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

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R23-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2023-11-23	R23-11	AUTOSAR Release Management	• Change primitive type of maxSlots/maxCycles in repetitiveEOC • Improve identification and constraints of root in hierarchicalEOC • Introduce Intra-LET paradigms • Remove EOC.ignoreOrderAllowed
2022-11-24	R22-11	AUTOSAR Release Management	• Added SL-LET feature • Rework document structure
2021-11-25	R21-11	AUTOSAR Release Management	• Removed some obsolete elements • Corrected description of the attribute ExecutionOrderConstraint.orderedElement • Corrected spelling errors and incorrect captions
2020-11-30	R20-11	AUTOSAR Release Management	• Added support for AUTOSAR Software Clusters and AUTOSAR Classic Platform Flexibility

△

2019-11-28	R19-11	AUTOSAR Release Management	• Replaced the verb ”must” by the verb ”shall” according to TPS_STDT_00053. • Corrected spelling errors, terminology, and removed empty pages • Changed attributes of TDEventFrameEthernet to be consistent with changes applied to ethernet communications. • Corrected splitkey definition in timingResource. • Revised section ”Occurrence Expression Language for Timing Events” to improve understanding of the purpose of complex timing description event, as well as the timing functions. • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Added support for Logical Execution Time • Added element SynchronizationPointConstraint • Removed constraint constr_4535 from specification. • Added element BswCompositionTiming
2017-12-08	4.3.1	AUTOSAR Release Management	• Editorial changes in chapter 6 and 7.
2016-11-30	4.3.0	AUTOSAR Release Management	• Added support for conditional timing • Added support for timing constraints for Ethernet communications • Added timing function to support mode dependency • Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation

▽

△

2015-07-31	4.2.2	AUTOSAR Release Management	• Minor corrections and editorial changes • Added appendices C and D
2014-10-31	4.2.1	AUTOSAR Release Management	• Added the capability in Execution Order Constraint to reference RTE and BSW Events • Added description about how to specify time sets • Minor corrections / clarifications / editorial changes; For details please refer to the BWCStatement
2014-03-31	4.1.3	AUTOSAR Release Management	• Revised the entire contents of chapter ”Application Notes” • Applied editorial changes to section ”Repetitive Execution Order Constraint”
2013-10-31	4.1.2	AUTOSAR Release Management	• Clarified the semantics of jitter and removed ambiguities in the description of the Periodic Event Triggering Constraint. • Added AUTOSAR constraints in order to ensure specification of consistent Execution Order Constraints. • Added capability to specify logical successor relationships between runnable entities and groups of runnable entities. • Changed the prefix of timing functions from ”ARTE” to ”TIMEX” in order to be consistent with the AUTOSAR standard definitions. • Clarified the use of event types in the various timing views defined in the specification.

▽

△

2013-03-15	4.1.1	AUTOSAR Release Management	• Applied editorial changes in order to improve readability and comprehensibility of the contents of the document • Added VFB event type TDEventTrigger and extended TDEventSwcInternalBehavior TypeEnum to indicate variable access of runnable entities • Extended the capability of SynchronizationTimingConstraint to reference timing description events • Revised and extended the capabilities of ExecutionOrderConstraint to specify hierarchical and repetitive execution order constraints • Added the capability to specify blueprints of VfbTimings • Added capabilities to reference timing description events in existing timing models and to support reuse of timing models, as well as AUTOSAR methodology
2011-12-22	4.0.3	AUTOSAR Release Management	• Added new timing constraint types AgeConstraint and ExecutionTimeConstraint • Added occurrence expression language for TimingDescriptionEvents • Improved TDEventModeDeclaration, BurstPatternEventTriggering and SwcTiming

▽

△

2011-04-15	4.0.2	AUTOSAR Release Management	• Dropped `InstanceRefs` and replaced with `ComponentInCompositionInstanceRef` • Restricted the semantics of `ExecutionOrderConstraint` and `OffsetTimingConstraint` • Parameterize the observable event 'FlexRayClusterCycleStart' by defining the cycle repetition
2009-12-18	4.0.1	AUTOSAR Release Management	• Initial Release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Contents

References

[1] Meta Model
AUTOSAR_FO_MMOD_MetaModel

[2] Methodology for Classic Platform
AUTOSAR_CP_TR_Methodology

[3] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[4] Virtual Functional Bus
AUTOSAR_CP_EXP_VFB

[5] Glossary
AUTOSAR_FO_TR_Glossary

[6] Generic Structure Template
AUTOSAR_FO_TPS_GenericStructureTemplate

[7] Software Component Template
AUTOSAR_CP_TPS_SoftwareComponentTemplate

[8] Basic Software Module Description Template
AUTOSAR_CP_TPS_BSWModuleDescriptionTemplate

[9] System-level Logical Execution Time:Augmenting the Logical Execution Time Paradigm for Distributed Real-time Automotive Software

[10] System Template
AUTOSAR_CP_TPS_SystemTemplate

[11] Requirements on Timing Extensions
AUTOSAR_FO_RS_TimingExtensions

1 Introduction

1.1 Overview

This AUTOSAR document contains the specification of the AUTOSAR Timing Extensions and describes the elements of the AUTOSAR meta-model [1] used for creating timing models for the respective AUTOSAR Platform. It is a supplement to the formal definition of the Timing Extensions by means of the AUTOSAR meta-model. In other words, this document in addition to the formal definition provides introductory description and rationale for the part of the AUTOSAR meta-model relevant for the creation of timing models.

1.2 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 [2] 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.

1.3 Scope

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

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

(e.g. because of legacy code), the timing specification allows the bottom-up design methodology as well.

The resulting overall specification (AUTOSAR Model *and* Timing Extensions) shall enable the analysis of a system's timing behavior and the validation of the analysis results against timing constraints. Thus, timing properties required for the analysis shall be contained in the timing augmented system model. Example: the priority of a task, the activation behavior of an interrupt, the sender timing of a PDU and frame etc.). Such timing properties can be found all across AUTOSAR. For example the System Template provides means to configure and specify the timing behavior of the communication stack. Furthermore the execution time of executable entities can be specified. In addition, the overall specification shall 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.

1.4 Document Conventions

Technical terms are typeset in mono spaced font, e.g. `PortPrototype`. As a general rule, plural forms of technical terms are created by adding "s" to the singular form, e.g. `PortPrototype`s. By this means the document resembles terminology used in the AUTOSAR XML Schema.

This document contains constraints in textual form that are distinguished from the rest of the text by a unique numerical constraint ID, a headline, and the actual constraint text starting after the ⌈ character and terminated by the ⌋ character.

The purpose of these constraints is to literally constrain the interpretation of the AUTOSAR meta-model such that it is possible to detect violations of the standardized behavior implemented in an instance of the meta-model (i.e. on M1 level).

Makers of AUTOSAR tools are encouraged to add the numerical ID of a constraint that corresponds to an M1 modeling issue as part of the diagnostic message issued by the tool.

The attributes of the classes introduced in this document are listed in form of class tables. They have the form shown in the example of the top-level element AUTOSAR:

Please note that constraints are not supposed to be enforceable at any given time in an AUTOSAR workflow. During the development of a model, constraints may legitimately be violated because an incomplete model will obviously show inconsistencies.

However, at specific points in the workflow, constraints shall be enforced as a safeguard against misconfiguration.

The points in the workflow where constraints shall be enforced, sometimes also known as the "binding time" of the constraint, are different for each model category, e.g. on the classic platform, the constraints defined for software-components are typically enforced prior to the generation of the RTE while the constraints against the definition of an Ecu extract shall be applied when the Ecu configuration for the Com stack is created.

For each document, possible binding times of constraints are defined and the binding times are typically mentioned in the constraint themselves to give a proper orientation for implementers of AUTOSAR authoring tools.

Let `AUTOSAR` be an example of a typical class table. The first rows in the table have the following meaning:

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

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

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

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

The headers in the table have the following meaning:

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

Type: The type of an attribute of the class.

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

Kind: Specifies, whether the attribute is aggregated in the class (`aggr` aggregation), an UML attribute in the class (`attr` primitive attribute), or just referenced by it (`ref` reference). Instance references are also indicated (`iref` instance reference) in this field.

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

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

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([3]).

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([3]).

2 Fundamentals

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

Events: Events refer to locations in systems at which the *occurrences* of events are observed. The AUTOSAR Specification of Timing Extensions defines a set of predefined event types for such *observable locations*. Those event types are used in different *timing views* and each of these timing views correspond to one of the AUTOSAR platform views: *VFB Timing* and Virtual Functional Bus (VFB) View:

- *SW-C Timing* and Software Component View
- *System Timing* and System View
- *BSW Module Timing* and Basic Software Module View
- *ECU Timing* and ECU View.

