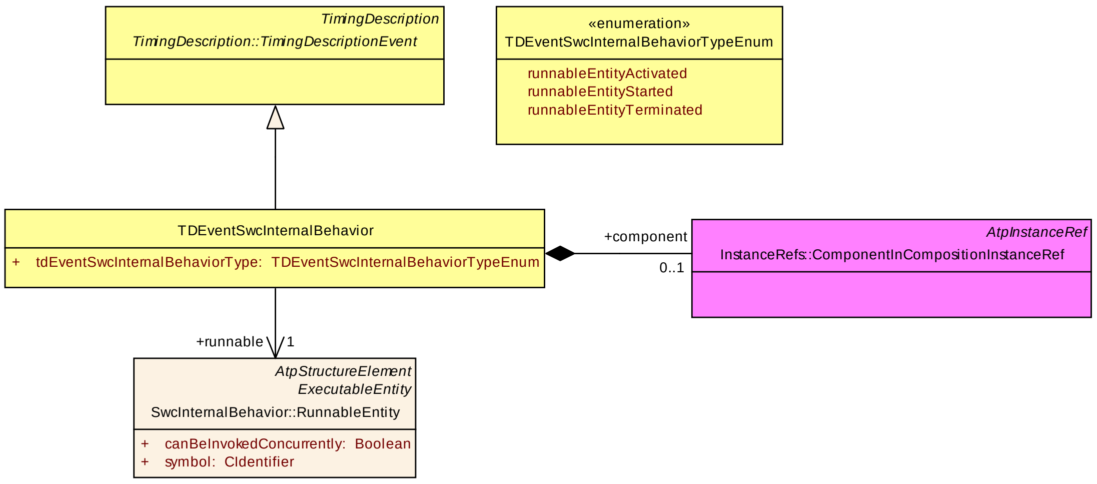

TimingDescription
TimingDescription::TimingDescriptionEvent
«enumeration»
TDEventSwcInternalBehaviorTypeEnum
runnableEntityActivated
runnableEntityStarted
runnableEntityTerminated
TDEventSwcInternalBehavior
+ tdEventSwcInternalBehaviorType: TDEventSwcInternalBehaviorTypeEnum
+component
0..1
AtpInstanceRef
InstanceRefs::ComponentInCompositionInstanceRef
+runnable
1
AtpStructureElement
ExecutableEntity
SwcInternalBehavior::RunnableEntity
+ canBeInvokedConcurrently: Boolean
+ symbol: CIdentifier

Figure 3.4: 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,TimingDescription,TimingDescriptionEvent			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
component	SwComponentPrototype	0..1	iref	The context for the scope of this timing event.
runnable	RunnableEntity	1	ref	The scope of this timing event.
tdEventSwcInternalBehaviorType	TDEventSwcInternalBehaviorTypeEnum	1	attr	The specific type of this timing event.

Table 3.17: 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".
runnable EntityStarted	A point in time where the associated RunnableEntity has entered the state "started" after its activation.
runnable EntityTerminated	A point in time where the associated RunnableEntity has terminated and entered the state "suspended".

Table 3.18: TDEventSwcInternalBehaviorTypeEnum

3.2.3 Timing events related to bus communication

[TPS_TIMEX_0021] 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_0001)*

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

- `SystemTiming` 2.3
- `EcuTiming` 2.6

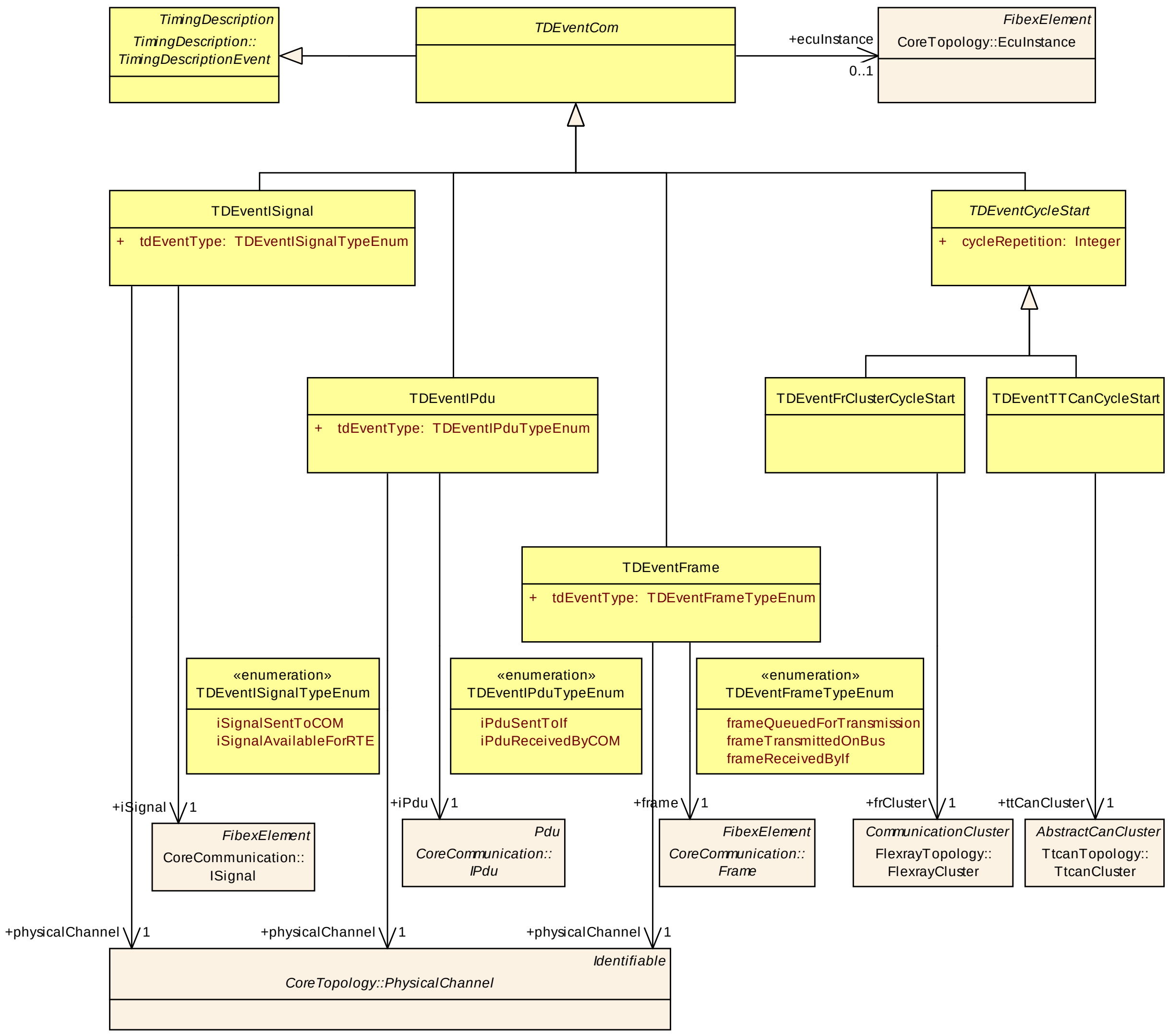

TimingDescription
TimingDescription::TimingDescriptionEvent
TDEventCom
+ecuInstance
0..1
FibexElement
CoreTopology::EcuInstance
TDEventISignal
+ tdEventType: TDEventISignalTypeEnum
TDEventCycleStart
+ cycleRepetition: Integer
TDEventIPdu
+ tdEventType: TDEventIPduTypeEnum
TDEventFrClusterCycleStart
TDEventTTCanCycleStart
TDEventFrame
+ tdEventType: TDEventFrameTypeEnum
«enumeration» TDEventISignalTypeEnum
iSignalSentToCOM
iSignalAvailableForRTE
«enumeration» TDEventIPduTypeEnum
iPduSentToIf
iPduReceivedByCOM
«enumeration» TDEventFrameTypeEnum
frameQueuedForTransmission
frameTransmittedOnBus
frameReceivedByIf
+iSignal 1
+iPdu 1
+frame 1
+frCluster 1
+ttCanCluster 1
FibexElement
CoreCommunication::ISignal
Pdu
CoreCommunication::IPdu
FibexElement
CoreCommunication::Frame
CommunicationCluster
FlexrayTopology::FlexrayCluster
AbstractCanCluster
TtcanTopology::TtcanCluster
+physicalChannel 1
+physicalChannel 1
+physicalChannel 1
Identifiable
CoreTopology::PhysicalChannel

Figure 3.5: 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***
ecuInstance	EcuInstance	0..1	ref	The ECU context for a particular timing event. The link is optional, because the EcuInstance can not be defined for events of type TDEventCycleStart.

Table 3.19: TDEventCom

[TPS_TIMEX_0022] `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_0001)*

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 3.20: 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***
iSignalAvail-ableForRTE	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 3.21: TDEventISignalTypeEnum

[TPS_TIMEX_0023] `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_0001)*

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***
tdEventType	TDEventIPduTypeEnum	1	attr	The specific type of this timing event.

Table 3.22: TDEventIPdu

Enumeration	**TDEventIPduTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventCom
Note	This is used to describe the specific event type of a TDEventIPdu.
Literal	***Description***
iPduReceivedByCOM	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.
iPduSentToIf	A point in time where the carrier COM I-PDU is routed through the PDUR and is pushed to the bus specific (FlexRay / CAN / LIN) Interface BSW module.

Table 3.23: TDEventIPduTypeEnum

[TPS_TIMEX_0024] `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_0001)*

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

Table 3.24: TDEventFrame

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

frameQueued ForTransmission	A point in time where the frame containing the named signal / I-PDU is queued for transmission within the related Communication Driver.
frameReceivedBy If	A point in time where the frame is pushed from the subscriber's communication controller to the corresponding (FlexRay / CAN / LIN) Interface BSW module.
frameTransmittedOn Bus	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 3.25: TDEventFrameTypeEnum

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

Table 3.26: TDEventCycleStart

[TPS_TIMEX_0025] `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_0001)*

Class	**TDEventFrClusterCycleStart**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::Timing DescriptionEvents::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,TDEvent CycleStart,TimingDescription,TimingDescriptionEvent			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
frCluster	FlexrayCluster	1	ref	The scope of this timing event.

Table 3.27: TDEventFrClusterCycleStart

[TPS_TIMEX_0026] `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_0001)*

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 3.28: TDEventTTCanCycleStart

3.2.4 Timing events related to the BSW

[TPS_TIMEX_0028] `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_0001)*

Three different kinds of observable events are conceivable for BSW:

- The behavior of a single BSW module: one specific `BswInternalBehavior` can be considered, focusing the dynamics of the associated `BSWModuleEntities`.
- The interaction of different BSW modules: relations between several BSW modules can be considered.

  The modules are treated as black boxes, where the interfaces between BSW modules are of interest for the `EcuTiming` specification. The information is attached to the `EcucValueCollection`.
- The mode communication on BSW level: mode requests and mode switches can be considered for BSW modules.

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

- `BswModuleTiming` 2.5
- `EcuTiming` 2.6

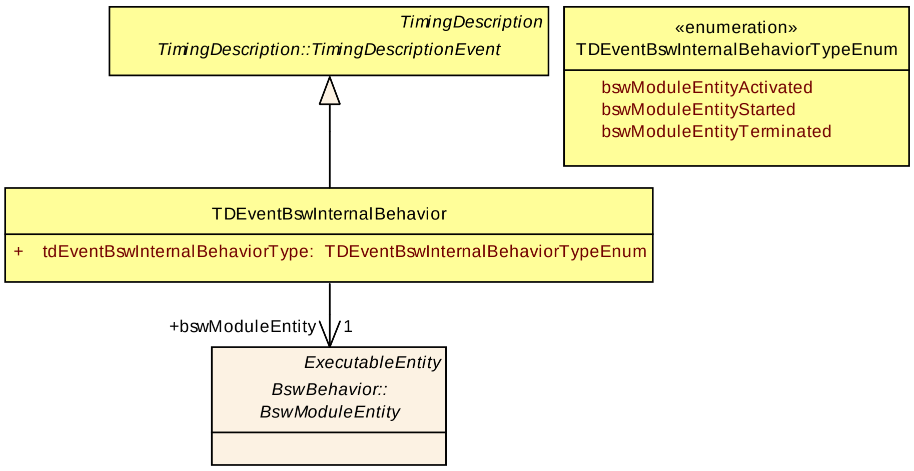

TimingDescription
TimingDescription::TimingDescriptionEvent
«enumeration»
TDEventBswInternalBehaviorTypeEnum
bswModuleEntityActivated
bswModuleEntityStarted
bswModuleEntityTerminated
TDEventBswInternalBehavior
+ tdEventBswInternalBehaviorType: TDEventBswInternalBehaviorTypeEnum
+bswModuleEntity
1
ExecutableEntity
BswBehavior::
BswModuleEntity

Figure 3.6: Events related to the internal structure of a BSW module

Class	**TDEventBswInternalBehavior**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::Timing DescriptionEvents::TDEventBswInternalBehavior			
Note	This is used to describe timing events related to the BswInternalBehavior of a BSW module.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TimingDescription,Timing DescriptionEvent			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
bswModul eEntity	BswModuleEntit y	1	ref	The scope of this timing event.
tdEventBs wInternalB ehaviorTyp e	TDEventBswInt ernalBehaviorTy peEnum	1	attr	The specific type of this timing event.

Table 3.29: 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::Timing DescriptionEvents::TDEventBswInternalBehavior
Note	This is used to describe the specific event type of a TDEventBswInternalBehavior.
Literal	***Description***
bswMod-uleEntity Activated	A point in time where the associated BswModuleEntity has been activated, which means that it has entered the state "to be started".
bswModule EntityStarted	A point in time where the associated BswModuleEntity has entered the state "started" after its activation.

bswModuleEntity Terminated	A point in time where the associated BswModuleEntity has terminated and entered the state "suspended"

Table 3.30: TDEventBswInternalBehaviorTypeEnum

[TPS_TIMEX_0029] 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_0001)*

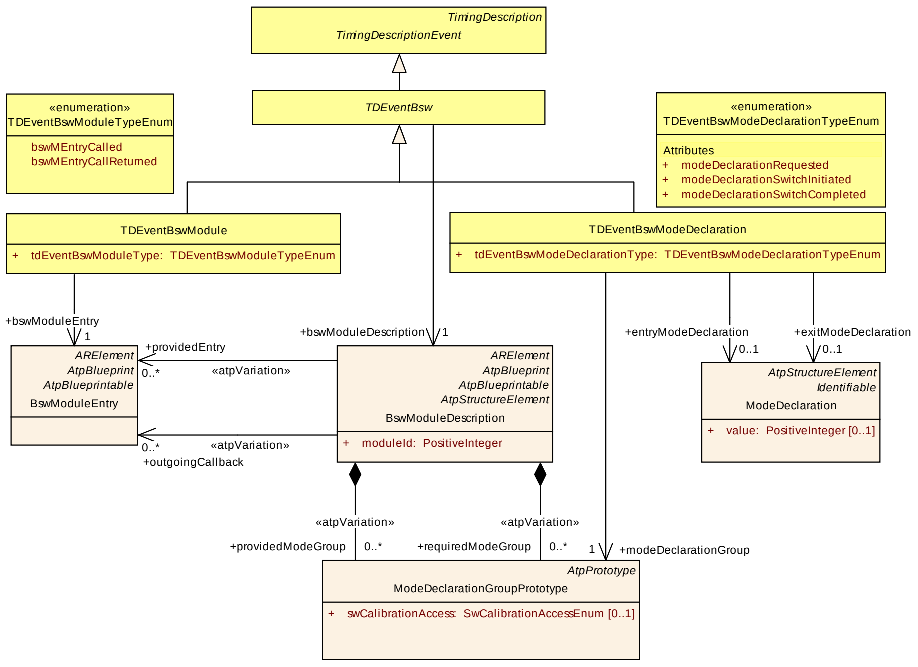


Figure 3.7: Events dealing with inter BSW module relations and mode communications on BSW level

[TPS_TIMEX_0030] `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_0001)*

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.

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
tdEventBswModuleType	TDEventBswModuleTypeEnum	1	attr	The specific type of this timing event.

Table 3.31: TDEventBswModule

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

Table 3.32: TDEventBswModuleTypeEnum

[TPS_TIMEX_0031] `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_0001)*

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.
modeDeclarationGroup	ModeDeclarationGroupPrototype	1	ref	The scope of this timing event.
tdEventBswModeDeclarationType	TDEventBswModeDeclarationTypeEnum	1	attr	The specific type of this timing event.

Table 3.33: 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.
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 by the BswM.

Table 3.34: TDEventBswModeDeclarationTypeEnum

3.2.5 Complex timing event

[TPS_TIMEX_0027] Purpose of `TDEventComplex` ⌈ The element `TDEventComplex` is used to specify relationships between occurrences of events. ⌋*(RS_TIMEX_0001)*

Complex timing events can be used during the specification of:

- `VfbTiming` 2.2
- `SwcTiming` 2.3
- `SystemTiming` 2.4
- `BswModuleTiming` 2.5
- `EcuTiming` 2.6

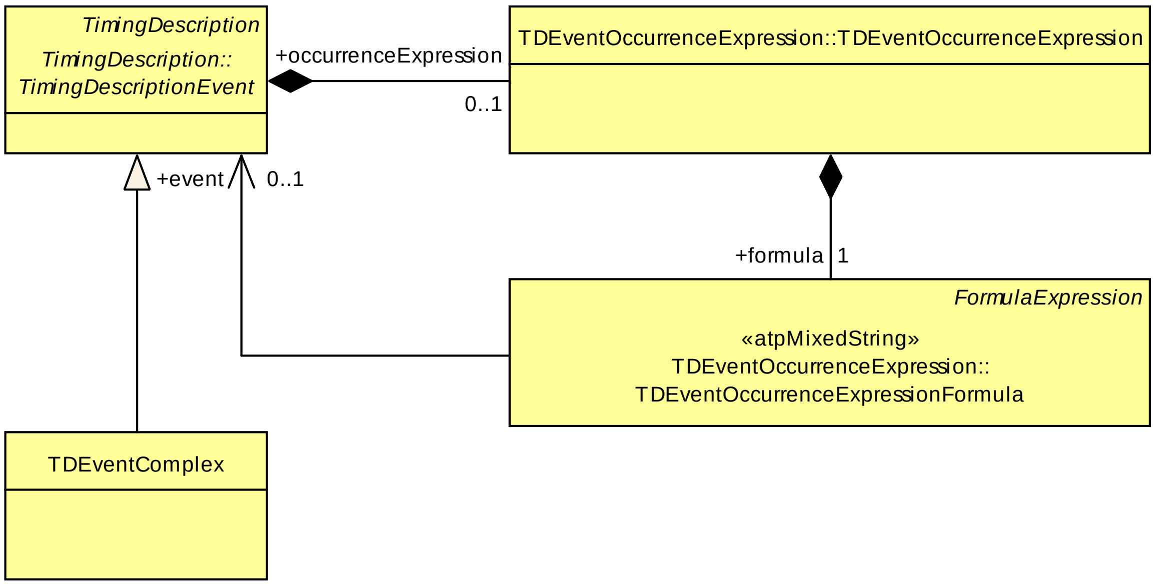


Figure 3.8: Complex timing event

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 3.35: 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 `VariableDataProtype` is a mandatory context information for events of type `TDEventVariableDataPrototype`). Instead, a complex event uses the occurrence expression (see next section 3.2.6) to specify the context by relating occurrences of usual (also called *atomic*) `TimingDescriptionEvent`s.