In particular, these events are used to specify:

- reading and writing of data from and to specific SWC ports
- calling of services and receiving their responses (VFB, SW-C, System and ECU Timing)
- sending and receiving data via networks and through communication stacks (System and ECU Timing)
- activating, starting and terminating executable entities (SW-C Timing and Basic SW Module Timing)
- calling basic software services and receiving their responses (ECU Timing and Basic SW Module Timing).

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

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

Timing Constraints imposed on Event Chains: Like event triggering constraints impose timing constraints on events and their occurrences; the latency and synchronization timing constraints impose constraints on event chains. In the former case, a constraint is used to specify a reaction and age, for example if a stimulus event occurs then the corresponding response event shall occur not later than a given amount of time. And in the latter case, the constraint is used to specify that stimuli or response events shall occur within a given time interval (tolerance) to be said to occur simultaneous and synchronous respectively.

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

These fundamental concepts sketch the representation in the meta-model and form the basis of the descriptions in the subsequent sections.

2.1 Formal specification of timing behavior

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

Having to deal with different abstraction levels and views (see chapter 3.1), and in order to avoid semantic confusion with existing concepts, a new abstract type `TimingDescriptionEvent` (see section 3.3.2) is introduced as a formal basis for the timing extensions. Depending on the model entity and the associated observable behavior, specific timing events are defined and linked to the different views.

For the analysis of a system's timing behavior usually not only single events but also the correlation of different events is of fundamental importance. To relate timing events to each other, a further concept called `TimingDescriptionEventChain` (see section 3.3.1) is introduced. Hereby, it is important to note that for the referenced events of an event chain a functional dependency is implicitly assumed. This means that an event

of a chain somehow causes subsequent chain events. An example for an end-to-end event chain with bus communication is depicted in Figure 3.5 in chapter 3.1. This event chain describes the path from software component instance ”SWC1” to software component instance ”SWC3”.

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

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.

3 Modeling

This chapter shall walk through the meta-model representation of the timing extensions in the following sub-sections.

3.1 TimingExtensions (Timing Views)

An AUTOSAR Timing Extension model starts with the meta-class `TimingExtension` or rather, one of the sub-classes of `TimingExtension` as the top-level element. This is the owning element for all other related elements. The sub-classes of `TimingExtension` define a set of timing views as shown in Figure 3.1 and detailed in the next sub-sections. The chosen view depends on the availability of necessary information and the stage in the workflow. The timing views are:

- **VfbTiming**: timing information related to the interaction of `SwComponentType`s at VFB level (3.1.1)
- **SwcTiming**: timing information related to the `SwcInternalBehavior` of `AtomicSwComponentType`s (3.1.2)
- **SystemTiming**: timing information related to a `System`, utilizing information about topology, software deployment, and signal mapping (3.1.3)
- **BswModuleTiming**/**BswCompositionTiming**: respectively, timing information related to the `BswInternalBehavior` of a single `BswModuleDescription`; timing information related to `BswInternalBehavior` of more than one implementation of a `BswModuleDescription` (3.1.4)
- **EcuTiming**: timing information related to the `EcucValueCollection` covering the entire software on an ECU with application software and configured basic software (3.1.5)

[TPS_TIMEX_00009] Optionality of Timing Extensions ⌈To maintain it's optional use in the AUTOSAR methodology, TIMEX always references "outwards" towards other external (i.e. in other template specifications) model elements.⌋ *(RS_TIMEX_00003)*

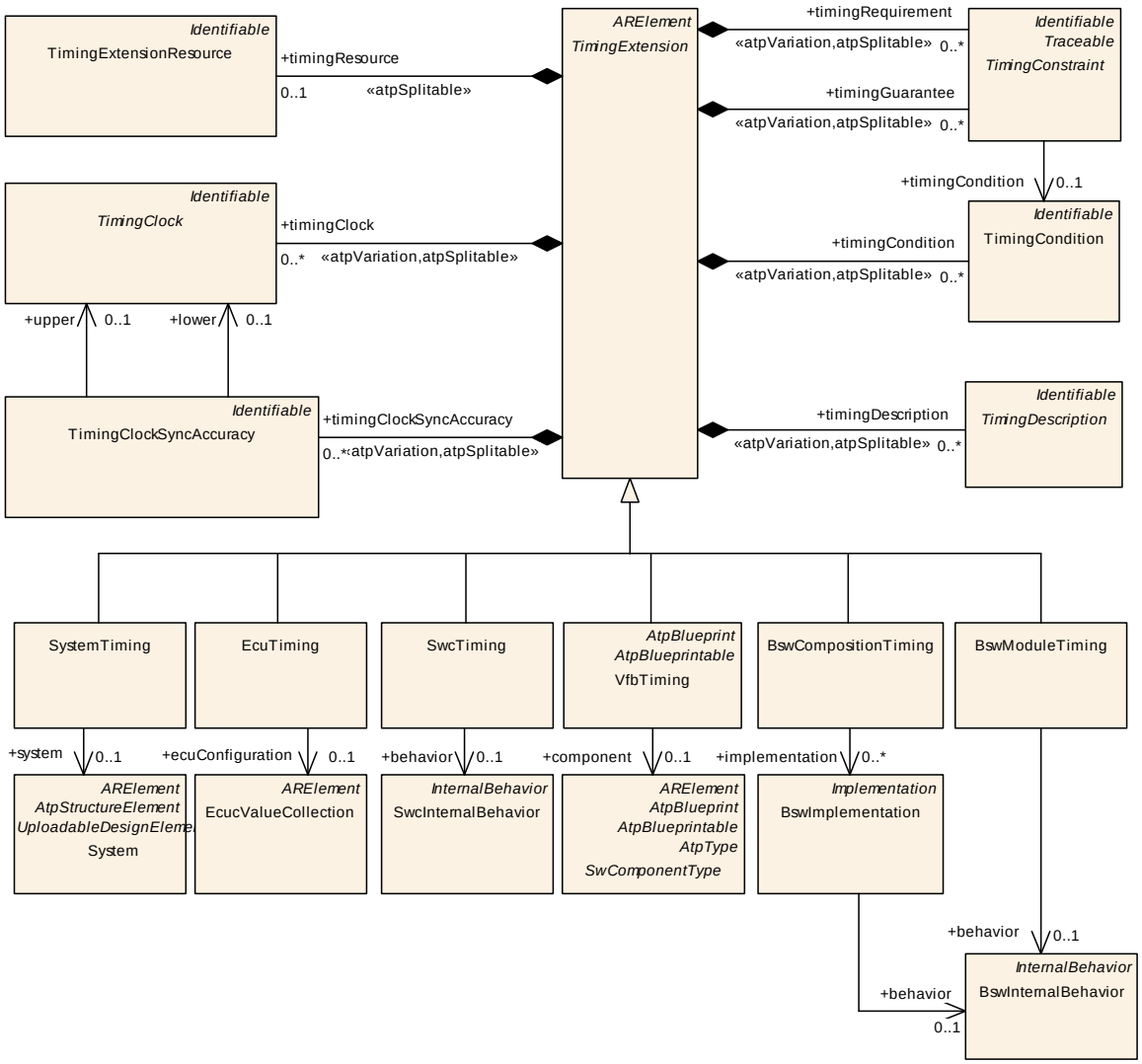


Figure 3.1: Timing Extensions top-level view

3.1.1 VfbTiming

AUTOSAR defines the *Virtual Functional Bus* [4] as a composition of SwComponentPrototypes at a logical level, regardless of their physical distribution. On this logical level a special view can be applied for timing specification. This section describes what kind of timing specification can be applied at VFB level for a system or sub-system. Typically, end-to-end timing constraints, including (physical) sensors and actuators, shall be captured in this view, allowing an early formalization of those constraints.

Neglecting the physical distribution means that the VfbTiming view does not deal with the question, in which system context the prototype of a CompositionSwComponentType shall be implemented. An additional restriction of the VfbTiming view is present 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), TimingDescrip-

tions at VFB level should only refer to SwComponentTypes, PortPrototypes and their connections, but not the InternalBehavior.

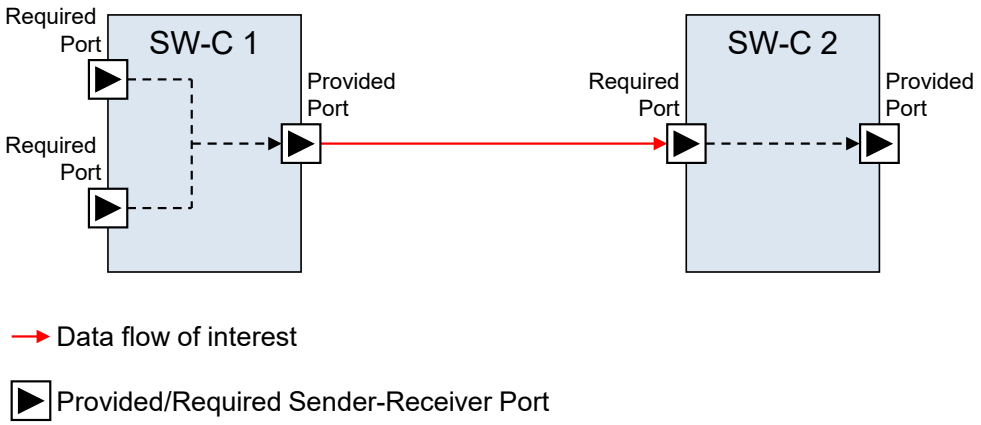


Figure 3.2: Example: Data flow in the scope of the VfbTiming view

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

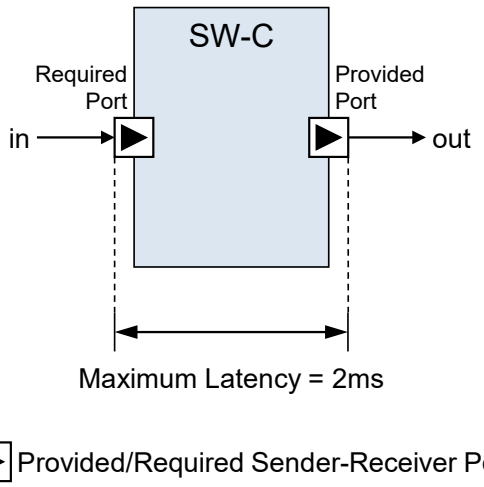


Figure 3.3: Example: Latency requirement

As an example, consider the timing constraint illustrated in Figure 3.3: "From the point in time, where the value *in* is received by the 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 *SW-C*.

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

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

Class	**VfbTiming**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingExtensions			
Note	A model element used to define timing descriptions and constraints at VFB level. TimingDescriptions aggregated by VfbTiming are restricted to event chains referring to events which are derived from the class TDEventVfb. **Tags:** atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *TimingExtension*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
component	SwComponentType	0..1	ref	This defines the scope of a VfbTiming. All corresponding timing descriptions and constraints shall be defined within this scope.

Table 3.1: VfbTiming

[constr_6848] Existence of `VfbTiming.component` ⌈For each `VfbTiming`, the reference to `SwComponentType` in the role `component` shall exist **`at the time when the VFB Timing Description is complete`**.⌋ *()*

3.1.2 SwcTiming

In contrast to the `VfbTiming` view, a specification engineer might especially be interested in the `SwcInternalBehavior` of `AtomicSwComponentType`s that are represented as black boxes at VFB level. The `SwcInternalBehavior` specifies a component's behavioral decomposition into runnable entities, which are executed at runtime. Thus, in `SwcTiming` view, a timing description is attached to the `SwcInternalBehavior` of a `SwComponentType`. It can refer to the activation, start, and termination (see section 3.3.2.2.1) of the execution of a `RunnableEntity`.

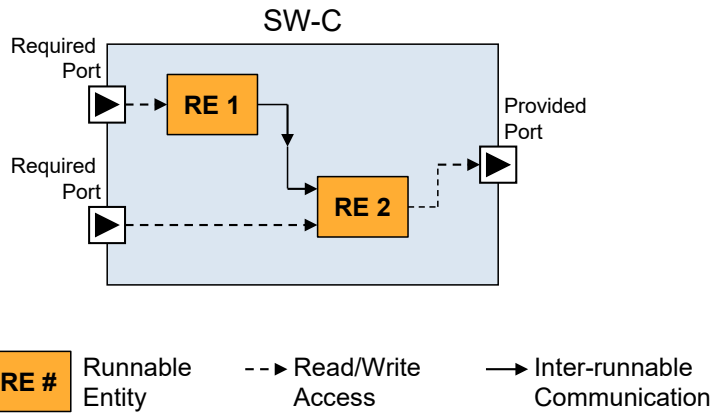


Figure 3.4: Example: Data flow in the scope of the SW-C Timing view

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

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

Table 3.2: SwcTiming

3.1.3 SystemTiming

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

The `SystemTiming` view is used to provide timing information at system level. As an extension, it can be attached to a `System`. As the `System` description aggregates all the information about `SwComponentType`s and their corresponding `SwcInternalBehavior`, it is possible to use the same concepts that are available in the views `VfbTiming` and `SwcTiming` also in this timing view. The difference is the specific system context that defines the validity of timing information at system level. Without knowledge of the mapping of software components to a target hardware respectively ECU, only a generic platform independent description can be provided.

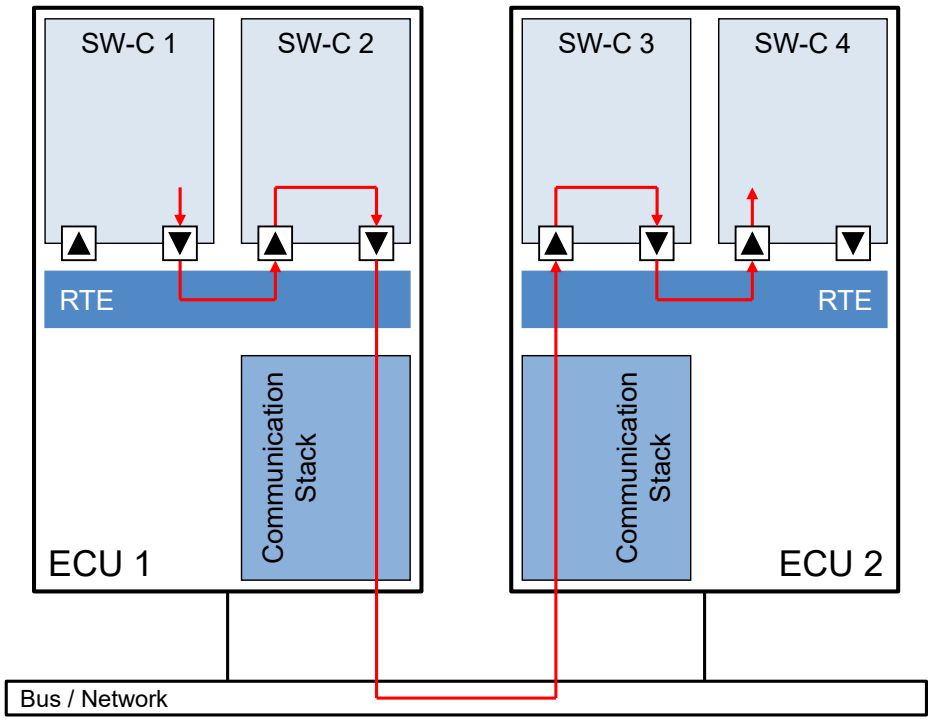


Figure 3.5: Example: Data flow in the scope of System Timing view

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

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

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

▽

△

Class	SystemTiming			
system	System	0..1	ref	This defines the scope of a SystemTiming. All corresponding timing descriptions and constraints shall be defined within this scope.

Table 3.3: SystemTiming

[constr_6849] Existence of `SystemTiming.system` ⌈For each `SystemTiming`, the reference to `System` in the role `system` shall exist **`at the time when the System Timing Description is complete`**.⌋ *()*

3.1.4 BswModuleTiming and BswCompositionTiming

The Basic Software Timing consists of two perspectives, which are described in the next sections.

3.1.4.1 BswModuleTiming

As part of the methodology, a `BswModuleDescription` is generated for each BSW module as part of the ECU configuration phase. For every module its internals, the `BswInternalBehavior`, shall be defined, i.e. structuring any `BswModuleEntity`. Similar to the timing view on the `SwcInternalBehavior` of an `AtomicSwComponentType` as described in section 3.1.2, the BSW module timing view focuses on the activation, start and end of the execution of any `BswModuleEntity`. The timing description for each module can be attached to the `BswModuleDescription`.