3.2.6 Occurrence Expression Language for timing events

The `TimingDescriptionEvent`s 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]).

3.2.6.1 Specifying an Occurrence Expression

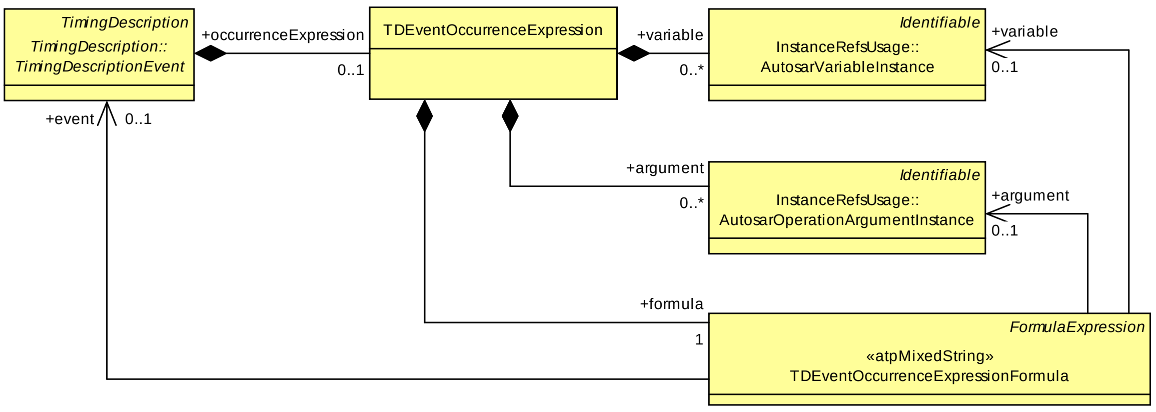

TimingDescription
TimingDescription::TimingDescriptionEvent
+occurrenceExpression
0..1
TDEventOccurrenceExpression
+variable
0..*
Identifiable
InstanceRefsUsage::AutosarVariableInstance
+variable
0..1
+event
0..1
+argument
0..*
Identifiable
InstanceRefsUsage::AutosarOperationArgumentInstance
+argument
0..1
+formula
1
FormulaExpression
«atpMixedString»
TDEventOccurrenceExpressionFormula

Figure 3.9: The occurrence expression

As shown in figure 3.9, each `TimingDescriptionEvent` aggegregates 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 [4]). The `TDE-`

ventOccurrenceExpressionFormula uses the capabilities of the FormulaExpression and adds the following functions to the expression language:

- The function *ARTE_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 *ARTE_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 *ARTE_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 *ARTE_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 *ARTE_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 *ARTE_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 *ARTE_occurs*, *ARTE_hasOccurred*, *ARTE_timeSinceLastOccurrence* and *ARTE_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 *occurs*, *hasOccurred*, *timeSinceLastOccurrence* and *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 `TimingDescriptionEvent`s (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 *ARTE_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 3.10 shows the software architecture required for this example.

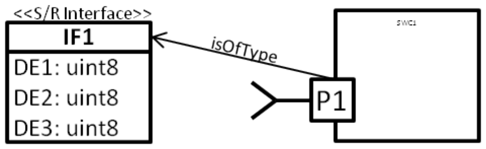


Figure 3.10: The software architecture used by the Occurrence Expression Example

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 `PortProtype` *P1* of software component *SWC1*.

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

Condition3 The `VariableDataPrototype`s *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
(ARTE_occurs(event(/example/expression/E1))
   || ARTE_occurs(event(/example/expression/E2)))
//Condition 2
&& ARTE_value(variable(/example/expression/EC/D3))>3
//Condition 3
&& abs(ARTE_timeSinceLastOccurrence(event(/example/expression/E1)) -
   ARTE_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 3.1: 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>
              ((ARTE_occurs(<EVENT-REF DEST="TD-EVENT-VARIABLE-DATA-
                 PROTOTYPE">/example/expression/E1</EVENT-REF>)
              || ARTE_occurs(<EVENT-REF DEST="TD-EVENT-VARIABLE-DATA-
                 PROTOTYPE">/example/expression/E2</EVENT-REF>))
              && ARTE_value(<VARIABLE-REF DEST="AUTOSAR-VARIABLE-INSTANCE">
                 /example/expression/EC/D3</VARIABLE-REF>) > 3
              && abs(ARTE_timeSinceLastOccurrence(<EVENT-REF DEST="TD-EVENT
                 -VARIABLE-DATA-PROTOTYPE">/example/expression/E1</EVENT-
                 REF>)
              - ARTE_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 aggegrates 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.
variable	AutosarVariableInstance	*	aggr	An occurrence expression can reference an arbitrary number of VariableDataPrototpyes in its expression. This association aggegrates instanceRefs to VariableDataPrototypes which can be referenced in the expression.

Table 3.36: TDEventOccurrenceExpression

Class	≪`atpMixedString`≫ **TDEventOccurrenceExpressionFormula**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventOccurrenceExpression			
Note	This is an extension of the FormulaExpression for the AUTOSAR Timing Extensions. A TDEventOccurrenceExpressionFormula provides the means to express the temporal characteristics of timing event occurrences in correlation with specific variable and argument values. The formal definition of the extended formula expression language is described in detail in the AUTOSAR Timing Extensions.			
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 3.37: TDEventOccurrenceExpressionFormula

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

Table 3.38: AutosarVariableInstance

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

Table 3.39: AutosarOperationArgumentInstance

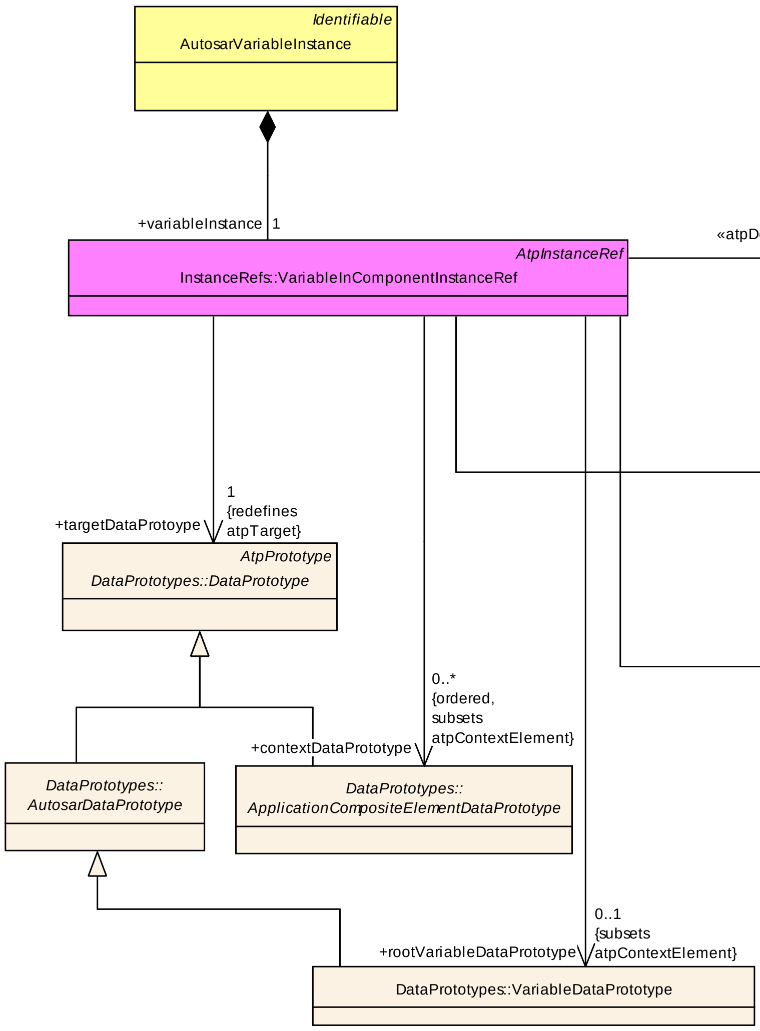


Figure 3.11: The required context information to reference to a variable instance within AUTOSAR.

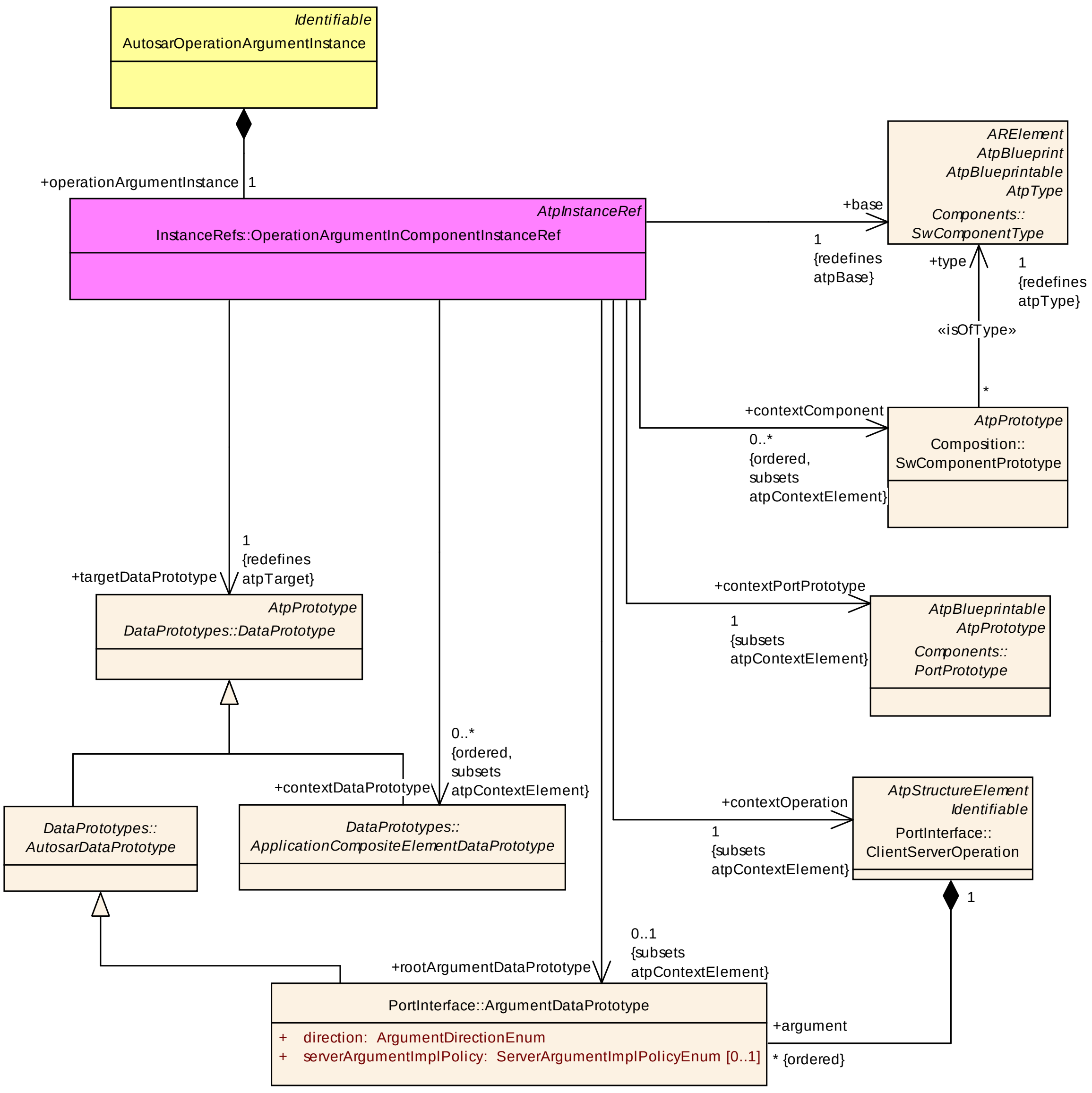

Identifiable
AutosarOperationArgumentInstance
+operationArgumentInstance 1
AtpInstanceRef
InstanceRefs::OperationArgumentInComponentInstanceRef
+base
1
{redefines atpBase}
ARElement
AtpBlueprint
AtpBlueprintable
AtpType
Components::SwComponentType
+type
1
{redefines atpType}
«isOfType»
*
+contextComponent
0..*
{ordered, subsets atpContextElement}
AtpPrototype
Composition::SwComponentPrototype
+targetDataPrototype
1
{redefines atpTarget}
AtpPrototype
DataPrototypes::DataPrototype
+contextPortPrototype
1
{subsets atpContextElement}
AtpBlueprintable
AtpPrototype
Components::PortPrototype
0..*
{ordered, subsets atpContextElement}
+contextDataPrototype
DataPrototypes::AutosarDataPrototype
DataPrototypes::ApplicationCompositeElementDataPrototype
+contextOperation
1
{subsets atpContextElement}
AtpStructureElement
Identifiable
PortInterface::ClientServerOperation
1
0..1
{subsets atpContextElement}
+rootArgumentDataPrototype
PortInterface::ArgumentDataPrototype
+ direction: ArgumentDirectionEnum
+ serverArgumentImplPolicy: ServerArgumentImplPolicyEnum [0..1]
+argument
* {ordered}

Figure 3.12: The required context information to reference to an operation argument instance within AUTOSAR.

3.2.6.2 Occurrence Expression Language syntax

The occurrence expression language is based on the syntax of the formula language defined in the Generic Structure Template [4]. 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 3.2: AUTOSAR Occurrence Expression Language

```
grammar occurrenceFilterLanguage;

// this grammar uses extends production rules of the autosarFormulaLanguage
    defined in the Generic Structure Template.
// The production rule 'atom' defined in the autosarFormulaLanguage is
   replaced via the statement below

atom
   :   DecIntegerLiteral
           | HexIntegerLiteral
           | OctIntegerLiteral
           | BinIntegerLiteral
           | DecimalLiteral
           | DoubleLiteral
           | Epsilon
           | reference
           | ArgumentOperand
           | DefinedFuncName LPAREN reference RPAREN
           | StringFuncName LPAREN stringArg COMMA stringArg RPAREN
           | BinaryFuncName LPAREN atomExpr COMMA atomExpr RPAREN
           | UnaryFuncName LPAREN atomExpr RPAREN
           | TimingUnaryFuncName LPAREN timingReference RPAREN
           | (LPAREN atomExpr RPAREN)
            ;

timingReference:
   :      ('event' | 'argument' | 'variable') LPAREN  '/'? NCName ('/'
      NCName)* RPAREN;

TimingUnaryFuncName
   :    'ARTE_value' | 'ARTE_occurs' | 'ARTE_hasOccurred' | '
      ARTE_timeSinceLastOccurrence' | 'ARTE_angleSinceLastOccurrence'
        ;
```

3.2.6.3 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 TimingDescriptionEvent, 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.

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

3.2.6.3.2 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) =
(ARTE_occurs(t,event(/example/expression/E1))
   || ARTE_occurs(t,event(/example/expression/E2)))
&& ARTE_value(t, variable(/example/expression/EC/D3)) > 3
&& abs(ARTE_timeSinceLastOccurrence(t, event(/example/expression/E1)) -
   ARTE_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 & Simulation: In a deterministic system environment, occurrences of `TimingDescriptionEvent`s 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 `TDEventSwcInternalBehavior`, 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:

- ARTE_value(t, <reference to an `AutosarVariableInstance`>): returns the variable value at time *t*
- ARTE_value(t, <reference to an `AutosarOperationArgumentInstance`>): returns the operation argument value at time *t*
- ARTE_value(t, <reference to an `ISignal`>): returns the isignal value at time *t*
- ARTE_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)
- ARTE_hasOccurred(t, <reference to a `TimingDescriptionEvent`): returns TRUE (or 1) if the referenced event has occurred AT LEAST ONCE before (or at) time *t*.
- ARTE_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.
- ARTE_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.

3.3 TimingDescriptionEventChain

[TPS_TIMEX_0002] 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_0001, RS_TIMEX_0004, RS_TIMEX_0005, RS_TIMEX_0009)*

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.

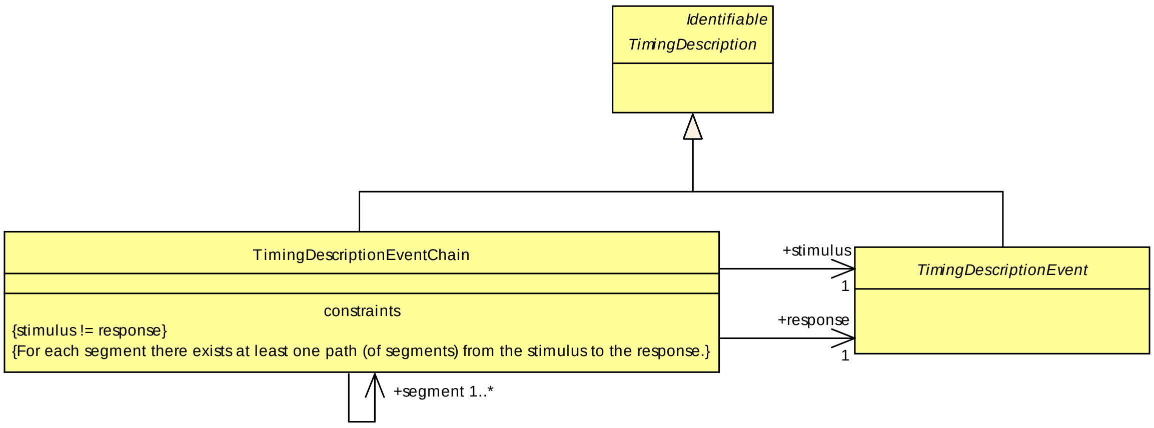

Identifiable
TimingDescription
TimingDescriptionEventChain
constraints
{stimulus != response}
{For each segment there exists at least one path (of segments) from the stimulus to the response.}
+stimulus
1
+response
1
TimingDescriptionEvent
+segment 1..*

Figure 3.13: TimingDescriptionEventChain

Class	**TimingDescriptionEventChain**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription			
Note	An event chain describes the causal order for a set of functionally dependent timing events. Each event chain has a well defined stimulus and response, which describe its start and end point. Furthermore, it can be hierarchically decomposed into an arbitrary number of sub-chains, so called *event chain segments*.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TimingDescription			
Attribute	***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 3.40: 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 3.14, 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 itsself via the segment relation. This implies that the event chain is atomic and cannot be further decomposed.