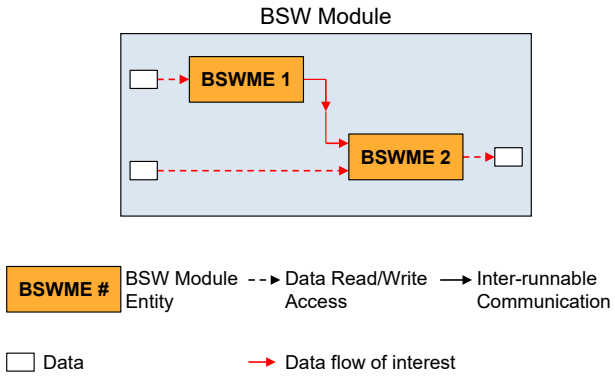


Figure 3.6: Example: Data flow in scope of BSW Module Timing view

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

Class	**BswModuleTiming**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingExtensions			
Note	A model element used to define timing descriptions and constraints for the BswInternalBehavior of one BSW Module. Thereby, for each BswInternalBehavior a separate timing can be specified. A constraint defined at this level holds true for all Implementations of that BswInternalBehavior. TimingDescriptions aggregated by BswModuleTiming are restricted to event chains referring to events which are derived from the class TDEventBswInternalBehavior. **Tags:** atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *TimingExtension*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
behavior	BswInternalBehavior	0..1	ref	This defines the scope of a BswModuleTiming. All corresponding timing descriptions and constraints shall be defined within this scope.

Table 3.4: BswModuleTiming

[constr_6850] Existence of `BswModuleTiming.behavior` ⌈For each `BswModuleTiming`, the reference to `BswInternalBehavior` in the role `behavior` shall exist **`at the time when the Bsw Timing Description is complete`**.⌋*()*

3.1.4.2 BswCompositionTiming

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

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

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

▽

△

Class	**BswCompositionTiming**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *TimingExtension*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
implementation	BswImplementation	*	ref	This defines the scope of a BswCompositionTiming. All corresponding timing descriptions and constraints shall be defined within this scope.

Table 3.5: BswCompositionTiming

[constr_6851] Existence of `BswCompositionTiming.implementation` ⌈For each `BswCompositionTiming`, the reference to `BswImplementation` in the role `implementation` shall exist at least once **`at the time when the Bsw Timing Description is complete`**.⌋*()*

3.1.5 EcuTiming

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

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

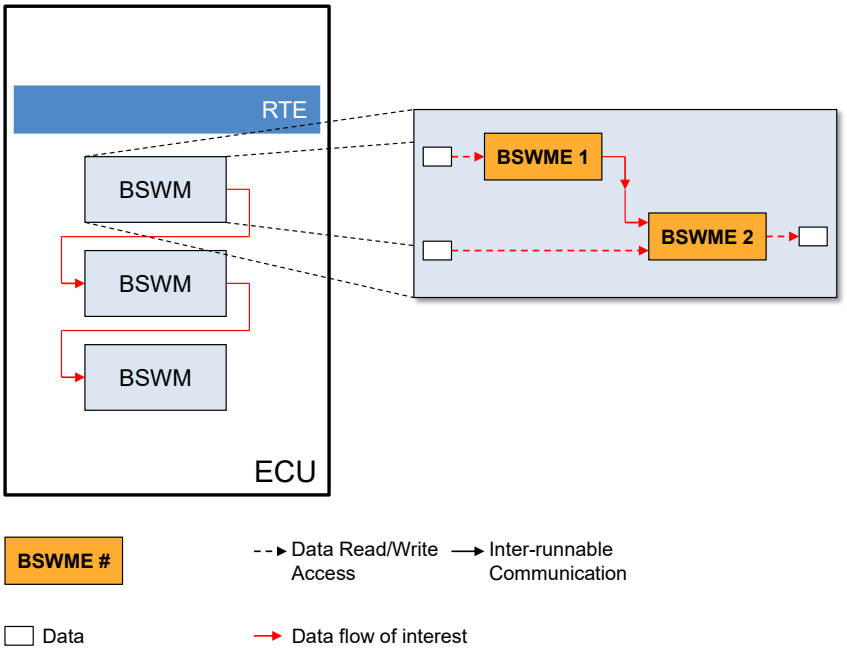


Figure 3.7: An example of data flow, whose timing behavior is in scope of ECU view

In this view, timing can reference all the ECU-relevant information: The deployed software component instances, the ECU related interactions including bus communication, Basic Software, etc. In other words, the EcuTiming has the same expressivity as the System Timing view but only focusing on one specific ECU in the system's topology. In addition, the entire BSW can be considered during timing modeling, because the complete composition and internal structure of the BSW modules are known. The internals of BSW modules and the inter-relations between BS modules are of interest in this timing view. The information is attached to the EcucValueCollection.

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

Class	**EcuTiming**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingExtensions			
Note	A model element used to define timing descriptions and constraints within the scope of one ECU configuration. TimingDescriptions aggregated by EcuTiming are allowed to use all events derived from the class TimingDescriptionEvent. **Tags:** atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *TimingExtension*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ecu Configuration	EcucValueCollection	0..1	ref	This defines the scope of an EcuTiming. All corresponding timing descriptions and constraints shall be defined within this scope.

Table 3.6: EcuTiming

[constr_6852] Existence of EcuTiming.ecuConfiguration ⌈For each EcuTiming, the reference to EcucValueCollection in the role ecuConfiguration shall exist at least once **at the time when the Ecu Timing Description is complete**.⌋ *()*

3.2 Timing Conditions

In almost all cases, systems, and the application software executed in those systems, operate under various conditions, like normal condition, error condition, start-up condition, etc. During the operation of such systems the conditions may change at any time.

As a consequence timing constraints imposed on the system may vary depending on these conditions, too. TIMEX provides a means to support the description of timing constraints depending on such conditions by *Conditional Timing* as described in this section.

Almost all software management systems controlling an internal combustion engine shall maintain a constant temperature of the coolant to ensure the optimal operation

under specific conditions. For example, critical thermal conditions may occur that may lead to severe damage of the engine's mechanical components. In order to prevent the engine's mechanical components from being damaged the software application shall respond faster than in the nominal case. Therefore, different timing constraints are imposed on a given event chain depending on a known condition.

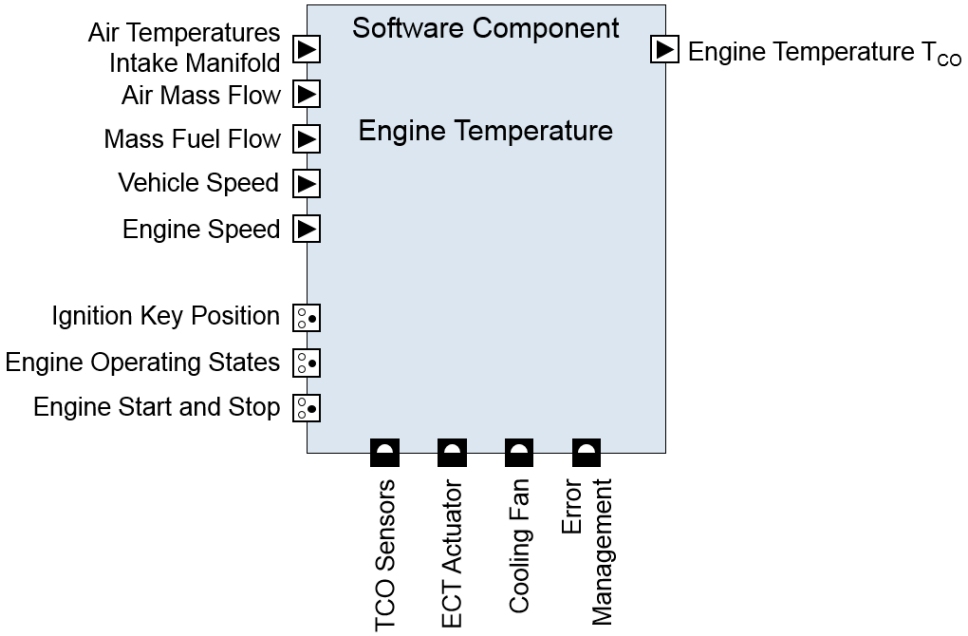


Figure 3.8: Rough sketch of an AUTOSAR software component (SW-C) controlling the coolant temperature (TCO) of an internal combustion engine. This SW-C shall consider various modes of operation as shown in the lower left corner of the components, namely the mode ports called "Ignition Key Position", "Engine Operating States", and "Engine Start and Stop".

As shown in Figure 3.8 the SWC controlling the coolant temperature requires information on the current operating conditions of the internal combustion engine. Depending on these conditions different timing constraints are imposed on an event chain requiring different reaction time constraints.

In one situation, the reaction time shall be much faster than in other situation. For example, the mode port called `Engine Operating States` indicates a specific condition respectively *mode* the internal combustion engine operates in. In this specific mode it may happen that the internal combustion engine faces a critical thermal condition.

In order to prevent the engine from being damaged the Electronically Controlled Thermostat (ETC) shall be actuated much faster (client port called `ECT Actuator`) than in other cases. This is accomplished by specifying a tighter reaction time for this condition; and under any other condition the reaction time may be more relaxed.

Besides this particular example, there are a lot more cases where the timing constraints imposed on given event chains change depending on the internal combustion engines mode.

In dependable systems an application is typically designed in a fault tolerant manner. It either operates under normal condition — normal mode — or under failure condition — failure mode. As sketched in the upper part of Figure 3.9 in such an application a number of runnable entities process the values of three sensors. In a first step the sensor values are corrected by a runnable entity and are passed to another runnable entity of the application that utilizes a control algorithm to calculate an output. This output value is used to control an actuator or is processed by other runnable entities in the system. In the same application another runnable entity checks the plausibility of the values received from the sensors.

The purpose of this runnable entity is to determine any unexpected and implausible deviation from the expected values of each sensor. If such an deviation is detected by the runnable entity checking the plausibility of the sensor values then a runnable entity correcting the sensor values is notified about this condition, namely indicating a failure of one of the sensors. The runnable entity correcting the sensor values is capable of deriving the value of the erroneous sensor based on the values of the other sensors which monitor other physical properties.

Thus, the application is operating under different conditions — normal and failure mode — and with regard to the software implementation this results in different worst case execution times of the runnable entity correcting the sensor values.

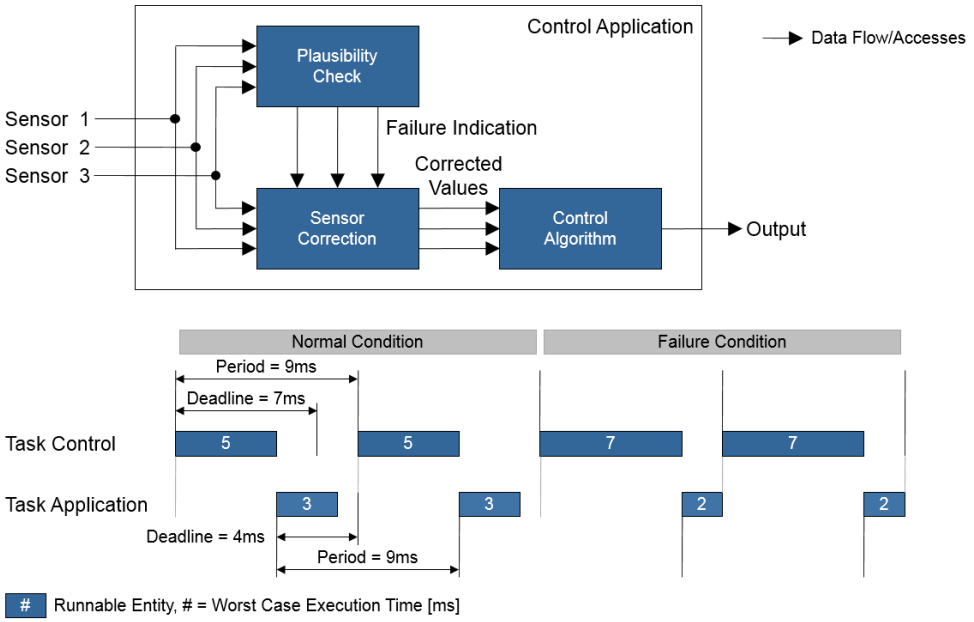


Figure 3.9: Generic fault tolerant control application

Assumed the application executing the three runnable entities is mapped to an ECU executing already runnable entities of another application. The runnable entities of these two applications are executed in the context of two tasks called `Task Control` and `Task Application`. The former task executes the runnable entities of the given fault tolerant application and the latter task executes the application already being mapped to the ECU and consisting of a number of runnable entities. Both tasks respectively

applications are activated periodically every 9ms. The task Task Control has a deadline of 7ms and the task Task Application has a deadline of 4ms.

As mentioned above the first application operates under two different conditions: Under normal condition the execution time of the runnable entities does not exceed 5ms, and under failure condition the execution time of the runnable entities increases to 7ms due to the additional time required for executing the algorithm to derive the value of the erroneous sensor.

A rough sketch of the task schedule containing both tasks is given in the lower part of Figure 3.9. On the left hand side the fault tolerant application is operating under normal conditions and on the right hand side this application is operating under failure condition. In this case — failure condition — the execution time 3ms of the task ask Application leads to a violation of the schedule, which means that the two applications cannot be integrated onto the same ECU unless both applications are mapped onto a more powerful ECU — resulting in a decrease of execution times. Another possibility would be that the application executed by the task Task Application provides capabilities to operate under degraded conditions resulting in a shorter execution time, for example 2ms.

However, in the cases described above timing constraints imposed on timing properties of SWCs and/or runnable entities depend on specific conditions a system shall operate properly.

TIMEX is capable of specifying timing constraints for specific conditions as shown in Figure 3.10. Such a timing constraint is valid if and only if the given timing condition holds.

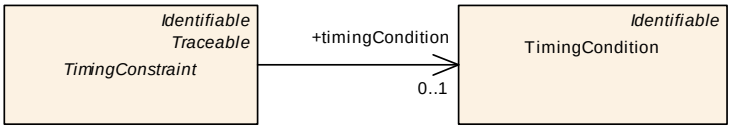


Figure 3.10: Conditional Timing Constraint

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

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

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

[TPS_TIMEX_00051] Purpose of TimingExtensionResource ⌈The purpose of the TimingExtensionResource is to subsume a number of re-usable elements, like instance references to various timing relevant elements, that are referenced from within a TimingConditionFormula. These elements of a TimingExtensionResource

are related to the context of the aggregating — *parent* — `TimingExtension`.⌋(*RS_TIMEX_00011*)