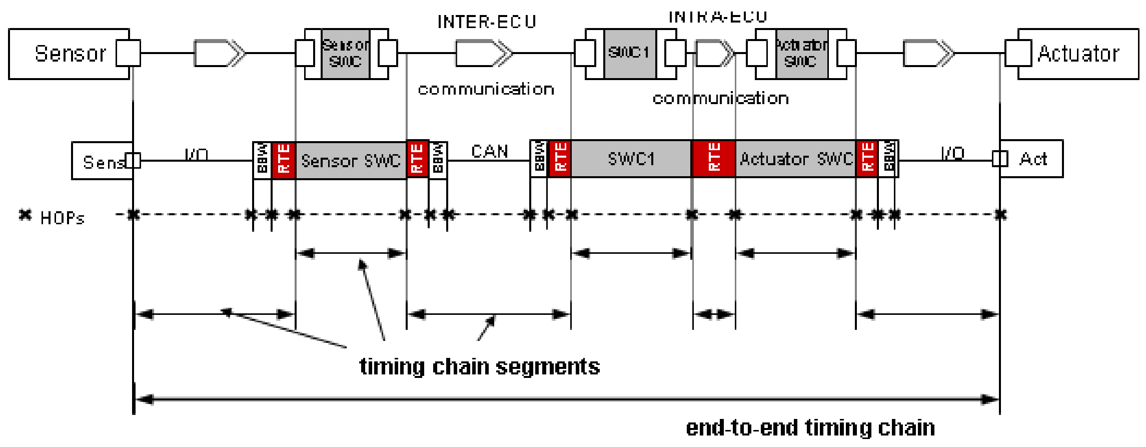


Figure 3.14: Example of an end-to-end event chain between sensor sampling and actuator access

3.4 EventTriggeringConstraint

[TPS_TIMEX_0003] `EventTriggeringConstraint` specifies occurrence behavior respectively model ⌈ The element `EventTriggeringConstraint` is used to specify the particular occurrences of a given timing description event. ⌋*(RS_TIMEX_0001, RS_TIMEX_0002, RS_TIMEX_0006, RS_TIMEX_0008)*

AUTOSAR offers four basic types of event triggering as depicted in figure 3.15.

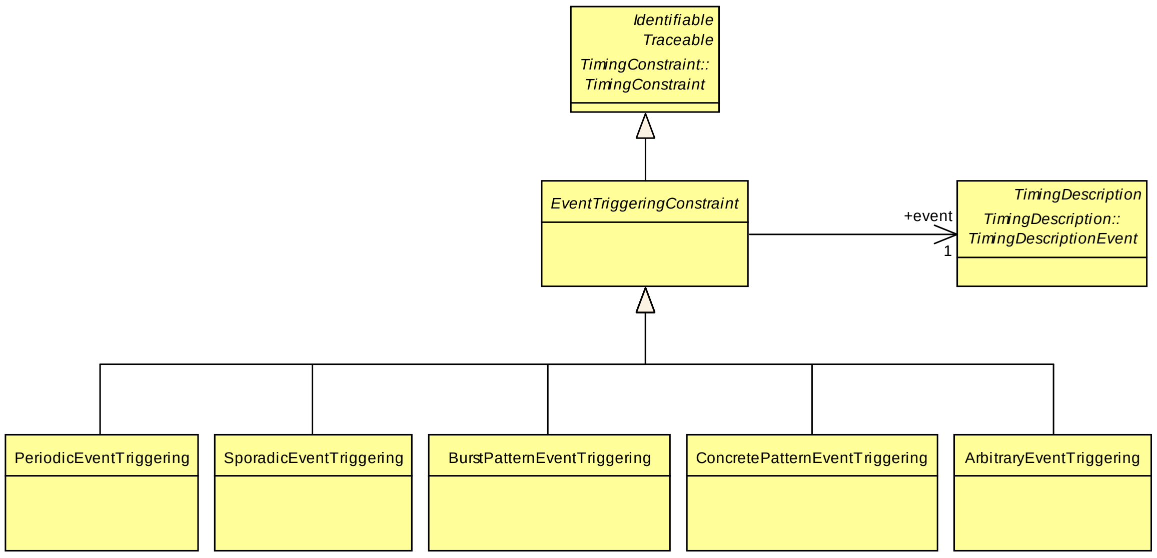


Figure 3.15: The different types of event triggerings

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 3.41: EventTriggeringConstraint

Timing analysis literature [5] defines that any event triggering used for timing analysis must provide the following information:

- the minimum (maximum) number of events that can occur in a given time interval
- the minimum (maximum) time interval in which a given number of events can occur

Figure 3.16 shows an example for an event triggering and the corresponding maximum occurrence curve over time.

In the following, the different types of event triggering specifications are described in more detail.

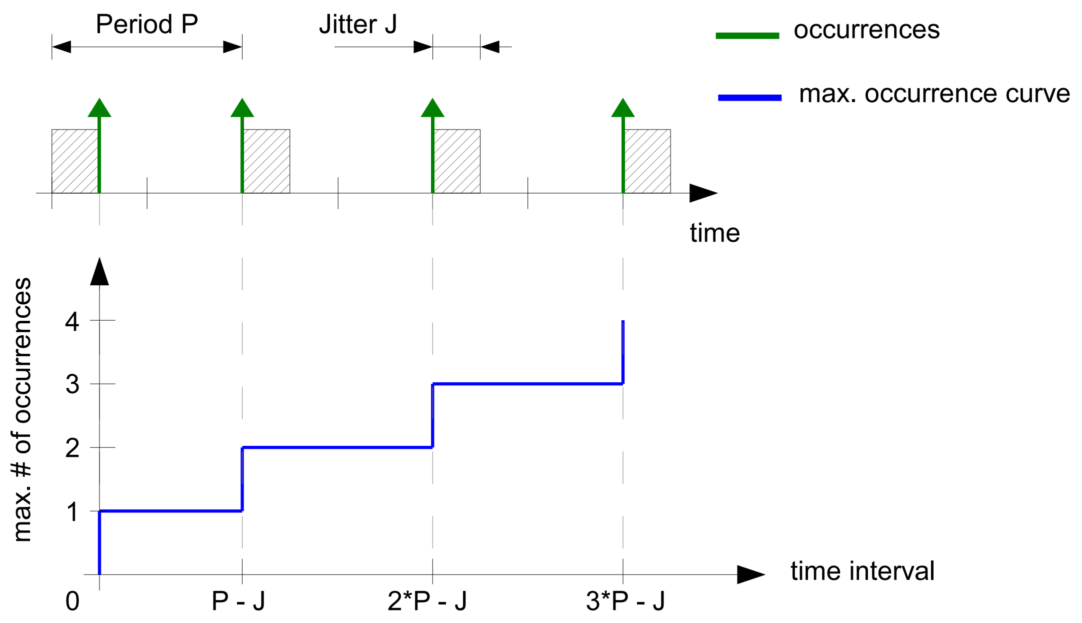


Figure 3.16: An example for determining the maximum occurrence curve of an event triggering

3.4.1 PeriodicEventTriggering

[TPS_TIMEX_0010] `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_0001, RS_TIMEX_0002, RS_TIMEX_0006, RS_TIMEX_0008)*

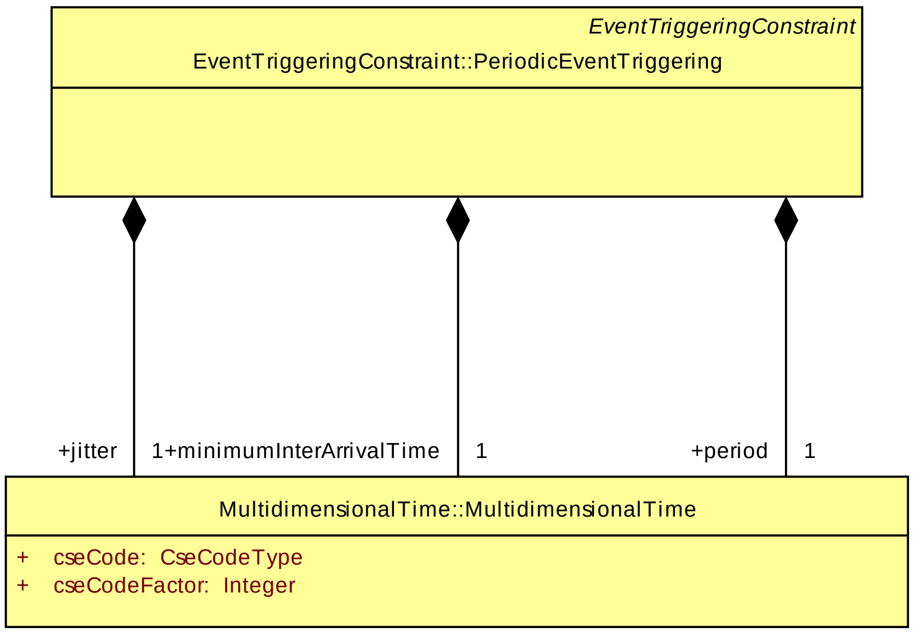

EventTriggeringConstraint
EventTriggeringConstraint::PeriodicEventTriggering
+jitter
1
+minimumInterArrivalTime
1
+period
1
MultidimensionalTime::MultidimensionalTime
+ cseCode: CseCodeType
+ cseCodeFactor: Integer

Figure 3.17: PeriodicEventTriggering

Class	**PeriodicEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	The PeriodicEventTriggering describes the behavior of an event with a strict periodic occurrence pattern, given by the period attribute. Additionally, it is possible to soften the strictness of the periodic occurrence behavior by specifying a jitter, so that there can be a deviation from the period up to the size of the jitter.			
Base	ARObject,EventTriggeringConstraint,Identifiable,MultilanguageReferrable,Referrable,TimingConstraint,Traceable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
jitter	MultidimensionalTime	1	aggr	The maximum jitter of the periodic event occurrence. Jitter=max |nthPeriod - standardPeriod| **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	1	aggr	The period of the event occurrence. **Tags:** xml.sequenceOffset=30

Table 3.42: PeriodicEventTriggering

With this event triggering, events with a periodic occurrence can be described (see examples in figure 3.18).

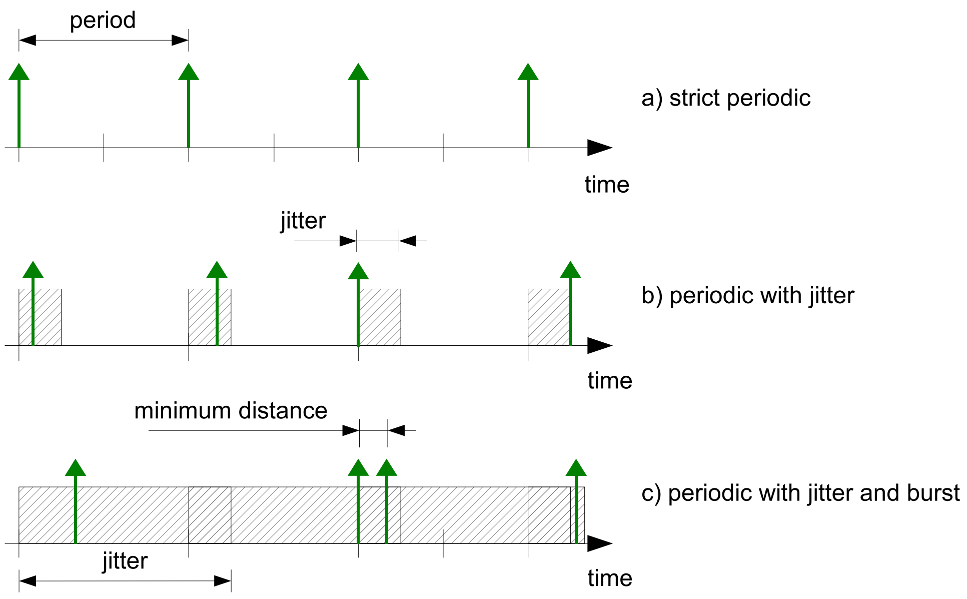


Figure 3.18: Examples showing the different kinds of periodic event triggering

For the latter case, if an occurrence behavior has a jitter, it can happen that several instances of the event can occur (theoretically) simultaneously (a so-called ”burst”). This can happen, if the jitter is greater or equal than the period of the event triggering (see example c in figure 3.18)).

In this case, the modeler may want to specify the minimum distance between two event instances. For example the event ”Signal X has arrived in the HW buffer of the receiving ECU” can never happen more then once at the same time, since there must be at least the transmission time on the bus for the signal in between.

3.4.2 SporadicEventTriggering

[TPS_TIMEX_0011] `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_0001, RS_TIMEX_0002, RS_TIMEX_0006, RS_TIMEX_0008)*

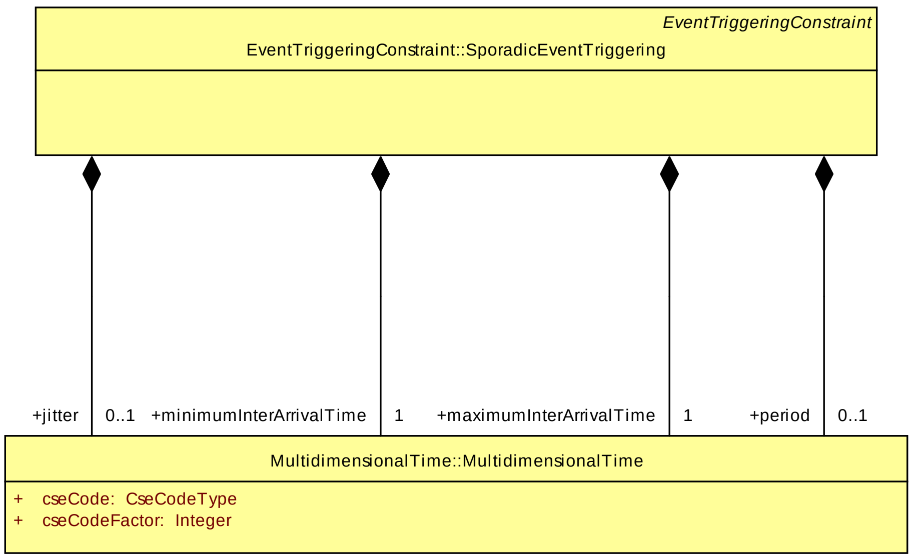


Figure 3.19: SporadicEventTriggering

Class	**SporadicEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	The SporadicEventTriggering describes the behavior of an event which occurs occasionally or singly.			
Base	ARObject,EventTriggeringConstraint,Identifiable,MultilanguageReferrable,Referrable,TimingConstraint,Traceable			
Attribute	***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 3.43: SporadicEventTriggering

This is a generalization of the periodic event triggering described in section 3.4.1. 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.

3.4.3 ConcretePatternEventTriggering

[TPS_TIMEX_0012] `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_0001, RS_TIMEX_0002, RS_TIMEX_0006, RS_TIMEX_0008)*

This describes events which occur following a known pattern.

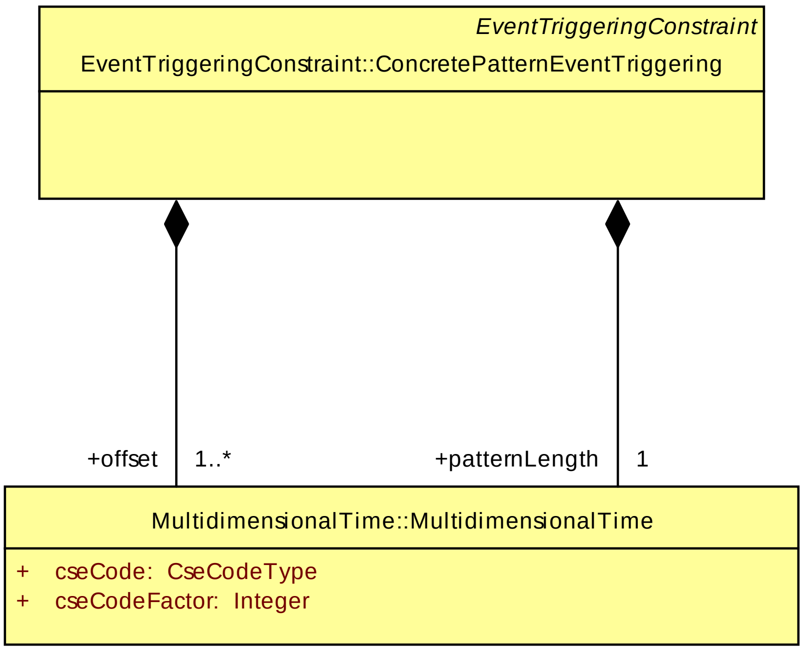

EventTriggeringConstraint
EventTriggeringConstraint::ConcretePatternEventTriggering
+offset
1..*
+patternLength
1
MultidimensionalTime::MultidimensionalTime
+ cseCode: CseCodeType
+ cseCodeFactor: Integer

Figure 3.20: ConcretePatternEventTriggering

Class	**ConcretePatternEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::Event TriggeringConstraint			
Note	The ConcretePatternEventTriggering describes the behavior of an event, which occurs following a precisely known pattern.			
Base	ARObject,EventTriggeringConstraint,Identifiable,Multilanguage Referrable,Referrable,TimingConstraint,Traceable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
offset	Multidimensiona lTime	1..*	aggr	The offset for each occurrence of the event in the specified time interval.
patternLen gth	Multidimensiona lTime	1	aggr	The length of the observed time interval.

Table 3.44: ConcretePatternEventTriggering

When modeling the `ConcretePatternEventTriggering`, at first the observed time interval must be specified with the attribute `patternLength`. Then, the arrival of the event occurrences in the specified interval is described by modeling an array of offsets for the event occurrences. In the example in figure 3.21 the interval length is 12 and the offset array for the offsets o0, o1, o2, etc. is [0,1,3,6,7,9,11].