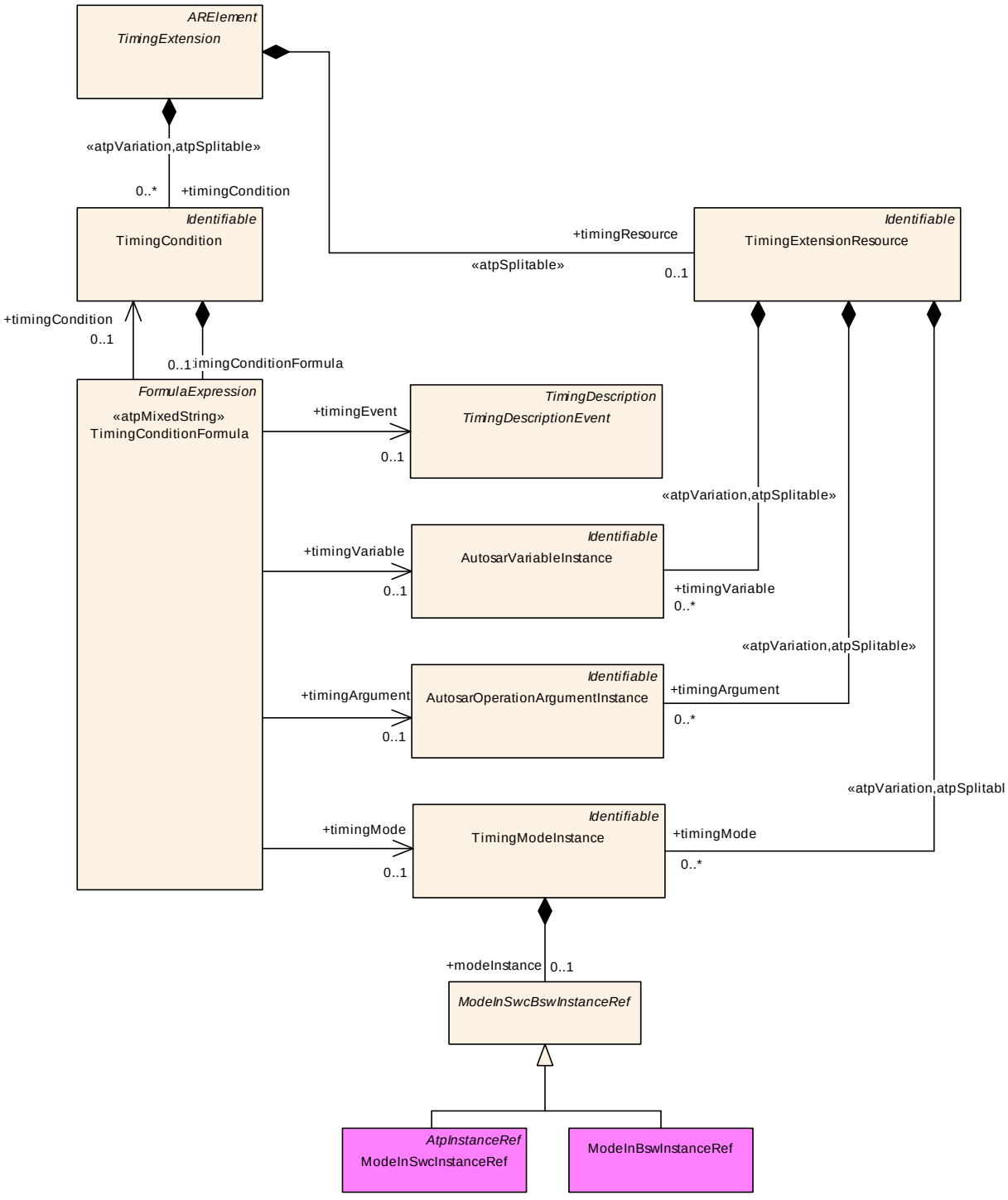


Figure 3.11: Conditional Timing

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

Table 3.7: TimingCondition

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

Table 3.8: TimingConditionFormula

Class	TimingExtensionResource			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingCondition			
Note	A TimingExtensionResource provides the capability to contain instance references referred from within a timing condition formula.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*TimingExtension*.timingResource			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
timingArgument	AutosarOperation ArgumentInstance	*	aggr	This refers to an instance reference of an argument of an operation call. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingArgument.shortName, timing Argument.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	TimingExtensionResource			
timingMode	TimingModeInstance	*	aggr	This refers to an instance reference of a mode declaration. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingMode.shortName, timing Mode.variationPoint.shortLabel vh.latestBindingTime=postBuild
timingVariable	AutosarVariable Instance	*	aggr	This refers to an instance reference of a variable. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingVariable.shortName, timing Variable.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table 3.9: TimingExtensionResource

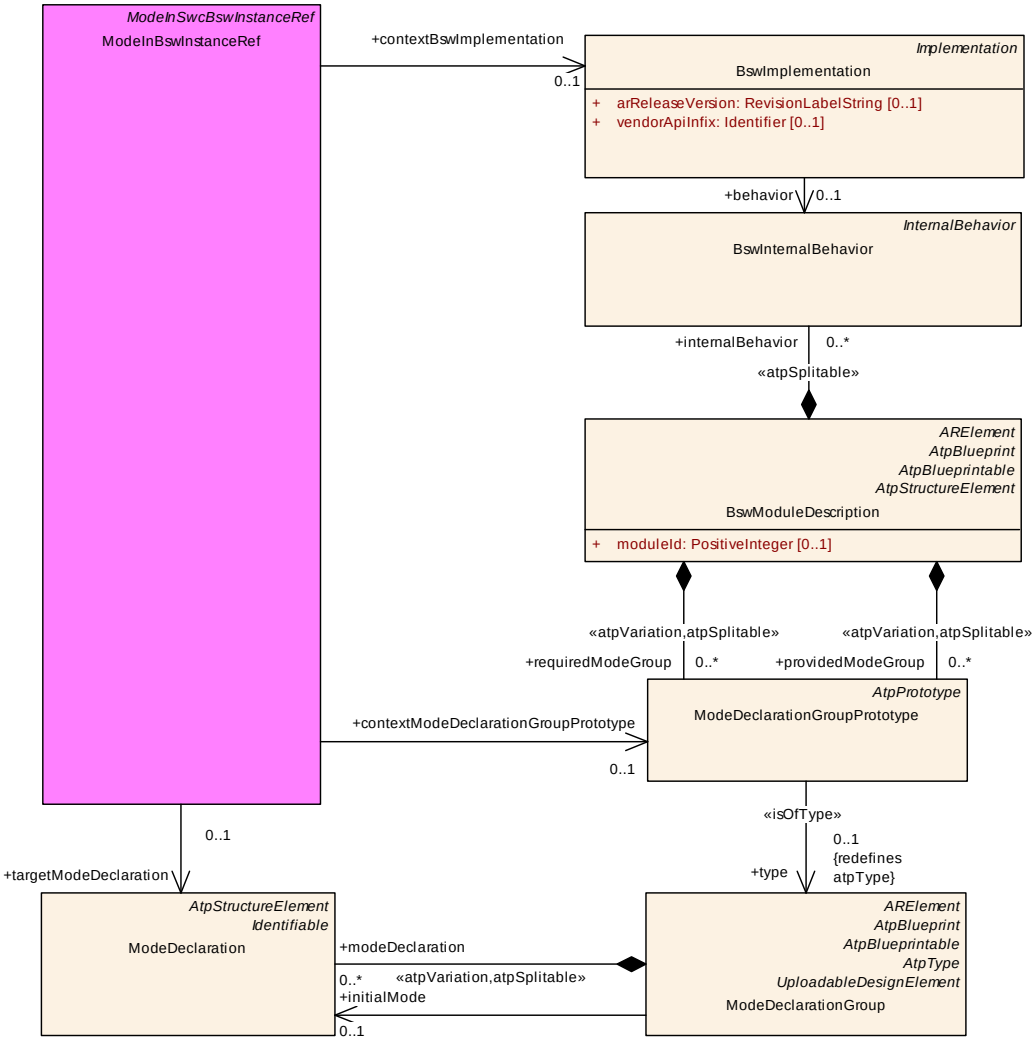


Figure 3.12: Instance Reference Mode for Basic Software Module

Since the notion of ”Type and Prototype” is not supported by the Basic Software Module Description Template, the element `ModeInBswInstanceRef` is not a specialization of

AtpInstanceRef. Therefore, the directed association playing the role of ”base” is not present.

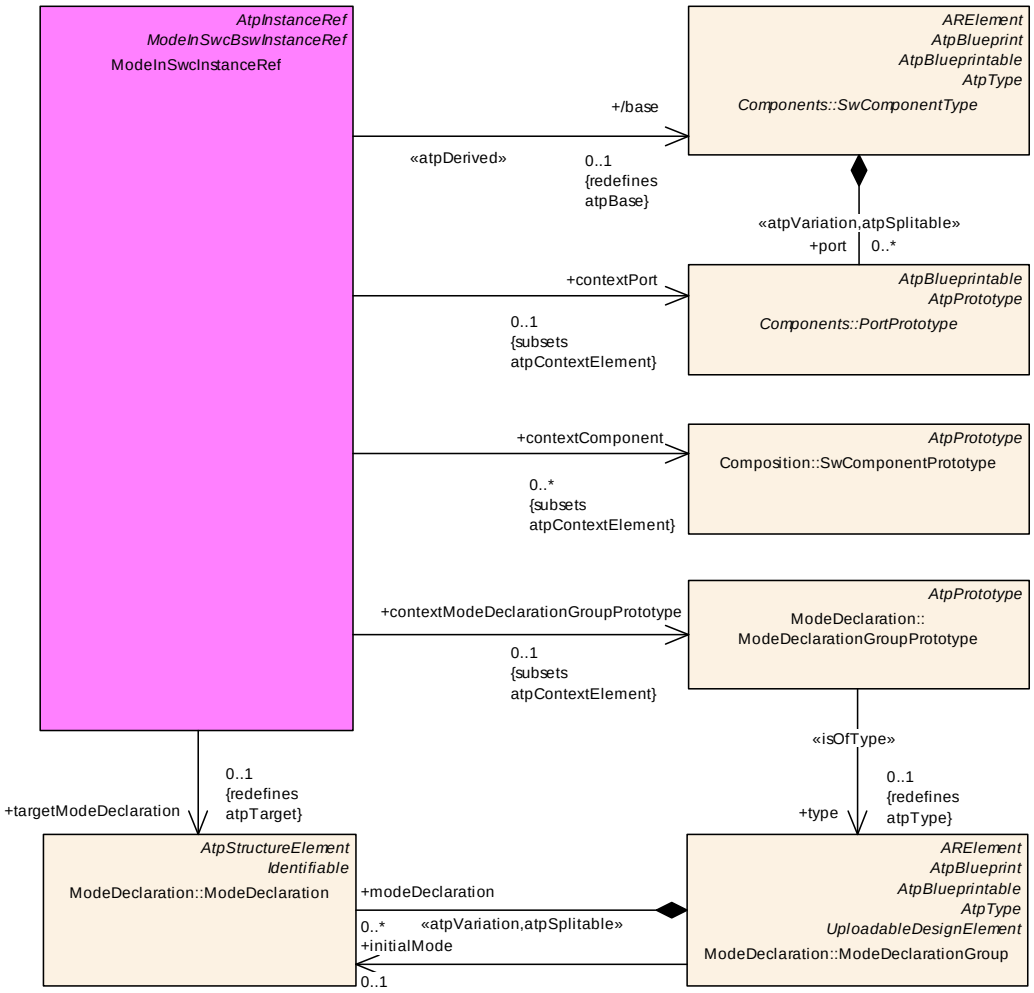


Figure 3.13: Instance Reference Mode for Software Component

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

Table 3.10: TimingModeInstance

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

Table 3.11: ModeInBswInstanceRef

[constr_6853] Existence of `ModeInBswInstanceRef.contextModeDeclarationGroupPrototype` ⌈For each `ModeInBswInstanceRef`, the reference to `ModeDeclarationGroupPrototype` in the role `contextModeDeclarationGroupPrototype` shall exist at least once **`at the time when the Bsw Timing Description is complete`**.⌋*()*