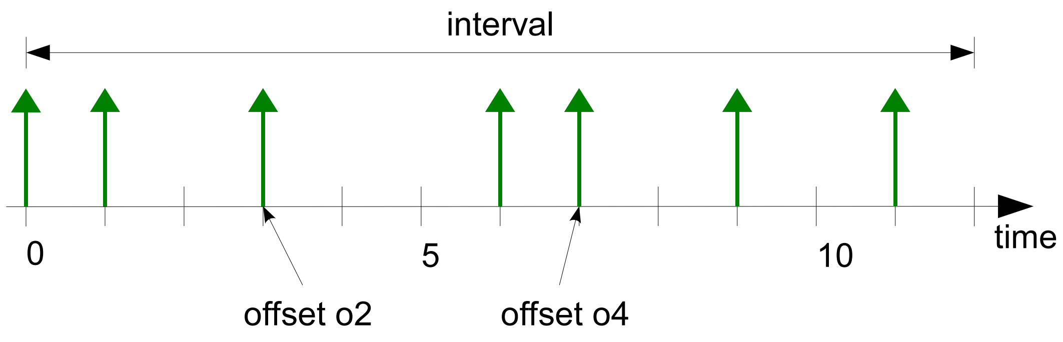


Figure 3.21: ConcretePatternEventTriggering including several offsets

3.4.4 BurstPatternEventTriggering

[TPS_TIMEX_0013] `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_0001, RS_TIMEX_0002, RS_TIMEX_0006, RS_TIMEX_0008)*

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.

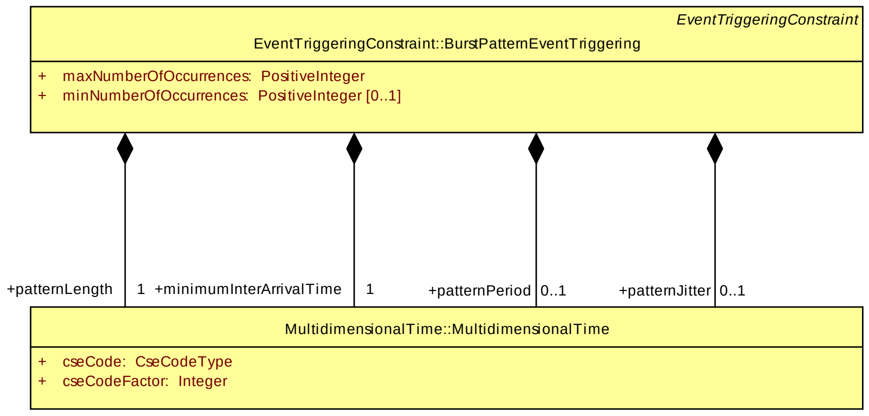

EventTriggeringConstraint
EventTriggeringConstraint::BurstPatternEventTriggering
+ maxNumberOfOccurrences: PositiveInteger
+ minNumberOfOccurrences: PositiveInteger [0..1]
+patternLength
1
+minimumInterArrivalTime
1
+patternPeriod
0..1
+patternJitter
0..1
MultidimensionalTime::MultidimensionalTime
+ cseCode: CseCodeType
+ cseCodeFactor: Integer

Figure 3.22: BurstPatternEventTriggering

Class	**BurstPatternEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	A BurstPatternEventTriggering describes the maximum number of occurrences of the same event in a given time interval. This is typically used to model a worst case activation scenario.			
Base	ARObject,EventTriggeringConstraint,Identifiable,MultilanguageReferrable,Referrable,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".
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 3.45: BurstPatternEventTriggering

The parameters are described in the following and are illustrated in Figure 3.23 and Figure 3.24.

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 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: $0 \leq \texttt{minNumberOfOccurrences} \leq \texttt{maxNumberOfOccurrences}$.
⌋

[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: $0 < \texttt{minimumInterArrivalTime} \leq \texttt{patternLength}$. ⌋

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

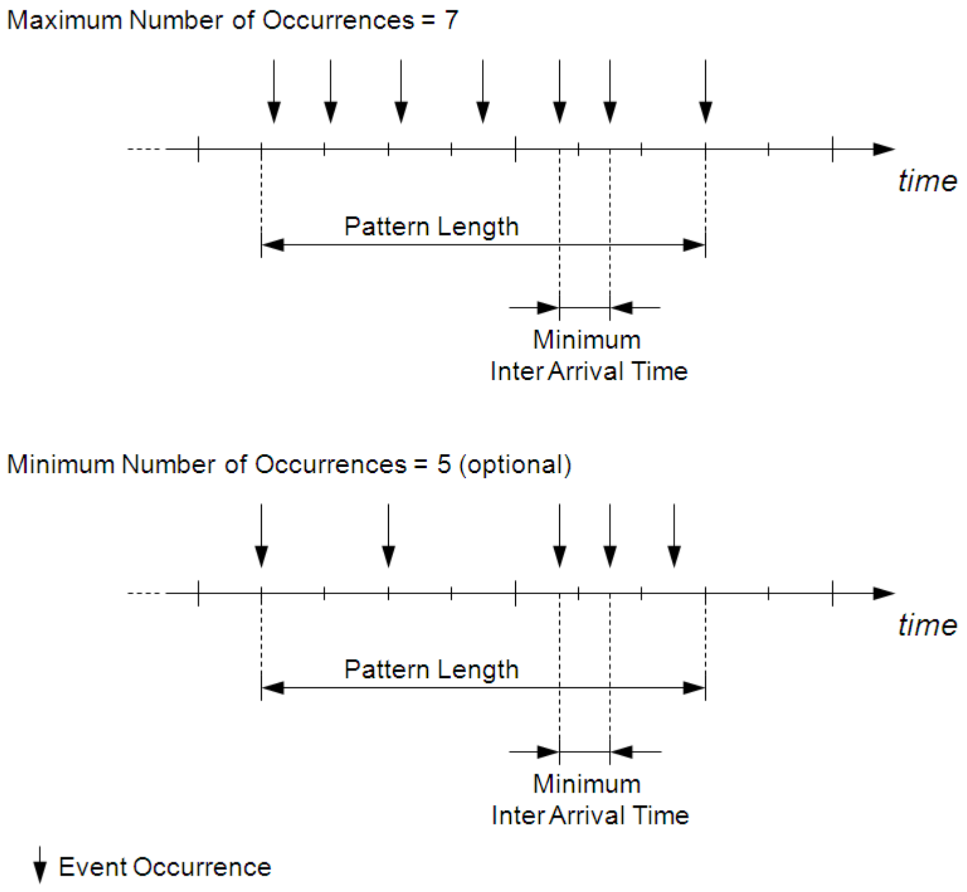


Figure 3.23: Paremeters characterizing the Burst Pattern Event Triggering

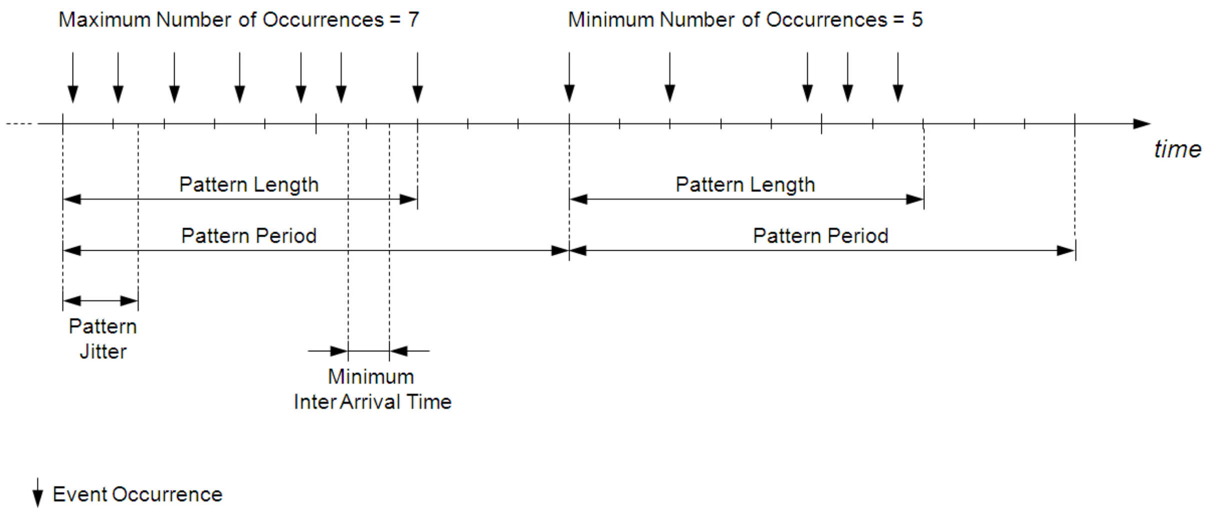


Figure 3.24: Parameters characterizing the Burst Pattern Event Triggering when periodically being repeated

3.4.5 ArbitraryEventTriggering

[TPS_TIMEX_0014] `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_0001, RS_TIMEX_0002, RS_TIMEX_0006, RS_TIMEX_0008)*

This describes the occasional occurrence of a timing event.

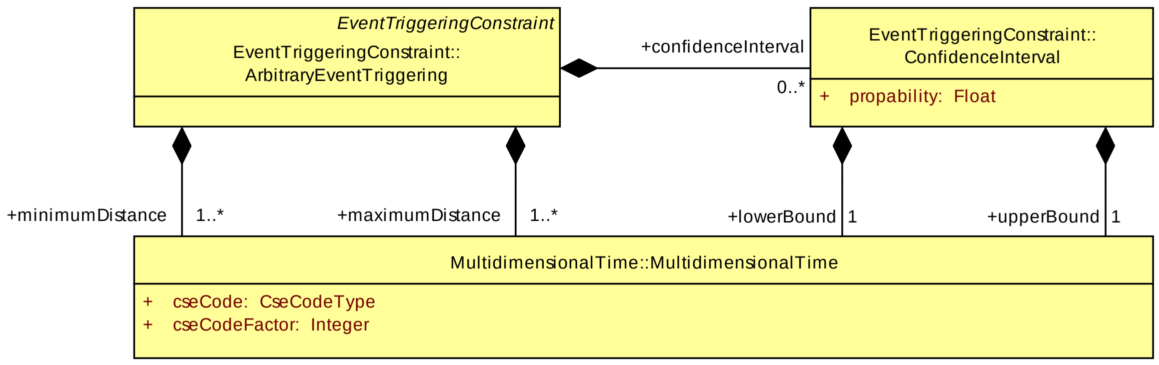


Figure 3.25: ArbitraryEventTriggering

Class	**ArbitraryEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::Event TriggeringConstraint			
Note	The ArbitraryEventTriggering describes that an event occurs occasionally, singly, irregularly or randomly. The primary purpose of this event triggering is to abstract event occurrences captured by data acquisition tools (background debugger, trace analyzer, etc.) during system runtime.			
Base	ARObject,EventTriggeringConstraint,Identifiable,Multilanguage Referrable,Referrable,TimingConstraint,Traceable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
confidence Interval	ConfidenceInter val	*	aggr	List of confidence intervals. **Tags:** xml.sequenceOffset=30
maximum Distance	Multidimensiona lTime	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.sequenceOffset=20

Attribute	Datatype	Mul.	Kind	Note
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.sequenceOffset=10

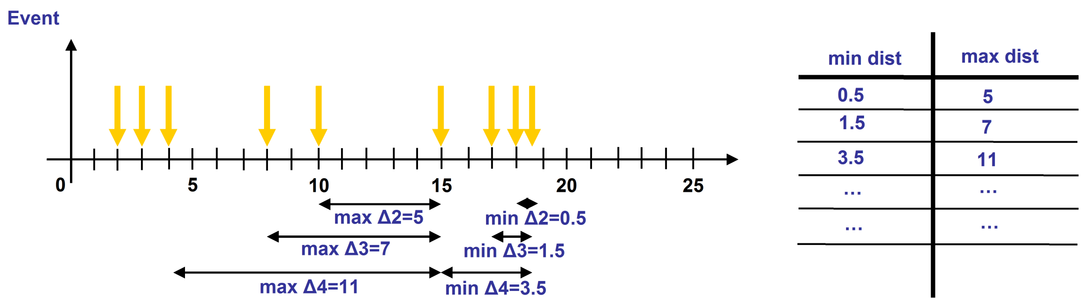
Table 3.46: ArbitraryEventTriggering

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.
propability	Float	1	attr	The probability for the measured lower and upper bound of the confidence interval.
upperBound	MultidimensionalTime	1	aggr	The upper bound of the expected distance of two consecutive event occurrences.

Table 3.47: 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 semantics of the attributes `minimumInterArrivalTime` and `maximumInterArrivalTime` are illustrated in figure 3.26 below.



min dist	max dist
0.5	5
1.5	7
3.5	11
...	...
...	...

Figure 3.26: ArbitraryEventTriggering

3.5 LatencyTimingConstraint

[TPS_TIMEX_0004] `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_0001, RS_TIMEX_0002, RS_TIMEX_0012)*

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 restricts the 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 3.27.

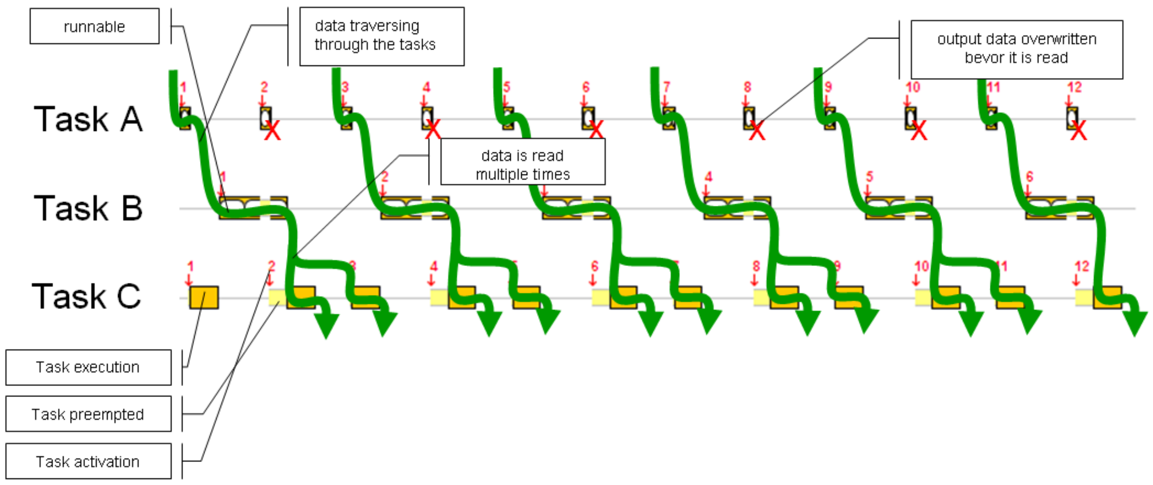

runnable
data traversing through the tasks
output data overwritten bevor it is read
Task A
Task B
Task C
data is read multiple times
Task execution
Task preempted
Task activation

Figure 3.27: 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 *age* of data 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.

[1] A synonym for delay

The *reaction* 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.

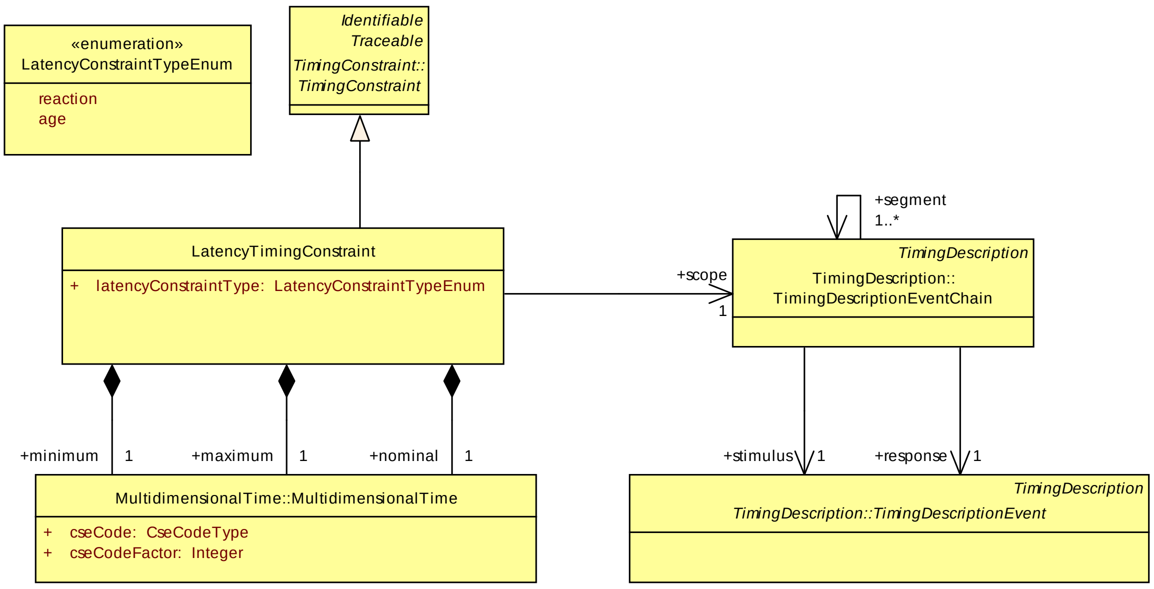

«enumeration»
LatencyConstraintTypeEnum
reaction
age
Identifiable
Traceable
TimingConstraint::TimingConstraint
LatencyTimingConstraint
+ latencyConstraintType: LatencyConstraintTypeEnum
+scope
1
+segment
1..*
TimingDescription
TimingDescription::TimingDescriptionEventChain
+minimum 1
+maximum 1
+nominal 1
+stimulus 1
+response 1
MultidimensionalTime::MultidimensionalTime
+ cseCode: CseCodeType
+ cseCodeFactor: Integer
TimingDescription
TimingDescription::TimingDescriptionEvent

Figure 3.28: Latency constraint

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

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
minimum	MultidimensionalTime	1	aggr	The minimum latency between the occurrence of the stimulus and the occureence of the corresponding response of the associatied event chain. **Tags:** xml.sequenceOffset=10
nominal	MultidimensionalTime	1	aggr	The nominal latency between the occurrence of the stimulus and the occureence of the corresponding response of the associatied event chain. **Tags:** xml.sequenceOffset=30
scope	TimingDescriptionEventChain	1	ref	The event chain that defines the scope of the constraint.

Table 3.48: 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 3.49: 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.

3.6 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 or not important in the context of such a constraint. In this case the definition of a `TDEventChain` is not possible.

[TPS_TIMEX_0005] `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_0001, RS_TIMEX_0009)*

Instead of an event chain, its scope is a `TDEventVariablePrototype`. The age constraint refers to that event. Every time the scope event occurs, the `VariableDataPrototype` shall have the specified data age.

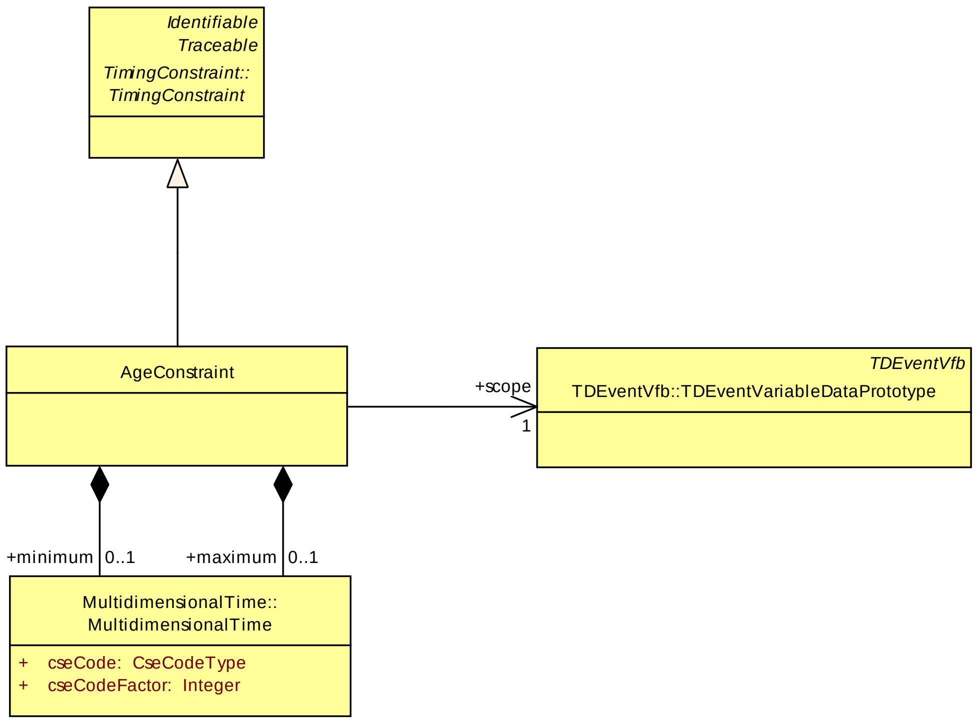


Figure 3.29: Age constraint

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

[constr_4504] Restricted usage of `AgeConstraint` ⌈ An `AgeConstraint` can only be defined for events of type `TDEventVariablePrototype` with enum type `variableDataPrototypeReceived`. ⌋

Class	**AgeConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::Age Constraint			
Note	The AgeConstraint is used to impose a constraint on the age of a VariableDataPrototype 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	Multidimensiona lTime	0..1	aggr	The maximum age.
minimum	Multidimensiona lTime	0..1	aggr	The minimum age.
scope	TDEventVariabl eDataPrototype	1	ref	The scope of an AgeConstraint is a TDEventVariableDataPrototype. The constraint is applied to the VariableDataPrototype referenced by this event.

Table 3.50: AgeConstraint

3.7 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. Herefore, synchronization constraints are used.

[TPS_TIMEX_0006] `SynchronizationTimingConstraint` specifies synchronicity constraints ⌈ The element `SynchronizationTimingConstraint` is used to specify a synchronicity constraint among the occurrences of two or more timing description events which are referenced. ⌋*(RS_TIMEX_0001, RS_TIMEX_0002, RS_TIMEX_0007, RS_TIMEX_0008)*

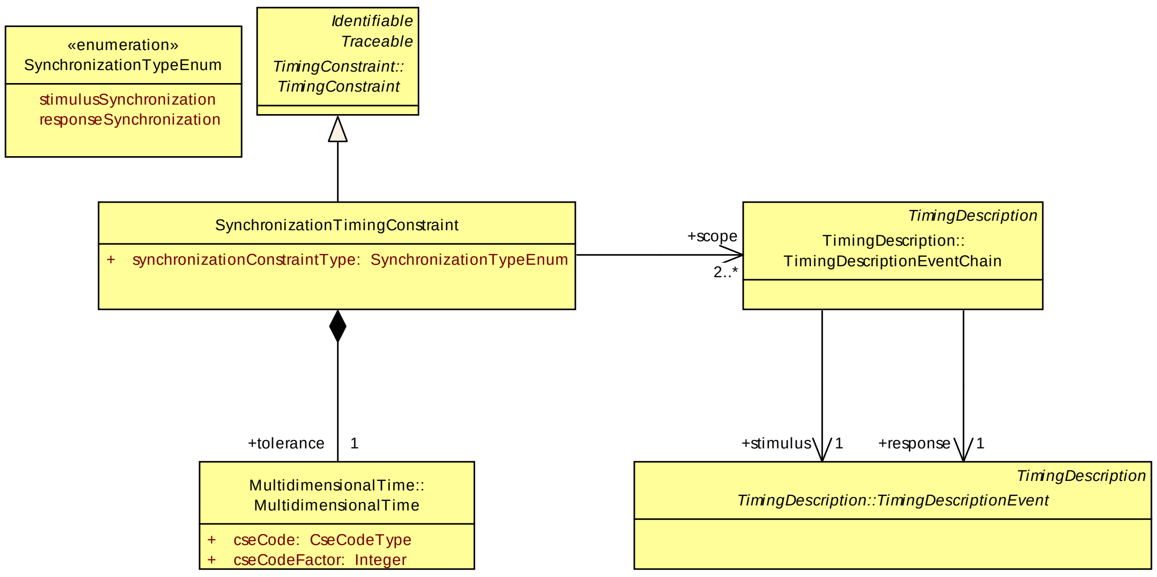


Figure 3.30: Synchronization constraint

In the timing specification a `SynchronizationTimingConstraint` is associated with two or more `TimingDescriptionEventChain`s, that either have a common stimulus with distinct responses (fork) or a common response with distinct stimuli (join). The type of the constraint is set by the attribute `synchronizationConstraintType` which can be either `stimulusSynchronization` or `responseSynchronization`.

Class	**SynchronizationTimingConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::SynchronizationTimingConstraint			
Note	This constraint is used to restrict the timing behavior of different, but correlated event chains, in regard to synchronization. Thereby, the following two scenarios are supported: 1) An arbitrary number of correlated event chains with a common stimulus, but different responses, where the responses shall occur synchronously with respect to a predefined tolerance. 2) 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.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TimingConstraint,Traceable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
scope	TimingDescriptionEventChain	2..*	ref	referenced event chains
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 3.51: SynchronizationTimingConstraint

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	The responses of the associated event chains must occur synchronously with respect to the specified tolerance. All associated event chains must have the same stimulus.
stimulusSynchronization	The stimuli of the associated event chains must occur synchronously with respect to the specified tolerance. All associated event chains must have the same response.

Table 3.52: SynchronizationTypeEnum

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.

3.8 OffsetTimingConstraint

[TPS_TIMEX_0015] `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_0001, RS_TIMEX_0002, RS_TIMEX_0008)*

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.

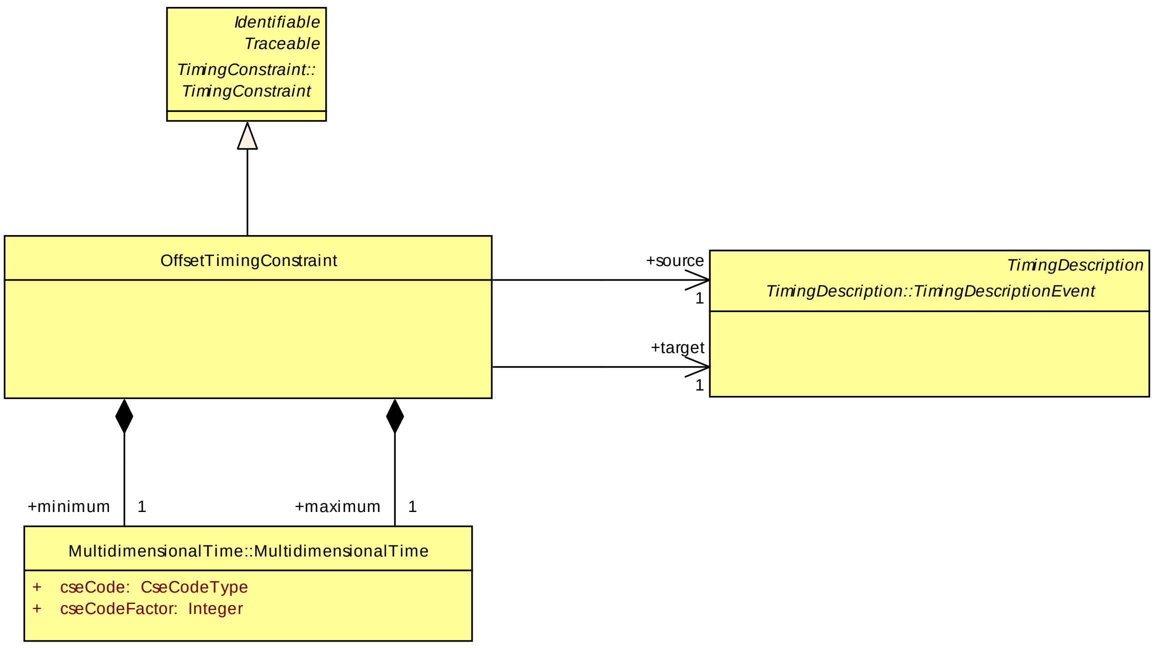


Figure 3.31: Offset constraint

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

Table 3.53: OffsetTimingConstraint

3.9 ExecutionOrderConstraint

[TPS_TIMEX_0007] `ExecutionOrderConstraint` specifies sequence of executing executable entities ⌈ The element `ExecutionOrderConstraint` is used to specify the order of execution of `ExecutableEntities`. ⌋*(RS_TIMEX_0001, RS_TIMEX_0002)*

An `ExecutionOrderConstraint` is valid, if it fulfills the following restrictions:

- All referenced `ExecutableEntities` have the same triggering frequency
- All referenced `ExecutableEntities` are active in the same mode

The above mentioned restrictions assure that the semantics of an `ExecutionOrderConstraint` is well defined.

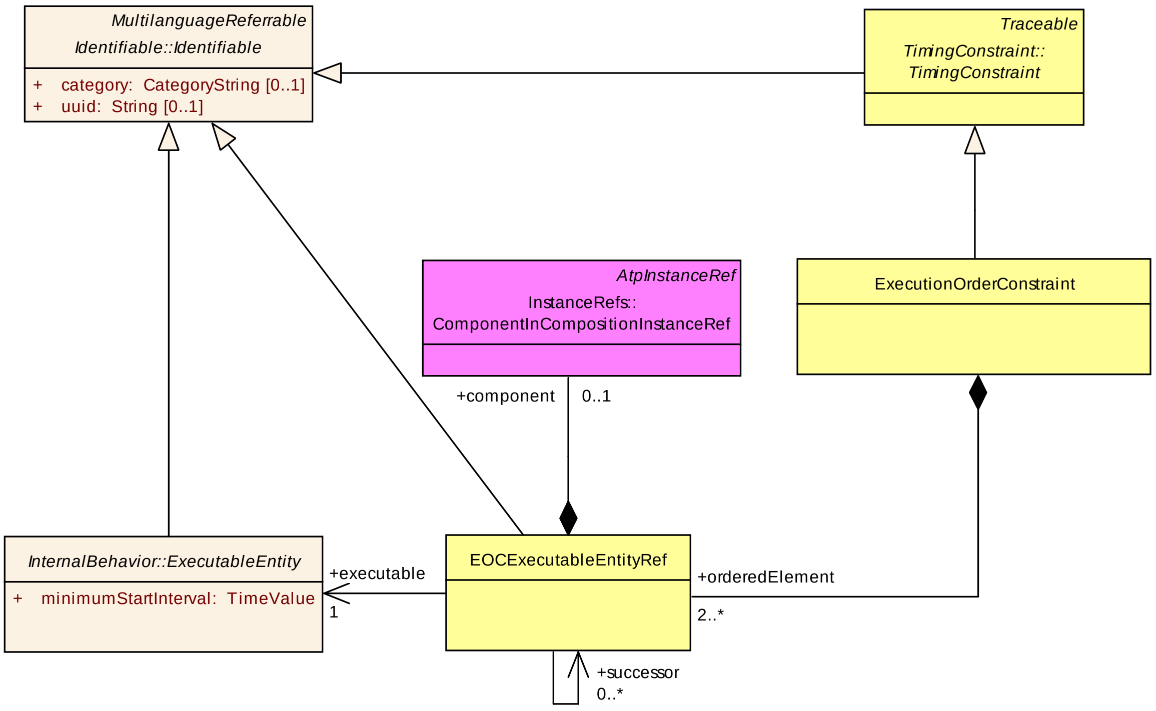


Figure 3.32: Specification of execution order constraints.

Class	**ExecutionOrderConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint			
Note	This constraint is used to restrict the order of execution for a set of ExecutableEntities. On software component level an ExecutionOrderConstraint can be defined for RunnableEntities as part of a SwcTiming. If the scope of the SwcTiming is an AtomicSwComponentType, the constrained RunnableEntities must be of the same SwcInternalBehavior (in this case, no InstanceRef is required for the definition of EOCExecutableEneityRef). If the scope of the SwcTiming is an CompositionSwComponentType, the constrained RunnableEntities must be of AtomicSwComponentTypes which belong to the composition. On BSW level, an ExectionOrderConstraint can be defined BSWModuleEntities of the same BswInternalBehavior as part of a BswModuleTiming. On system level an ExecutionOrderConstraint can be defined for RunnableEntities belonging to the same SoftwareComposition as part of a SystemTiming.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TimingConstraint,Traceable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
orderedElement	EOCExecutableEntityRef	2..*	aggr	The list of references to ExecutableEntities which shall be ordered.

Table 3.54: ExecutionOrderConstraint

Class	**EOCExecutableEntityRef**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint			
Note	This is used to define a reference to an ExecutableEntity and its successors in the execution order. If the ExecutionOrderConstraint is defined on system level, a reference to the SwComponentPrototype (via the ComponentInCompositionInstanceRef) that the referenced ExecutableEntity belongs to, must be provided as context information.			
Base	ARObject,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	1	ref	The ExecutableEntity whose execution order is restricted by the contraint.
successor	EOCExecutableEntityRef	*	ref	The list of successors for the executable entity referenced by this.

Table 3.55: EOCExecutableEntityRef

3.10 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 BSWModuleDescriptionTemplate document [6]. This execution time description represents a timing property of the system.

[TPS_TIMEX_0008] ExecutionTimeConstraint to specify execution time constraints ⌈ The element ExecutionTimeConstraint is used to specify minimum and maximum execution time constraints of executable entities. ⌋*(RS_TIMEX_0001)*

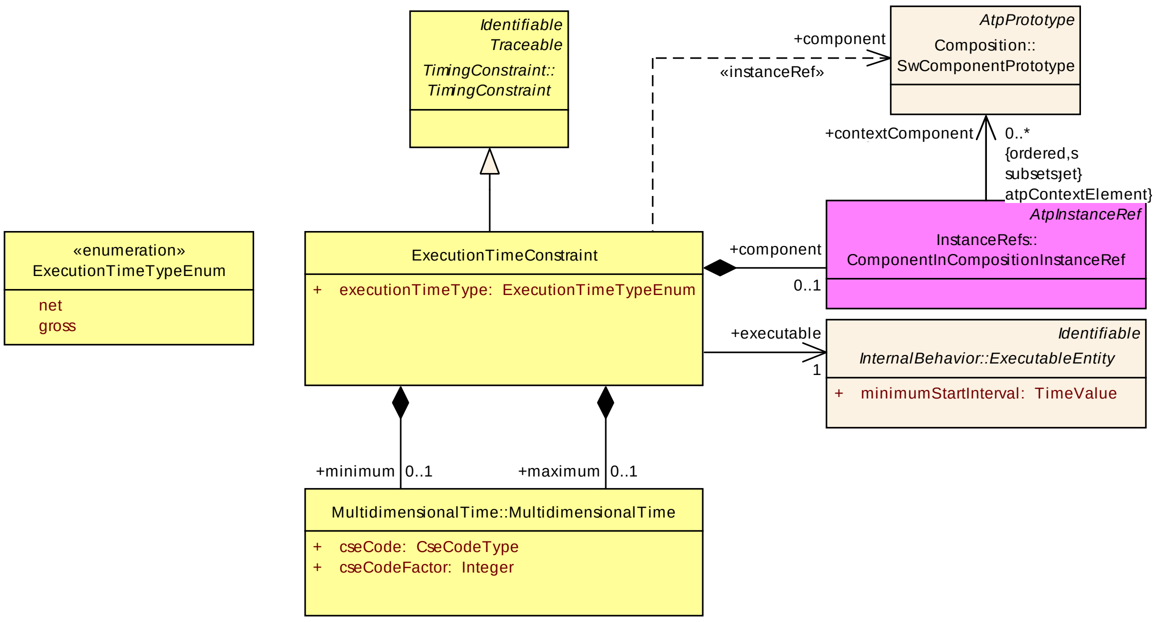


Figure 3.33: Execution time constraint

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

Class	**ExecutionTimeConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionTimeConstraint			
Note	An ExecutionTimeConstraint is used to specify the execution time of the referenced ExecutableEntity in the referenced component. A minimum and maximum execution time can be defined. Two types of execution time semantics can be used. The desired semantics can be set by the attribute executionTimeType: The "net" execution time is the time used to execute the ExecutableEntity without interruption and without external calls. The "gross" execution time is the time used to execute the ExecutableEntity without interruption including external calls to other entities. The time to execute the ExecutableEntity including interruptions by other entities and including external calls is commonly called "response time". The TimingExtensions provide the concept of event chains and latency constraints for that purpose. An event chain from the start of the entity to the termination of the entity with according latency constraint represents a response time constraint for that executable entity.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable,TimingConstraint,Traceable			
Attribute	***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 3.56: 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 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 3.57: ExecutionTimeTypeEnum

3.11 Traceability of Constraints

[TPS_TIMEX_0037] `TimingConstraint` is a `Traceable` ⌈ The element `TimingConstraint` and all of its specializations, commonly called timing constraints, are traceable. ⌋*(RS_TIMEX_0010)*

The support for traceability [7] enables one to specify for example a relationship between an RTE event activating a runnable entity and a given timing constraint. A system integrator has chosen the RTE event `TimingEvent` with a period of 20ms, because of a given timing requirement respectively constraint `PeriodicEventTriggering` requiring the periodic activation of a runnable entity every 20ms. In this case, a trace from the RTE event's document section to the corresponding timing constraint can be set. In addition this capability ensures validity between constraints and properties.

4 Application Notes

This chapter outlines two application examples describing a potential approach to use the Specification of Timing Extension in a practical scenario. Furthermore, chapter 4.3 describes the use of external VFB events.