[constr_6854] Existence of `ModeInBswInstanceRef.targetModeDeclaration` ⌈For each `ModeInBswInstanceRef`, the reference to `ModeDeclaration` in the role `targetModeDeclaration` shall exist at least once **`at the time when the Bsw Timing Description is complete`**.⌋*()*

Class	**ModeInSwcInstanceRef**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingCondition			
Note	Instance reference to be capable of referencing a ModeDeclaration at a specific Mode Switch Port of a SW-C.			
Base	*ARObject*, *AtpInstanceRef*, *ModeInSwcBswInstanceRef*			
Aggregated by	TimingModeInstance.modeInstance			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
base	SwComponentType	0..1	ref	Specifies the SW component representing the base of the context. **Stereotypes:** atpDerived **Tags:** xml.sequenceOffset=10
context Component	SwComponent Prototype	*	ref	Specifies the SW component prototype representing the context. **Tags:** xml.sequenceOffset=20
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	ref	Specifies the mode declaration group prototype that manifests the context. **Tags:** xml.sequenceOffset=40
contextPort	PortPrototype	0..1	ref	Specifies the port prototype representing the context. **Tags:** xml.sequenceOffset=30

▽

△

Class	**ModeInSwcInstanceRef**			
targetMode Declaration	ModeDeclaration	0..1	ref	Specifies the specific mode declaration in the given context. **Tags:** xml.sequenceOffset=50

Table 3.12: ModeInSwcInstanceRef

[constr_6855] Existence of ModeInSwcInstanceRef.contextModeDeclarationGroupPrototype ⌈For each ModeInSwcInstanceRef, the reference to ModeDeclarationGroupPrototype in the role contextModeDeclarationGroupPrototype shall exist at least once **at the time when the Swc Timing Description is complete**.⌋*()*

[constr_6856] Existence of ModeInSwcInstanceRef.contextPort ⌈For each ModeInSwcInstanceRef, the reference to PortPrototype in the role contextPort shall exist at least once **at the time when the Swc Timing Description is complete**.⌋*()*

[constr_6857] Existence of ModeInSwcInstanceRef.targetModeDeclaration ⌈For each ModeInSwcInstanceRef, the reference to ModeDeclaration in the role targetModeDeclaration shall exist at least once **at the time when the Swc Timing Description is complete**.⌋*()*

[constr_6899] Existence of ModeInSwcInstanceRef.base ⌈For each ModeInSwcInstanceRef, the reference to SwComponentType in the role base shall exist at least once **at the time when the Swc Timing Description is complete**.⌋*()*

3.3 TimingDescription

The TimingDescription is an abstract class which provides the base for the two abstract sub-classes TimingDescriptionEventChain and TimingDescriptionEvent - which further provide the base for the respective concrete event types as shown in Figure 3.14. These are detailed in the next sections.

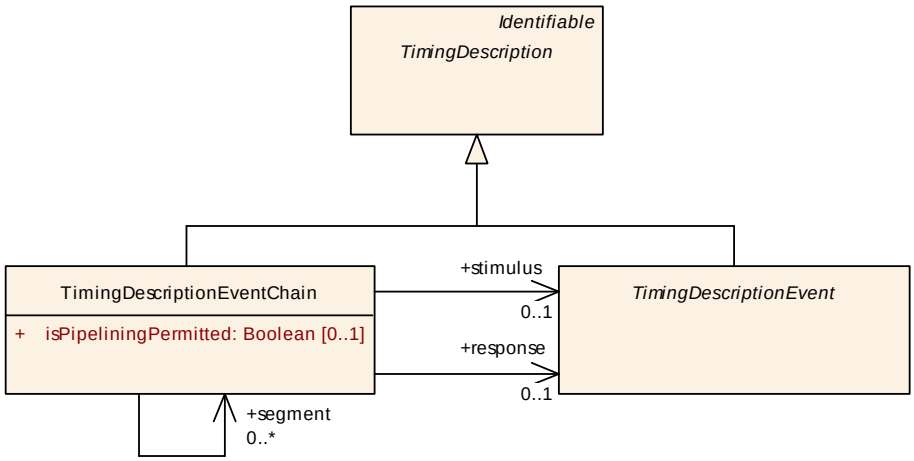


Figure 3.14: TimingDescription

3.3.1 TimingDescriptionEventChain

A `TimingDescriptionEventChain` describes a causal order for a set of functionally dependent `TimingDescriptionEvent`s. Each `TimingDescriptionEventChain` defines at least the relationship between a `stimulus` and a `response`. By means of an `TimingDescriptionEventChain`, 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 `TimingConstraint`s.

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

[TPS_TIMEX_00111]{DRAFT} **Semantics of `TimingDescriptionEventChain.stimulus`** ⌈A `stimulus` represents the starting point (or initial `TimingDescriptionEvent`) of a `TimingDescriptionEventChain`⌋*()*

[TPS_TIMEX_00114]{DRAFT} **Semantics of `TimingDescriptionEventChain.response`** ⌈A `response` represents the end point (or final `TimingDescriptionEvent`) of a `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*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	TimingDescriptionEventChain			
isPipelining Permitted	Boolean	0..1	attr	States whether the scheduled entities in an LET interval shall use pipelined execution or not i.e. "permitted pipelining property" If TRUE, then the scheduled entities must implement pipelining. If FALSE or undefined, no pipelining applies. **Tags:** atp.Status=draft
response	TimingDescriptionEvent	0..1	ref	The response event representing the point in time where the event chain is terminated. **Tags:** xml.sequenceOffset=20
segment	TimingDescriptionEvent Chain	*	ref	A composed event chain consists of an arbitrary number of sub-chains. **Tags:** xml.sequenceOffset=30
stimulus	TimingDescriptionEvent	0..1	ref	The stimulus event representing the point in time where the event chain is activated. **Tags:** xml.sequenceOffset=10

Table 3.13: TimingDescriptionEventChain

Depending on the value of the `category`s of the `TimingDescriptionEventChain`, it may be used in different use-cases.

[TPS_TIMEX_00110]{DRAFT} **Standardized `category`s of `TimingDescriptionEventChain`** ⌈AUTOSAR standardizes the following `category`s of `TimingDescriptionEventChain` and their semantics:

- `undefined`: as per `STANDARD`
- `STANDARD`: No specific semantics are imposed on the `TimingDescriptionEventChain`. It indicates the standard behavior.
- `LET_INTERVAL`: The `TimingDescriptionEventChain` represents an LET interval
- `SL_LET_INTERVAL`: The `TimingDescriptionEventChain` represents a SL-LET interval

⌋()

[constr_4515] Orthogonality of `stimulus` and `response` in a `TimingDescriptionEventChain` ⌈The reference `TimingDescriptionEventChain.stimulus` and the reference `TimingDescriptionEventChain.response` shall not reference the same `TimingDescriptionEvent.shortName`.⌋()

[constr_6906]{DRAFT} **Conformity of `stimulus` and `response` in a `TimingDescriptionEventChain`** ⌈The `TimingDescriptionEvent`s referenced in the roles `stimulus` and `response` shall be of the same sub-class (of `TimingDescriptionEvent`) **at the imposition time associated with the concrete subclass of `TimingDescriptionEvent`**.⌋()