4.1 Component integration

One of the main concerns for the usage of the AUTOSAR development methodology and data exchange formats is the need of the OEMs and suppliers to exchange specification data in a machine-readable, reliable and straightforward way. As the focus of the "‘Specification of Timing Extensions"’ is the facility of requesting a certain timing behavior, this issue will be addressed in the following.

- **Integration**
  Integrating a software component instance delivered by an external party requires the provision of timing data related to this component. As this information can be attached to specific `SwComponentTypes` with regards to its communication partners, the according view `SwcTiming` (see 2.3) will be used. Additionally, specific constraints for implementing this SW-C on Bus view are given, too.

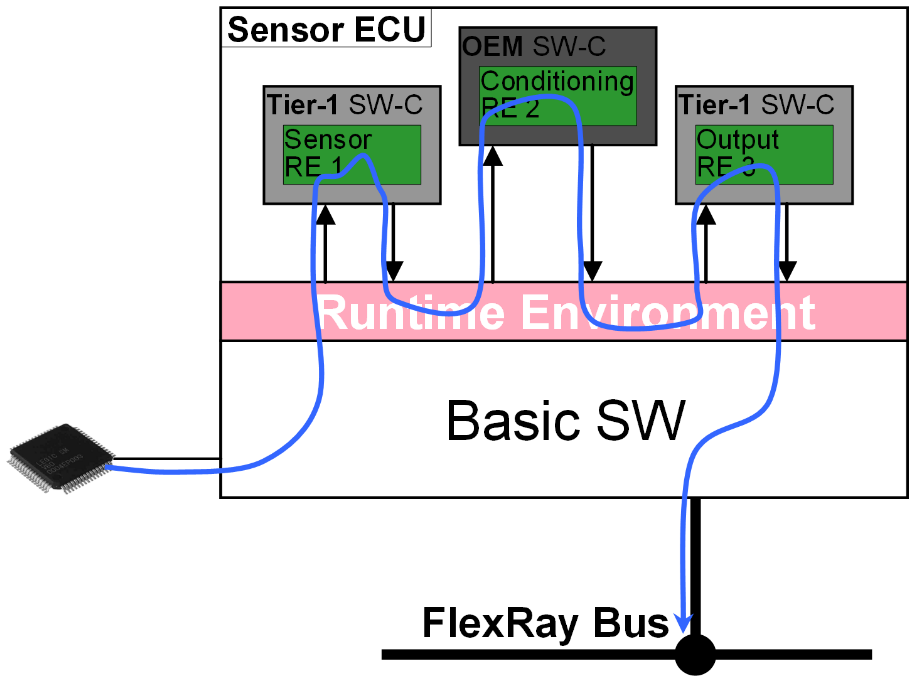


Figure 4.1: FlexRay-based sensor ECU with three software components instances

Figure 4.1 outlines the scenario in a demonstrative way. The shown ECU holds three software components; the first (containing `RunnableEntity` RE1) reads data from a hardware sensor, the second processes signal data conditioning like filtering and averaging. Finally, an Output SW-C converts internal data representations (like 32bit

Float) to ”ready-to-send” data representations (like UInt16 representation). As certain requirements for sensor data conditioning as an input for several other functions within the vehicle exist, the software component "‘Conditioning"’ may be delivered by the OEM, directly. A partial description of these components can be seen in listing 4.1.

```
<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>
    <!-- DataElements for Y- and Z-axis were left out -->
  </DATA-ELEMENTS>
</SENDER-RECEIVER-INTERFACE>
<SENDER-RECEIVER-INTERFACE>
  <SHORT-NAME>OutputSensorData</SHORT-NAME>
  <DATA-ELEMENTS>
    <VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>outValueX</SHORT-NAME>
      <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/datatypes/
         UInt16</TYPE-TREF>
    </VARIABLE-DATA-PROTOTYPE>
  </DATA-ELEMENTS>
</SENDER-RECEIVER-INTERFACE>
<SENSOR-ACTUATOR-SOFTWARE-COMPONENT-TYPE>
  <SHORT-NAME>SensorComponent</SHORT-NAME>
  <PORTS>
    <P-PORT-PROTOTYPE>
      <SHORT-NAME>InternalSensorData</SHORT-NAME>
      <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
         SensorPackage/InternalSensorData</PROVIDED-INTERFACE-TREF>
    </P-PORT-PROTOTYPE>
  </PORTS>
</SENSOR-ACTUATOR-SOFTWARE-COMPONENT-TYPE>
<APPLICATION-SOFTWARE-COMPONENT-TYPE>
  <SHORT-NAME>ConditioningComponent</SHORT-NAME>
  <PORTS>
    <R-PORT-PROTOTYPE>
      <SHORT-NAME>UnprocessedSensorData</SHORT-NAME>
      <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
         SensorPackage/InternalSensorData</REQUIRED-INTERFACE-TREF>
    </R-PORT-PROTOTYPE>
    <P-PORT-PROTOTYPE>
      <SHORT-NAME>ProcessedSensorData</SHORT-NAME>
      <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
         SensorPackage/InternalSensorData</PROVIDED-INTERFACE-TREF>
    </P-PORT-PROTOTYPE>
  </PORTS>
</APPLICATION-SOFTWARE-COMPONENT-TYPE>
<APPLICATION-SOFTWARE-COMPONENT-TYPE>
  <SHORT-NAME>OutputComponent</SHORT-NAME>
  <PORTS>
    <R-PORT-PROTOTYPE>
      <SHORT-NAME>UnprocessedSensorData</SHORT-NAME>
      <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
         SensorPackage/InternalSensorData</REQUIRED-INTERFACE-TREF>
    </R-PORT-PROTOTYPE>
    <P-PORT-PROTOTYPE>
      <SHORT-NAME>ProcessedSensorData</SHORT-NAME>
      <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/
         SensorPackage/OutputSensorData</PROVIDED-INTERFACE-TREF>
    </P-PORT-PROTOTYPE>
  </PORTS>
</APPLICATION-SOFTWARE-COMPONENT-TYPE>
```

Listing 4.1: SWC descriptions and Interface definitions

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

The software component instance "'Conditioning"' 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 instances to this ECU is fixed, already. To fulfill certain non-functional requirements, the implementing `RunnableEntity` of software component instance "'Conditioning"' needs to be executed straight away after `RunnableEntities` of component instance "'Sensor"' and right before `RunnableEntities` of component instance "'Output"'. In addition, the Tier-1 needs information about the execution times he has to await by integrating "'Conditioning"'. Specifying this can be doney by describing the measured (or simulated/estimated ...) execution times of the `RunnableEntity`. The following gives a brief idea how this can be done[1].

4.1.1 VFB view

At first, timing descriptions and constraints on VFB level shall be defined. "'Conditioning"' receives data at its Required Port "'UnprocessedSensorData"' at a certain point in time. This point is denoted by the event "'ConditioningReceived"' whereas "'ConditioningSent"' denotes the sending TDEventVariableDataPrototype of this SW-C. To prescribe a ”maximum age” semantics for the reading input the *LatencyTimingConstraint* is to be used. For this, the external event "'SensorDataProduced"' is to be defined. Based on this, an event chain between this external event and the "'ConditioningReceived"' event is specified. The LatencyTimingConstraint is pointing to this event chain, then. See the representation of the events and the corresponding constraint in listing 4.2.

```
<VFB-TIMING>
  <SHORT-NAME>SensorVFB_Timing</SHORT-NAME>
  <TIMING-CONSTRAINTS>
    <LATENCY-TIMING-CONSTRAINT>
```

[1]To avoid too complex xml representations, several referenced entities are not included.

```
    <LATENCY-CONSTRAINT-TYPE>NO—SAMPLING—EFFECTS</LATENCY-
       CONSTRAINT-TYPE>
    <MAXIMUM>
      <CSE-CODE>1</CSE-CODE>
      <CSE-CODE-FACTOR>50</CSE-CODE-FACTOR>
    </MAXIMUM>
    <MINIMUM>
      <CSE-CODE>1</CSE-CODE>
      <CSE-CODE-FACTOR>40</CSE-CODE-FACTOR>
    </MINIMUM>
    <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingPackage
       /SensorVFB_Timing/InputVFB_Chain</SCOPE-REF>
  </LATENCY-TIMING-CONSTRAINT>
</TIMING-CONSTRAINTS>
<TIMING-DESCRIPTIONS>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>ConditioningReceived</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <VARIABLE-DATA-PROTOTYPE>
      <CONTEXT-PORT-REF DEST="R-PORT-PROTOTYPE">/SensorPackage/
         ConditioningComponent/UnprocessedSensorData</CONTEXT-PORT-
         REF>
      <TARGET-DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/
         SensorPackage/InternalSensorData/internValueX</TARGET-DATA
         -ELEMENT-REF>
    </VARIABLE-DATA-PROTOTYPE>
    <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>
    <VARIABLE-DATA-PROTOTYPE>
      <CONTEXT-PORT-REF DEST="P-PORT-PROTOTYPE">/SensorPackage/
         ConditioningComponent/ProcessedSensorData</CONTEXT-PORT-
         REF>
      <TARGET-DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/
         SensorPackage/InternalSensorData/internValueX</TARGET-DATA
         -ELEMENT-REF>
    </VARIABLE-DATA-PROTOTYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>SensorDataProduced</SHORT-NAME>
    <IS-EXTERNAL>true</IS-EXTERNAL>
    <VARIABLE-DATA-PROTOTYPE>
      <CONTEXT-PORT-REF DEST="R-PORT-PROTOTYPE">/SensorPackage/
         ConditioningComponent/UnprocessedSensorData</CONTEXT-PORT-
         REF>
      <TARGET-DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/
         SensorPackage/InternalSensorData/internValueX</TARGET-DATA
         -ELEMENT-REF>
    </VARIABLE-DATA-PROTOTYPE>
    <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>InputVFB_Chain</SHORT-NAME>
          <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/
             TimingPackage/SensorVFB_Timing/ConditioningReceived</
             RESPONSE-REF>
          <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/
             TimingPackage/SensorVFB_Timing/SensorDataProduced</STIMULUS-
             REF>
        </TIMING-DESCRIPTION-EVENT-CHAIN>
      </TIMING-DESCRIPTIONS>
      <COMPONENT-REF DEST="COMPOSITION-COMPONENT-TYPE">/SensorPackage/
         TopLevelComposition</COMPONENT-REF>
    </VFB-TIMING>
```

Listing 4.2: LatencyConstraint and related events on VfbTiming view

4.1.2 ECU view

After generating the ECU extract implementation related details of the ECU is available. As mentioned before, execution order constraints for the mapped software components (more precise, their such a constrain is related to the respective `RunnableEntities` exist. For the sake of easiness, each software component implements one `RunnableEntity`. Constraining their execution order is to be seen in listing 4.3.

```
  <TIMING-CONSTRAINTS>
    <EXECUTION-ORDER-CONSTRAINT>
      <SHORT-NAME>EOC1</SHORT-NAME>
      <!--prescribes the execution order of SensorRunnable,
         ConditioningRunnable and OutputRunnable -->
      <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">/SensorPackage/
               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">/SensorPackage
                /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-CONSTRAINTS>
    ...
```

Listing 4.3: Execution order constraint for 3 runnable entities

Another typical constraints describe the maximum time to be elapsed for sending data on the bus. Therefore, an TDEventFrame "'DataTransmitted"' representing the point in time the data is sent on the bus needs to be described (listing 4.4. Additionally we describe a `TimingDescriptionEventChain` having "'ConditioningSent"' as `StimulusEvent` and "'DataTransmitted"' as `ResponseEvent` (see listing 4.5).

```
    <TD-EVENT-FRAME>
      <SHORT-NAME>DataTransmitted</SHORT-NAME>
      <FRAME-REF DEST="FRAME">/SystemDescriptionPackage/SensorFrame</FRAME-
          REF>
      <PHYSICAL-CHANNEL-REF DEST="FLEXRAY-PHYSICAL-CHANNEL">/
          SystemDescriptionPackage/SampleFRcluster/FRchannel10MBit</PHYSICAL
          -CHANNEL-REF>
      <TD-EVENT-TYPE>FRAME-TRANSMITTED-ON-BUS</TD-EVENT-TYPE>
    </TD-EVENT-FRAME>
```

Listing 4.4: Event describing the point in time where data is sent on the bus

```
      <TIMING-DESCRIPTION-EVENT-CHAIN>
        <SHORT-NAME>SensorECU_Chain</SHORT-NAME>
        <RESPONSE-REF DEST="TD-EVENT-FRAME">/TimingPackage/SensorECU_Timing
            /DataTransmitted</RESPONSE-REF>
        <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/
            TimingPackage/SensorVFB_Timing/ConditioningSent</STIMULUS-REF>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
```

Listing 4.5: Event chain describing the sending path of data

The resulting constraint prescribing the maximum latency from Stimulus to Response of this event chain can be seen by means of listing 4.6.

```
      <LATENCY-TIMING-CONSTRAINT>
        <LATENCY-CONSTRAINT-TYPE>LAST-STIMULUS-FIRST-RESPONSE</LATENCY-
            CONSTRAINT-TYPE>
          <MAXIMUM>
            <CSE-CODE>1</CSE-CODE>
            <CSE-CODE-FACTOR>400</CSE-CODE-FACTOR>
          </MAXIMUM>
          <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingPackage/
              SensorECU_Timing/SensorECU_Chain</SCOPE-REF>
        </LATENCY-TIMING-CONSTRAINT>
      </TIMING-CONSTRAINTS>
```

Listing 4.6: Latency constraint prescribing the maximum latency of sending path within the ECU

4.2 Engine control

This example should illustrate an example for the definition of timing constraints in an engine mangement system. Although the system is simplified to be included within the application notes chapter it is based on a real world example in it's basic concepts.

4.2.1 Overview

The example system is an air mass controlled gasoline engine control system. Roughly, the functionality of software components can be separated in the following categories:

- **Sensor preprocessing:** Three `SensorSwComponentType`s *MassAirFlowSensorSWC*, *AcceleratorPedalSensorSWC*, and *ThrottleSensorSWC* are responsible for reading in the most important control factors.
- **Application based calculation:** Most `ApplicationSwComponentType`s calculate the new control factors for the engine. In summary these components are *AcceleratorPedalVoterSWC, ThrottleControllerSWC, ThrottleChangeSWC, BaseFuelMassSWC, TransientFuelMassSWC, IgnitionSWC,* and *TotalFuelMassSWC*.
- **Actuators:** The control of the actuators is encapsulated by the `ActuatorSwComponentType`s *ThrottleActuatorSWC, InjectionActuatorSWC,* and *IgnitionAcutatorSWC*.
- **Engine mode and control:** The motor can be operated in different operation modes. The `AtomicSwComponentType` *OperatingModeSWC* includes a state machine which decides what setting for the application based calculation is used depending on the current mode (e.g. normal drive, idle speed, ...). Similar values are delivered by the *IdleSpeedControlSWC* which determines important inputs for application calculation during idle speed.
- **Misc:** The `AtomicSwComponentType` *InjBatVoltCorrectionSensorSWC* 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.

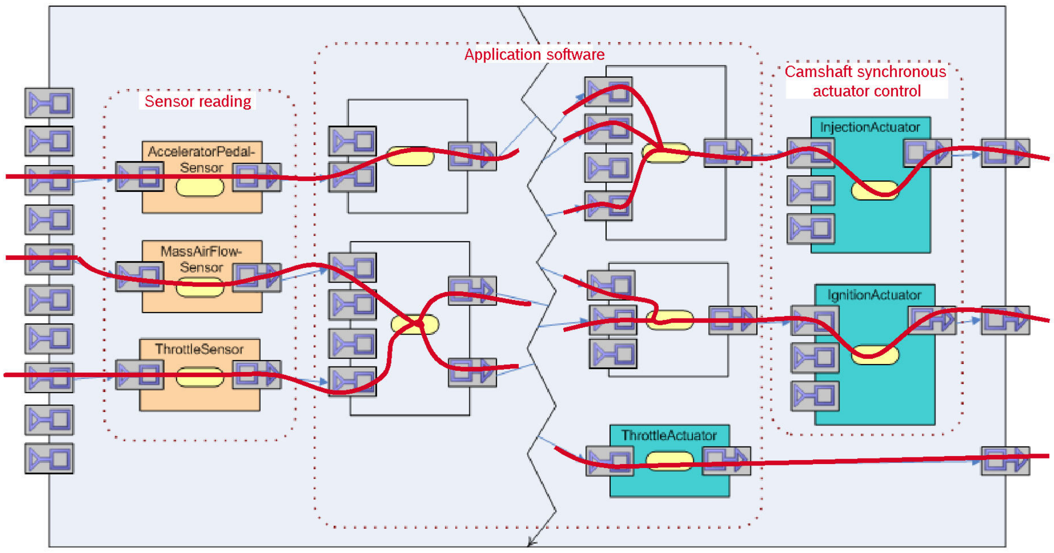


Figure 4.2: Overview of cause and effect chains in a engine control example

Since giving a complete overview of the system would result in an highly connected graph, figure 4.2 shows a simplified view of the `System`. The red lines show important signal paths that are considered to be important for timing analysis.

Let's assume that on the application level the following timing requirements can be determined:

4.2.2 Timing Requirements

1. Beginning from the request on the accelarator pedal a reaction on the throttle actuator must be initiated within 30ms.
2. The application software requires a change in the position of the throttle angle with a maximum delay of 10ms.
3. Beginning from the request on the accelarator pedal an effect of injection mass and ingnition must have a delay of 50ms.
4. Starting from the camshaft synchronous `BswInterruptEntity` the ignition time must be calculated within 3ms.
5. Each cylinder must intiate and finish injection mass calculation within less than 20ms.

The first task is to break down the timing requirements into a formal description in the AUTOSAR Timing specification.

4.2.3 Formal description of timing constraints on VFB-level

It must be clear that the requirements from section 4.2.2 can be mapped to timing constraints that reference different parts of the system. Sine an overview of the whole system would be too detailed for each requirement a snapshot of relevant components, events and event-chains is given.

The requirements 1-3 can be expressed on the VFB-level.

4.2.3.1 Requirement: 1

Figure 4.3 shows the simplified signal flow and involved components. It has been identified that the critical path of execution will effect four components. The component *AccelleratorPedalSWC* is responsible for reading in the signal and passing it to the application software component *AccelleratorPedalVoterSWC* . Afterwards the processed signal is further processed in the application software component *ThrottleController-SWC* until it is finally send to the actuator in the actuator software component *ThrottleActuatorSWC*. For specification of the constraint the a timing description chain must be defined along the involved interface elements (ports). The names are given in the XML-specification and cover the whole path from the sensor to the actuator.

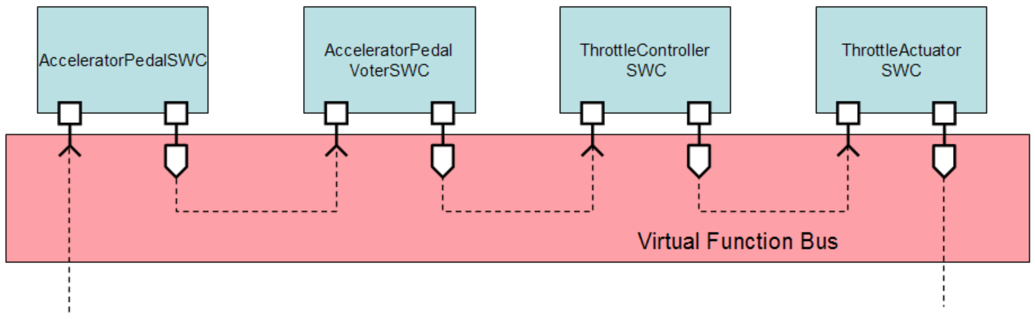


Figure 4.3: Involved components for signal flow from AccelleratorPedalSWC to Throttle-ActuatorSWC for Timing Requirement 1.

Since we are dealing with a sensor-actuator communication the chosen constraint is a SENSOR-ACTUATOR-LATENCY-CONSTRAINT. Also note that the overall timing chain references all important sub-chains.

```
<AR:ELEMENTS>

<!-- VFB Timing -->
<AR:VFB-TIMING>
<AR:SHORT-NAME>EngineControVFBTiming</AR:SHORT-NAME>
<AR:TIMING-DESCRIPTIONS>


 <!-- ============== Requirement 1 ================= -->
 <!-- Events for Requirement 1 -->
 <AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
```

```
<AR:SHORT-NAME>RAcceleratorPedalPositionsSensor</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="R-PORT-PROTOTYPE">RAcceleratorPedalPositionsSensor
    </AR:ATP-TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE–DATA–PROTOTYPE–RECEIVED<
    /AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
<AR:IS-EXTERNAL></AR:IS-EXTERNAL>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>PAceleratorPedalPositionsSent</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="P-PORT-PROTOTYPE">PAceleratorPedalPositions</
    AR:ATP-TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE–DATA–PROTOTYPE–SENT</
    AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>RAceleratorPedalPositionsReceived</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="R-PORT-PROTOTYPE">RAceleratorPedalPositions</
    AR:ATP-TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE–DATA–PROTOTYPE–RECEIVED<
    /AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>PVotedPedalPositionSent</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="P-PORT-PROTOTYPE">PVotedPedalPosition</AR:ATP-
    TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE–DATA–PROTOTYPE–SENT</
    AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>RVotedPedalPositionReceived</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="R-PORT-PROTOTYPE">RVotedPedalPosition</AR:ATP-
    TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE–DATA–PROTOTYPE–RECEIVED<
    /AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>PUnlimitedThrottlePosSent</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="P-PORT-PROTOTYPE">PUnlimitedThrottlePos</AR:ATP-
    TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
```

```
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE−DATA−PROTOTYPE−SENT</
    AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>RUnlimitedThrottlePosReceived</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="R-PORT-PROTOTYPE">RUnlimitedThrottlePos</AR:ATP-
    TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE−DATA−PROTOTYPE−RECEIVED<
    /AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>PUnlimitedThrottlePosAtActuator</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="P-PORT-PROTOTYPE">PUnlimitedThrottlePosActuator</
    AR:ATP-TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE−DATA−PROTOTYPE−SENT</
    AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
<AR:IS-EXTERNAL></AR:IS-EXTERNAL>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<!-- Timing Chain for Requirement 1 -->

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement1Seg0</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    RAcceleratorPedalPositionsSensor</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    PAceleratorPedalPositions</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement1Seg1</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    PAceleratorPedalPositions</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    RAceleratorPedalPositionsReceived</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement1Seg1_1</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    RAceleratorPedalPositionsReceived</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    VotedPedalPositionSent</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement1Seg2</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    VotedPedalPositionSent</AR:STIMULUS-REF>
```

```
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   RVotedPedalPositionReceived</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement1Seg2_1</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   RVotedPedalPositionReceived</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   PUnlimitedThrottlePosSent</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement1Seg3</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   PUnlimitedThrottlePosSent</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   RUnlimitedThrottlePosReceived</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement1Seg4</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   RUnlimitedThrottlePosReceived</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   PUnlimitedThrottlePosAtActuator</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>


<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement1AllSeg</AR:SHORT-NAME>
<AR:SEGMENT-REFS>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement1Seg0</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement1Seg1</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement1Seg1_1</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement1Seg2</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement1Seg2_1</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement1Seg3</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement1Seg4</AR:SEGMENT-REF>
</AR:SEGMENT-REFS>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

</AR:TIMING-DESCRIPTIONS>

<AR:TIMING-CONSTRAINTS>
<!-- ============= Requirement 1 ================ -->
<!-- Timing constraint for Requirement 1 -->
<AR:SENSOR-ACTUATOR-LATENCY-CONSTRAINT>
```

```
<AR:LATENCY-CONSTRAINT-TYPE>FIRST-STIMULUS-FIRST-RESPONSE</AR:LATENCY-
    CONSTRAINT-TYPE>
<AR:SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
    TimingChainRequirement1AllSeg</AR:SCOPE-REF>
<AR:MAXIMUM><AR:CSE-CODE>3</AR:CSE-CODE><AR:CSE-CODE-FACTOR>30</AR:CSE-CODE
    -FACTOR></AR:MAXIMUM>
<AR:NOMINAL><AR:CSE-CODE>3</AR:CSE-CODE><AR:CSE-CODE-FACTOR>30</AR:CSE-CODE
    -FACTOR></AR:NOMINAL>
</AR:SENSOR-ACTUATOR-LATENCY-CONSTRAINT>
</AR:TIMING-CONSTRAINTS>

</AR:VFB-TIMING>
```

Listing 4.7: Event definition and constraints for Requirement 1

4.2.3.2 Requirement: 2

Requirement 2 specifies a typical constraint concerning sensor ages. For calculation in the `AtomicSwComponentType` *BaseFuelMassSWC* (which is here chosen as an example of the application software) a maximum age of input data concerning the ThrottleAngle must be guaranteed. The sensor value is determined in the `SensorSwComponentType` *ThrottleSensorSWC* and is passed to the application software. Figure 4.4 shows the involved components.

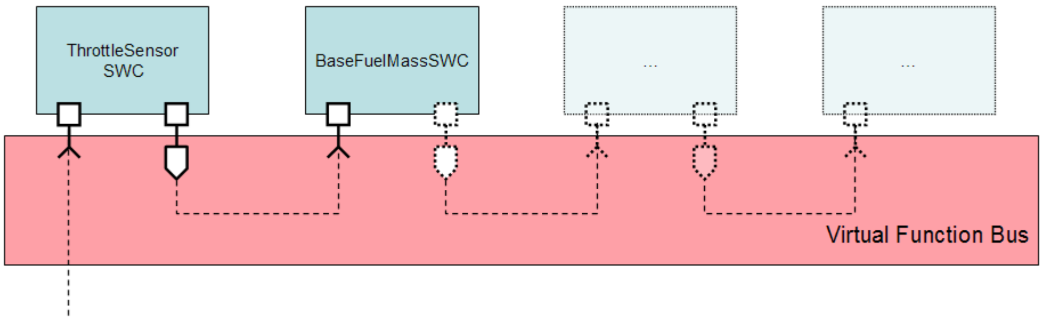


Figure 4.4: Involved components for signal flow from ThrottleSensorSWC to application components (Here BaseFuelMassSWC) for Timing Requirement 2.

Please note that even if the signal flow continous to other parts of the system, it is possible to exactly specify only this aspect of the desired timing behavior.

```
<!-- VFB Timing -->
<AR:VFB-TIMING>
<AR:SHORT-NAME>EngineControVFBTiming</AR:SHORT-NAME>
<AR:TIMING-DESCRIPTIONS>

 <!-- ============== Requirement 2 ================= -->
 <!-- Events for Requirement 2 -->
 <AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>RThrottlePositionSensor</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
```

```
<AR:ATP-TARGET-REF DEST="R-PORT-PROTOTYPE">RThrottlePositionSensor</AR:ATP-
    TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE—DATA—PROTOTYPE—RECEIVED<
    /AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
<AR:IS-EXTERNAL></AR:IS-EXTERNAL>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>PThrottlePositionSent</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="P-PORT-PROTOTYPE">PThrottlePosition</AR:ATP-TARGET
    -REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE—DATA—PROTOTYPE—SENT</
    AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>


<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>RThrottlePositionReceived</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="R-PORT-PROTOTYPE">RThrottlePosition</AR:ATP-TARGET
    -REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE—DATA—PROTOTYPE—SENT</
    AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<!-- Timing Chain for Requirement 2 -->
<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement2Seg0</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    RThrottlePositionSensor</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    PThrottlePositionSent</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement2Seg1</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    PThrottlePositionSent</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    RThrottlePositionReceived</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement2AllSeg</AR:SHORT-NAME>
<AR:SEGMENT-REFS>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
    TimingChainRequirement2Seg0</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
    TimingChainRequirement2Seg1</AR:SEGMENT-REF>
</AR:SEGMENT-REFS>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>
</AR:TIMING-DESCRIPTIONS>
```

```
<AR:TIMING-CONSTRAINTS>

<!-- ============= Requirement 2 ================ -->
<!-- Timing constraint for Requirement 2 -->
<AR:SENSOR-ACTUATOR-LATENCY-CONSTRAINT>
<AR:LATENCY-CONSTRAINT-TYPE>FIRST-STIMULUS-FIRST-RESPONSE</AR:LATENCY-
    CONSTRAINT-TYPE>
<AR:SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
    TimingChainRequirement2Seg1</AR:SCOPE-REF>
<AR:MAXIMUM><AR:CSE-CODE>3</AR:CSE-CODE><AR:CSE-CODE-FACTOR>10</AR:CSE-CODE
    -FACTOR></AR:MAXIMUM>
</AR:SENSOR-ACTUATOR-LATENCY-CONSTRAINT>
</AR:TIMING-CONSTRAINTS>

</AR:VFB-TIMING>
```

Listing 4.8: Event definition and constraints for Requirement 2

4.2.3.3 Requirement: 3

In requirement 3 a very complex `TimingDescriptionEventChain` is constraint. The first part of the chain is already defined in the context of requirement 1. Thus, we can reference to the set of defined events as well as to the predefined timing event chains. The second part of the chain captures the feedback in the system that will be observed reading out the sensors. Please note that all events must have a functional dependency, so it is important to understand that the `SensorActuatorSwComponentType` *ThrottleSensorSWC* must wait for the response of the output of `SensorActuatorSwComponentType` *ThrottleActuatorSWC*. Figure 4.5 shows the complete timing event chain.

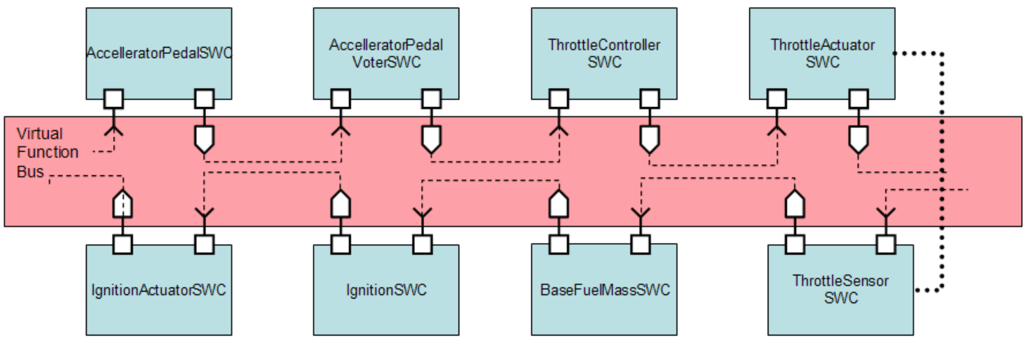


Figure 4.5: for Timing Requirement 3.

```
<!-- VFB Timing -->
<AR:VFB-TIMING>
<AR:SHORT-NAME>EngineControVFBTiming</AR:SHORT-NAME>
<AR:TIMING-DESCRIPTIONS>

<!-- ============== Requirement 3 ================= -->
```

```
<!-- Events for Requirement 3 -->
<!-- Only those events must be defined that are not already used in
    Requirement 1+2 -->
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>PMafRateOutSent</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="P-PORT-PROTOTYPE">PMafRateOut</AR:ATP-TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE—DATA—PROTOTYPE—SENT</
    AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>RMafRateOutReceived</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="R-PORT-PROTOTYPE">RMafRateOut</AR:ATP-TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE—DATA—PROTOTYPE—RECEIVED<
    /AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>PIgnitionTimingSent</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="P-PORT-PROTOTYPE">PIgnitionTimingSent</AR:ATP-
    TARGET-REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE—DATA—PROTOTYPE—SENT</
    AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>
<AR:SHORT-NAME>RIgnitionTimingReceived</AR:SHORT-NAME>
<AR:VARIABLE-DATA-PROTOTYPE>
<AR:ATP-TARGET-REF DEST="R-PORT-PROTOTYPE">RIgnitionTiming</AR:ATP-TARGET-
    REF>
</AR:VARIABLE-DATA-PROTOTYPE>
<AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE—DATA—PROTOTYPE—RECEIVED<
    /AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
</AR:TD-EVENT-VARIABLE-DATA-PROTOTYPE>

<!-- Timing Chain for Requirement 3 -->
<!-- Only additional timing chains must be defined that are not part of
    requirement 2 + 3 -->
<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement3Seg0</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    RUnlimitedThrottlePosReceived</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">PMafRateOutSent</
    AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement3Seg1</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">PMafRateOutSent</
    AR:STIMULUS-REF>
```

```
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   RMafRateOutReceived</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement3Seg1_1</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   RMafRateOutReceived</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   PIgnitionTimingSent</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>


<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement3Seg2</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   PIgnitionTimingSent</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   RIgnitionTimingReceived</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

<!-- Now define overall chain -->

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement3AllSeg</AR:SHORT-NAME>
<AR:SEGMENT-REFS>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement1AllSeg</AR:SEGMENT-REF>
<!-- Note that a reaction of the real system must be considered in between
   (Segment 0!) -->
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement2Seg0</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement2Seg1</AR:SEGMENT-REF>
<!-- Now continue in chain 3 -->
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement3Seg0</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement3Seg1</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement3Seg1_1</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement3Seg2</AR:SEGMENT-REF>
</AR:SEGMENT-REFS>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>


</AR:TIMING-DESCRIPTIONS>


<AR:TIMING-CONSTRAINTS>

<!-- ============= Requirement 3 ================ -->
<!-- Timing constraint for Requirement 3 -->
<AR:SENSOR-ACTUATOR-LATENCY-CONSTRAINT>
```

```
<AR:LATENCY-CONSTRAINT-TYPE>FIRST−STIMULUS−FIRST−RESPONSE</AR:LATENCY-
   CONSTRAINT-TYPE>
<AR:SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement3AllSeg</AR:SCOPE-REF>
<AR:MAXIMUM><AR:CSE-CODE>3</AR:CSE-CODE><AR:CSE-CODE-FACTOR>50</AR:CSE-CODE
   -FACTOR></AR:MAXIMUM>
</AR:SENSOR-ACTUATOR-LATENCY-CONSTRAINT>

<!-- Define Offset constraint for actuator/sensor reaction -->
<AR:OFFSET-TIMING-CONSTRAINT>
<AR:SOURCE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   PUnlimitedThrottlePosAtActuator</AR:SOURCE-REF>
<AR:TARGET-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
   RThrottlePositionSensor</AR:TARGET-REF>
<AR:MAXIMUM><AR:CSE-CODE>3</AR:CSE-CODE><AR:CSE-CODE-FACTOR>5</AR:CSE-CODE-
   FACTOR></AR:MAXIMUM>

</AR:OFFSET-TIMING-CONSTRAINT>

</AR:TIMING-CONSTRAINTS>
</AR:VFB-TIMING>
```

Listing 4.9: Event definition and constraints for Requirement 3

4.2.4 Formal description of timing constraints on ECU-level

Since requirement 4 references to events that are caused by basic software modules (namely the interrupt system), the events must be defined in the scope of the ECU.

4.2.4.1 Requirement: 4

The timing event chain for the requirements starts with the activation of the `BswInterruptEntity` in the basic software module. It is assumed here that the `BswModuleEntity` invokes the IgnitionActuatorSWC's `RunnableEntity`. The timing event chain finishes with the finalization of calculations when the IgnitionActuatorSWC's `RunnableEntity` terminates.

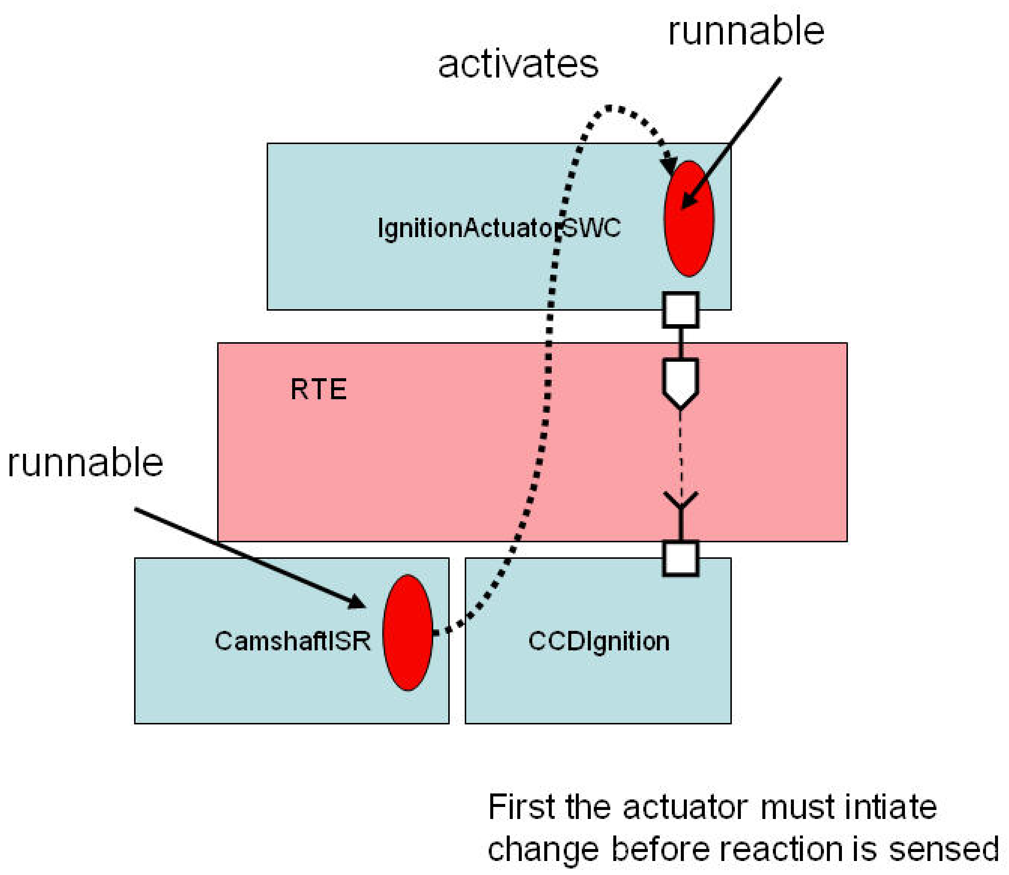


Figure 4.6: for Timing Requirement 4.

```
<AR:ECU-TIMING>
<AR:SHORT-NAME>EngineControECUTiming</AR:SHORT-NAME>
<AR:TIMING-DESCRIPTIONS>
<!-- ============== Requirement 4 ================= -->
 <!-- Events for Requirement 4 -->
<AR:TD-EVENT-BSW-BEHAVIOR>
<AR:SHORT-NAME>CamShaftISRStart</AR:SHORT-NAME>
<AR:BSW-MODULE-ENTITY-REF DEST="BSW-INTERRUPT-ENTITY">CamShaftISR</AR:BSW-
    MODULE-ENTITY-REF>
<AR:TD-EVENT-BSW-BEHAVIOR-TYPE>BSW—MODULE—ENTITY—ACTIVATED</AR:TD-EVENT-BSW
    -BEHAVIOR-TYPE>
</AR:TD-EVENT-BSW-BEHAVIOR>

<AR:TD-EVENT-BSW-BEHAVIOR>
<AR:SHORT-NAME>CamShaftISRTerminate</AR:SHORT-NAME>
<AR:BSW-MODULE-ENTITY-REF DEST="BSW-INTERRUPT-ENTITY">CamShaftISR</AR:BSW-
    MODULE-ENTITY-REF>
<AR:TD-EVENT-BSW-BEHAVIOR-TYPE>BSW—MODULE—ENTITY—TERMINATED</AR:TD-EVENT-
    BSW-BEHAVIOR-TYPE>
</AR:TD-EVENT-BSW-BEHAVIOR>

<AR:TD-EVENT-INTERNAL-BEHAVIOR>
<AR:SHORT-NAME>IgnitionActuatorCalcStart</AR:SHORT-NAME>
```

```
<AR:RUNNABLE-REF DEST="RUNNABLE-ENTITY">IgnitionActuatorCalculation</
    AR:RUNNABLE-REF>
<AR:TD-EVENT-INTERNAL-BEHAVIOR-TYPE>RUNNABLE—ENTITY—ACTIVATED</AR:TD-EVENT-
    INTERNAL-BEHAVIOR-TYPE>
</AR:TD-EVENT-INTERNAL-BEHAVIOR>

<AR:TD-EVENT-INTERNAL-BEHAVIOR>
<AR:SHORT-NAME>IgnitionActuatorCalcTerminated</AR:SHORT-NAME>
<AR:RUNNABLE-REF DEST="RUNNABLE-ENTITY">IgnitionActuatorCalculation</
    AR:RUNNABLE-REF>
<AR:TD-EVENT-INTERNAL-BEHAVIOR-TYPE>RUNNABLE—ENTITY—TERMINATED</AR:TD-EVENT
    -INTERNAL-BEHAVIOR-TYPE>
</AR:TD-EVENT-INTERNAL-BEHAVIOR>

<!-- Event chains for Requirement 4 -->

<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement4Seg1</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">CamShaftISRStart</
    AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    CamShaftISRTerminate</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>


<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement4Seg2</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    CamShaftISRTerminate</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    IgnitionActuatorCalcStart</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>


<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement4Seg3</AR:SHORT-NAME>
<AR:STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    IgnitionActuatorCalcStart</AR:STIMULUS-REF>
<AR:RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">
    IgnitionActuatorCalcTerminated</AR:RESPONSE-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>


<!-- Overall chain for Requirement 4 -->
<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement4AllSeg</AR:SHORT-NAME>
<AR:SEGMENT-REFS>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
    TimingChainRequirement4Seg1</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
    TimingChainRequirement4Seg2</AR:SEGMENT-REF>
<AR:SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
    TimingChainRequirement4Seg3</AR:SEGMENT-REF>
</AR:SEGMENT-REFS>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>
```

```
</AR:TIMING-DESCRIPTIONS>

<AR:TIMING-CONSTRAINTS>
<!-- ============= Requirement 4 ================ -->
<!-- Timing constraint for Requirement 4 -->
<AR:LATENCY-TIMING-CONSTRAINT>
<AR:LATENCY-CONSTRAINT-TYPE>NO-SAMPLING-EFFECTS</AR:LATENCY-CONSTRAINT-TYPE
   >
<AR:SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">
   TimingChainRequirement4All</AR:SCOPE-REF>
<AR:MAXIMUM><AR:CSE-CODE>3</AR:CSE-CODE><AR:CSE-CODE-FACTOR>3</AR:CSE-CODE-
   FACTOR></AR:MAXIMUM>
</AR:LATENCY-TIMING-CONSTRAINT>

</AR:TIMING-CONSTRAINTS>
</AR:ECU-TIMING>
```

Listing 4.10: Event definition and constraints for Requirement 4

4.2.5 Formal description of timing constraints on SWC-level

Requirement 5 refers to execution behavior of one software component's `RunnableEntity` so the scope will be set to SWC-level.

4.2.5.1 Requirement: 5

The SWC-level references to behavior of `RunnableEntity` of SWCs. Here we only reference the `RunnableEntity` of the SWC *IgnitionSWC*. Basicly we want to ensure that 1. the delay between activation an termination of the `RunnableEntity` is <= 20 ms and 2. the `RunnableEntity` is triggered with a frequency of maximal 20 ms. The second part of the constraint is a good example of a constraint that could be also regarded as a guarantee.

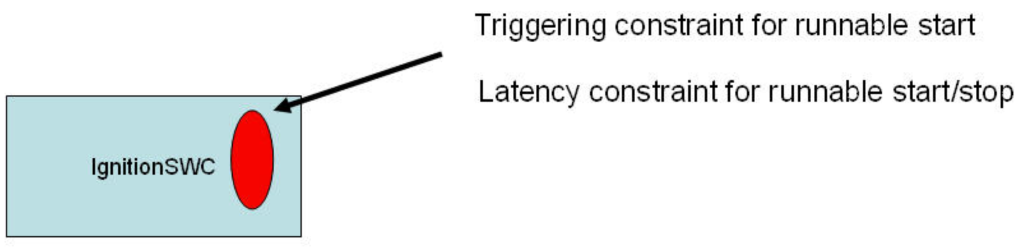

Triggering constraint for runnable start
Latency constraint for runnable start/stop
IgnitionSWC

Figure 4.7: for Timing Requirement 5.

```
<AR:SWC-TIMING>
<AR:SHORT-NAME>EngineControSWCTiming</AR:SHORT-NAME>

<AR:TIMING-DESCRIPTIONS>
```

```
<!-- ============== Requirement 5 ================= -->
 <!-- Events for Requirement 5 -->
<AR:TD-EVENT-INTERNAL-BEHAVIOR>
<AR:SHORT-NAME>ActivateCalculationInjectionMass</AR:SHORT-NAME>
<AR:RUNNABLE-REF DEST="RUNNABLE-ENTITY">InjectionMassCalculationProc</
   AR:RUNNABLE-REF>
<AR:TD-EVENT-INTERNAL-BEHAVIOR-TYPE>RUNNABLE—ENTITY—ACTIVATED</AR:TD-EVENT-
   INTERNAL-BEHAVIOR-TYPE>
</AR:TD-EVENT-INTERNAL-BEHAVIOR>

<AR:TD-EVENT-INTERNAL-BEHAVIOR>
<AR:SHORT-NAME>TerminateCalculationInjectionMass</AR:SHORT-NAME>
<AR:RUNNABLE-REF DEST="RUNNABLE-ENTITY">InjectionMassCalculationProc</
   AR:RUNNABLE-REF>
<AR:TD-EVENT-INTERNAL-BEHAVIOR-TYPE>RUNNABLE—ENTITY—TERMINATED</AR:TD-EVENT
   -INTERNAL-BEHAVIOR-TYPE>
</AR:TD-EVENT-INTERNAL-BEHAVIOR>

<!-- Timing Chain for Requirement 5 -->
<!-- Only additional timing chains must be defined that are not part of
   requirement 2 + 3 -->
<AR:TIMING-DESCRIPTION-EVENT-CHAIN>
<AR:SHORT-NAME>TimingChainRequirement5</AR:SHORT-NAME>
<AR:RESPONSE-REF DEST="TD-EVENT-INTERNAL-BEHAVIOR">
   TerminateCalculationInjectionMass</AR:RESPONSE-REF>
<AR:STIMULUS-REF DEST="TD-EVENT-INTERNAL-BEHAVIOR">
   ActivateCalculationInjectionMass</AR:STIMULUS-REF>
</AR:TIMING-DESCRIPTION-EVENT-CHAIN>

</AR:TIMING-DESCRIPTIONS>


<!-- ============= Requirement 5 ================ -->
<!-- Timing constraints for Requirement 5 -->

<AR:TIMING-CONSTRAINTS>

<AR:LATENCY-TIMING-CONSTRAINT>
<AR:LATENCY-CONSTRAINT-TYPE>NO—SAMPLING—EFFECTS</AR:LATENCY-CONSTRAINT-TYPE
   >
<AR:SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">TimingChainRequirement5
   </AR:SCOPE-REF>
<AR:MAXIMUM><AR:CSE-CODE>3</AR:CSE-CODE><AR:CSE-CODE-FACTOR>20</AR:CSE-CODE
   -FACTOR></AR:MAXIMUM>
</AR:LATENCY-TIMING-CONSTRAINT>

<AR:SPORADIC-EVENT-TRIGGERING>
<AR:EVENT-REF DEST="TD-EVENT-BSW">ActivateCalculationInjectionMass</
   AR:EVENT-REF>
<AR:MAXIMUM-INTER-ARRIVAL-TIME><AR:CSE-CODE>3</AR:CSE-CODE><AR:CSE-CODE-
   FACTOR>20</AR:CSE-CODE-FACTOR></AR:MAXIMUM-INTER-ARRIVAL-TIME>
<AR:PERIOD><AR:CSE-CODE>3</AR:CSE-CODE><AR:CSE-CODE-FACTOR>20</AR:CSE-CODE-
   FACTOR></AR:PERIOD>
</AR:SPORADIC-EVENT-TRIGGERING>
```

```
</AR:TIMING-CONSTRAINTS>

</AR:SWC-TIMING>
```

Listing 4.11: Event definition and constraints for Requirement 5

4.3 Describing and Constraining Sensor and Actuator Timing

Chapter 3.2.1 describes the specification of VFB events and introduces the attribute "‘isExternal"’ of such events. If the attribute is set to TRUE, then the event is considered to be external, i.e. on the hardware IO of a `SensorActuatorSoftwareComponent` (respectively `ComplexDeviceDriverComponent`). In this chapter we describe how this attribute is used to describe EventChains for sensor and actuator timing, how the different events of such kind of chain relate to each other and how the timing can be constrained using `TimingConstraints`.

The Timing Extensions aim at the specification of end-to-end constraints also in early development phases. However, on VFB-level there exist no such things as sensors, actuators or other hardware elements to attach events to. Therefore, certain VFB events can be used to declare external events. For sensor and actuator timing four cases can be distinguished: external events can be observed at sensors and actuators, which in turn can access the hardware via a sender/receiver or client/server interface.

4.3.1 External Event of a Sensor accessed via S/R

In this case the `SensorActuatorSoftwareComponent` receives data from the `EcuAbstractionComponent` through a S/R interface on its receiver port. On that port there can exist two `TDEventVariableDataPrototypes` for each `Variable-DataPrototype` with the enum type `variableDataPrototypeReceived`, one of which with isExternal set to TRUE.

The semantics of the external event is that it occurs at the hardware IO. The not external event occurs when the data is received by the `SensorActuatorSoftware-Component`. Therefore the external event occurs at any time before the not external one.

4.3.2 External Event of an Actuator accessed via S/R

In this case the `SensorActuatorSoftwareComponent` sends data to the `EcuAb-stractionComponent` through a S/R interface on its sender port. On that port there can exist two `TDEventVariableDataPrototypes` for each `VariableDataPro-totype` with the enum type `variableDataPrototypeSent`, one of which with isExternal set to TRUE.

The not external event occurs when the data is sent by the `SensorActuatorSoftwareComponent`. The external event occurs at any time after the not external one.

4.3.3 External Event of a Sensor accessed via C/S

In this case the `SensorActuatorSoftwareComponent` receives data from the `EcuAbstractionComponent` through a C/S interface on its receiver port. On that port there can exist two `TDEventOperations` for each `ClientServerOperation` with the enum type `operationCallResponseReceived`, one of which with isExternal set to TRUE.

The not external event occurs when the response is received by the `SensorActuatorSoftwareComponent`. The external event occurs at any time before the not external one.

4.3.4 External Event of an Actuator accessed via C/S

In this case the `SensorActuatorSoftwareComponent` sends data to the `EcuAbstractionComponent` through a C/S interface on its sender port. On that port there can exist two `TDEventOperations` for each `ClientServerOperation` with the enum type `operationCalled`, one of which with isExternal set to TRUE.

The not external event occurs when the operation is called by the `SensorActuatorSoftwareComponent`. The external event occurs at any time after the not external one.

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

To express an end-to-end sensor or actuator `EventChain` 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 `EventChain` 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 `SensorActuatorSoftwareComponent`. The input latency is the time between the two events described in 4.3.1 and 4.3.3, 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 `SensorActuatorSoftwareComponent`, 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 4.3.2 and 4.3.4, respectively, depending on the communication type.

4.3.6 Constraining Input or Output Latency

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

A History of Constraints and Specification Items

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

A.1.1 Changed Constraints in R4.0.1

No constraints were changed in this release.

A.1.2 Added Constraints in R4.0.1

No constraints were added in this release.

A.1.3 Deleted Constraints in R4.0.1

No constraints were deleted in this release.

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

A.2.1 Changed Constraints in R4.0.2

No constraints were changed in this release.

A.2.2 Added Constraints in R4.0.2

No constraints were added in this release.

A.2.3 Deleted Constraints in R4.0.2

No constraints were deleted in this release.

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

A.3.1 Changed Constraints in R4.0.3

No constraints were changed in this release.

A.3.2 Added Constraints in R4.0.3

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

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

Table A.1: Added Constraints in R4.0.3

A.3.3 Deleted Constraints in R 4.0.3

No constraints were changed in this release.

A.4 Added Specification Items in R4.0.3

Number	Heading
[TPS_TIMEX_0001]	Purpose of `TimingDescriptionEvent`
[TPS_TIMEX_0002]	Purpose of `TimingDescriptionEventChain`
[TPS_TIMEX_0003]	`EventTriggeringConstraint` specifies occurrence behavior respectively model
[TPS_TIMEX_0004]	`LatencyTimingConstraint` specifies latency constraints
[TPS_TIMEX_0005]	`AgeConstraint` to specify age constraints
[TPS_TIMEX_0006]	`SynchronizationTimingConstraint` specifies synchronicity constraints
[TPS_TIMEX_0007]	`ExecutionOrderConstraint` specifies sequence of executing executable entities
[TPS_TIMEX_0008]	`ExecutionTimeConstraint` to specify execution time constraints
[TPS_TIMEX_0009]	Optional use of timing extensions
[TPS_TIMEX_0010]	`PeriodicEventTriggering` specifies periodic occurrences of events
[TPS_TIMEX_0011]	`SporadicEventTriggering` specifies sporadic occurrences of events
[TPS_TIMEX_0012]	`ConcretePatternEventTriggering` specifies concrete pattern of occurrences of events
[TPS_TIMEX_0013]	`BurstPatternEventTriggering` specifies burst of occurrences of events

[TPS_TIMEX_0014]	ArbitraryEventTriggering specifies arbitrary occurrences of an event
[TPS_TIMEX_0015]	OffsetTimingConstraint specifies offset between occurrences of events
[TPS_TIMEX_0016]	Purpose of TDEventVfb
[TPS_TIMEX_0017]	TDEventVariableDataPrototype specifies events observable at sender/receiver ports
[TPS_TIMEX_0018]	TDEventOperation specifies events obeservable at client/server ports.
[TPS_TIMEX_0019]	TDEventModeDeclaration specifies events obeservable at mode ports.
[TPS_TIMEX_0020]	TDEventSwcInternalBehavior specifies observable events of runnable entities
[TPS_TIMEX_0021]	Purpose of TDEventCom
[TPS_TIMEX_0022]	TDEventISignal specifies events related to the exchange of I-Signals
[TPS_TIMEX_0023]	TDEventIPdu specifies events related to the exchange of I-PDUs
[TPS_TIMEX_0024]	TDEventFrame specifies events related to the exchange of network frames
[TPS_TIMEX_0025]	TDEventFrClusterCycleStart specifies the event related to the start of a FlexRay communication cycle
[TPS_TIMEX_0026]	TDEventTTCanCycleStart specifies the event related to the start of a TTCAN communication cycle
[TPS_TIMEX_0027]	Purpose of TDEventComplex
[TPS_TIMEX_0028]	TDEventBswInternalBehavior specifies observable events of BSW module entities
[TPS_TIMEX_0029]	Purpose of TDEventBsw
[TPS_TIMEX_0030]	TDEventBswModule specifies observable events when basic software entries are called
[TPS_TIMEX_0031]	TDEventBswModeDeclaration specifies observable events in case of BSW mode communication
[TPS_TIMEX_0032]	Purpose of VfbTiming
[TPS_TIMEX_0033]	Purpose of SwcTiming
[TPS_TIMEX_0034]	Purpose of SystemTiming
[TPS_TIMEX_0035]	Purpose of BswModuleTiming
[TPS_TIMEX_0036]	Purpose of EcuTiming
[TPS_TIMEX_0037]	TimingConstraint is a Traceable

Table A.2: Added Specification Items in 4.0.3