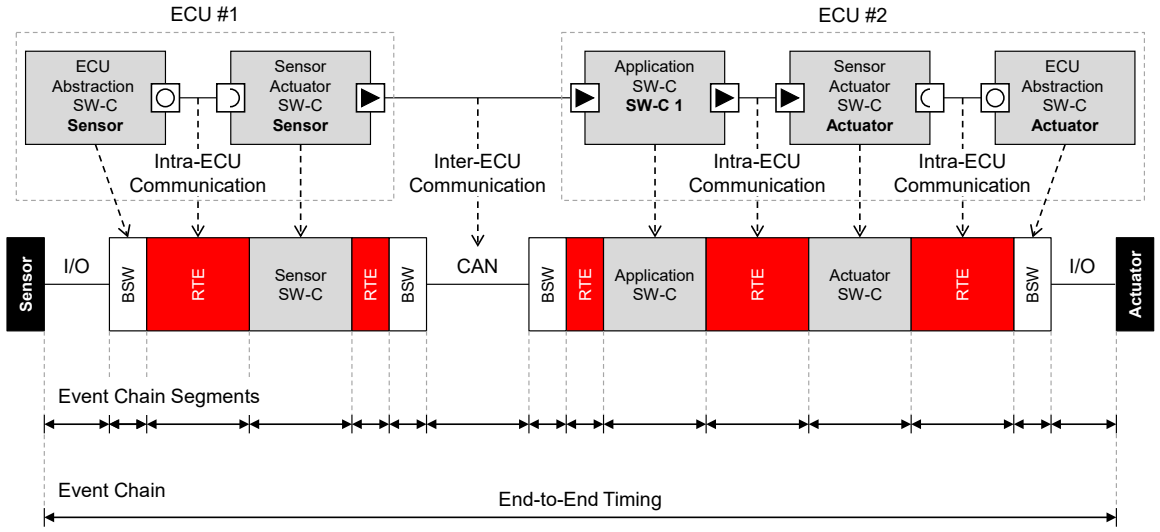
[constr_4560] Restriction of `TimingDescriptionEventChain.category` ⌈Any `TimingDescriptionEventChain.category` not in the list in [TPS_TIMEX_00110] shall be ignored.⌋*()*

[constr_6895] Existence of `TimingDescriptionEventChain.response` ⌈For each `TimingDescriptionEventChain`, the reference in the role `response` shall exist **at the imposition time associated with the concrete subclass of `TimingDescriptionEvent`**.⌋*()*

[constr_6896] Existence of `TimingDescriptionEventChain.stimulus` ⌈For each `TimingDescriptionEventChain`, the reference in the role `stimulus` shall exist **at the imposition time associated with the concrete subclass of `TimingDescriptionEvent`**.⌋*()*

[constr_6897] Existence of `TimingDescriptionEventChain.segment` ⌈For each `TimingDescriptionEventChain`, the reference in the role `segment` shall exist at least once **at the imposition time associated with the concrete subclass of `TimingDescriptionEvent`**.⌋*()*

Figure 3.15 shows an event chain ”End-to-End Timing” describing the causal dependency between ”Sensor” and ”Actuator”. The sequence of event chain segments shows the details of ”End-to-End Timing” according to the AUTOSAR timing views.



BSW Basic Software
RTE Runtime Environment
SW-C Software Component

Figure 3.15: Example of an end-to-end event chain between sensor sampling and actuator access

3.3.1.1 Segments

[constr_4516] Specifying event chain `segments` ⌈If a `TimingDescriptionEventChain` consists of further event chain `segments` then at least one sequence of

event chain segments shall exist from the event chain's stimulus to the response.⌋ *()*

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

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

3.3.1.2 Approach

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

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

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

3.3.1.2.1 Decomposition

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

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

Figure 3.16 shows a time critical path between the event "requesting the brake pedal position" (*Stimulus*) and the event "making available the determined vehicle speed" (*Response*). This event chain (*EC*) is subject to a timing constraint, namely a LatencyTimingConstraint, and is budgeted accordingly. For example, the time budget for the event chain *EC* is constrained by a maximum latency of 2 ms.
In subsequent steps of the development and with deeper knowledge about the system's dynamics, this event chain and its time budget can be split across the system's components. This results in the event chain segments *EC1*, *EC2* and *EC3* and their

appropriate time budgets. The sum of these time budgets shall not exceed the given time budget of 2 ms.

3.3.1.2.2 Composition

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

Figure 3.16 shows the connected event chains $EC1$, $EC2$ and $EC3$. For each event chain a time budget, using a `LatencyTimingConstraint`, is specified: The time budget of event chain *EC1* is 0.5 ms, of event chain *EC2* is 0.6 ms and of event chain *EC3* is 0.7 ms. The high-level event chain *EC* is a composition of the event chains *EC1*, *EC2* and *EC3*. The stimulus event of the high-level event chain is the event ”requesting the brake pedal position” (*Stimulus*) and the response event of the high-level event chain is the event ”making available the determined vehicle speed” (*Response*). Eventually, a time budget is assigned to the high-level event chain using a `LatencyTimingConstraint`, for example 2 ms. This value is consistent with the aggregated time budget of the event chain segments (0.5 ms + 0.6 ms + 0.7 ms = 1.8 ms).

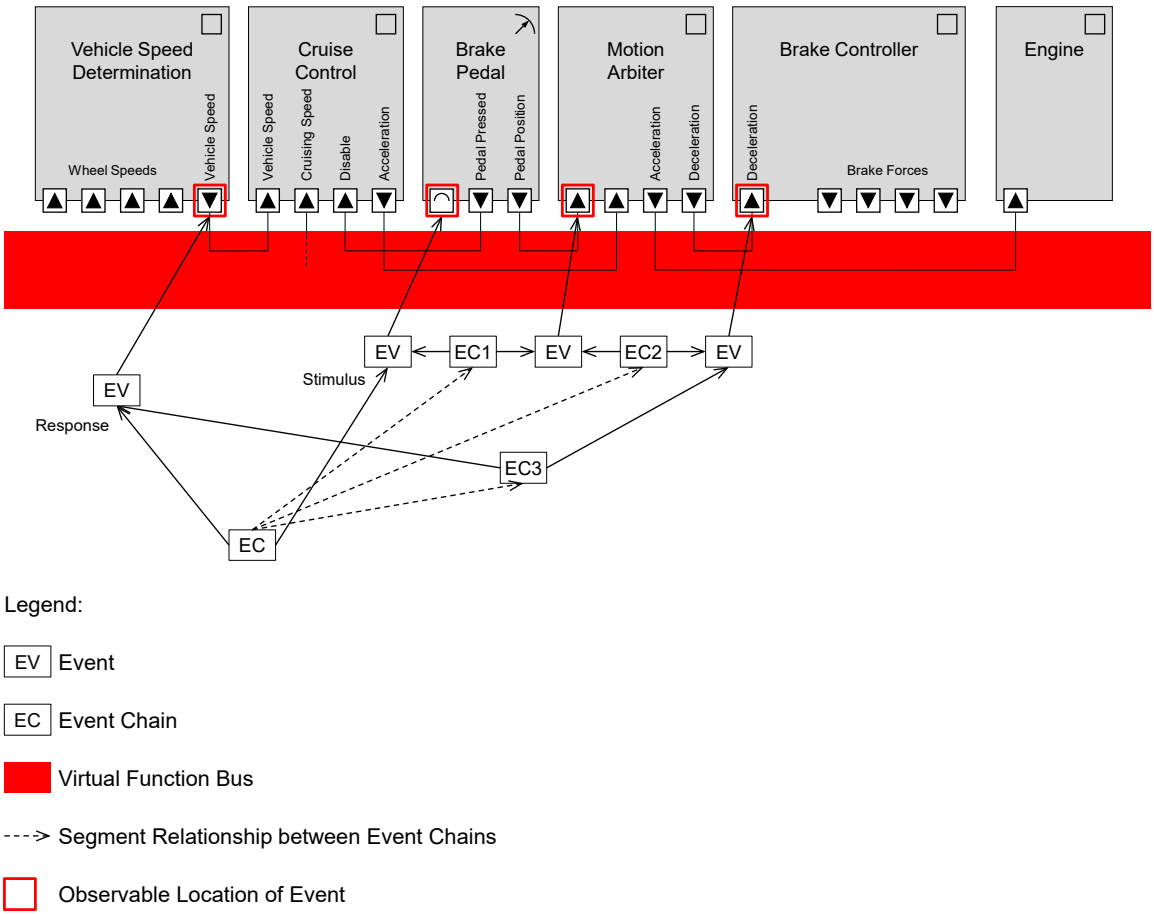


Figure 3.16: Example of a composed and decomposed event chain

3.3.1.3 Patterns

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

3.3.1.3.1 Sequence

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

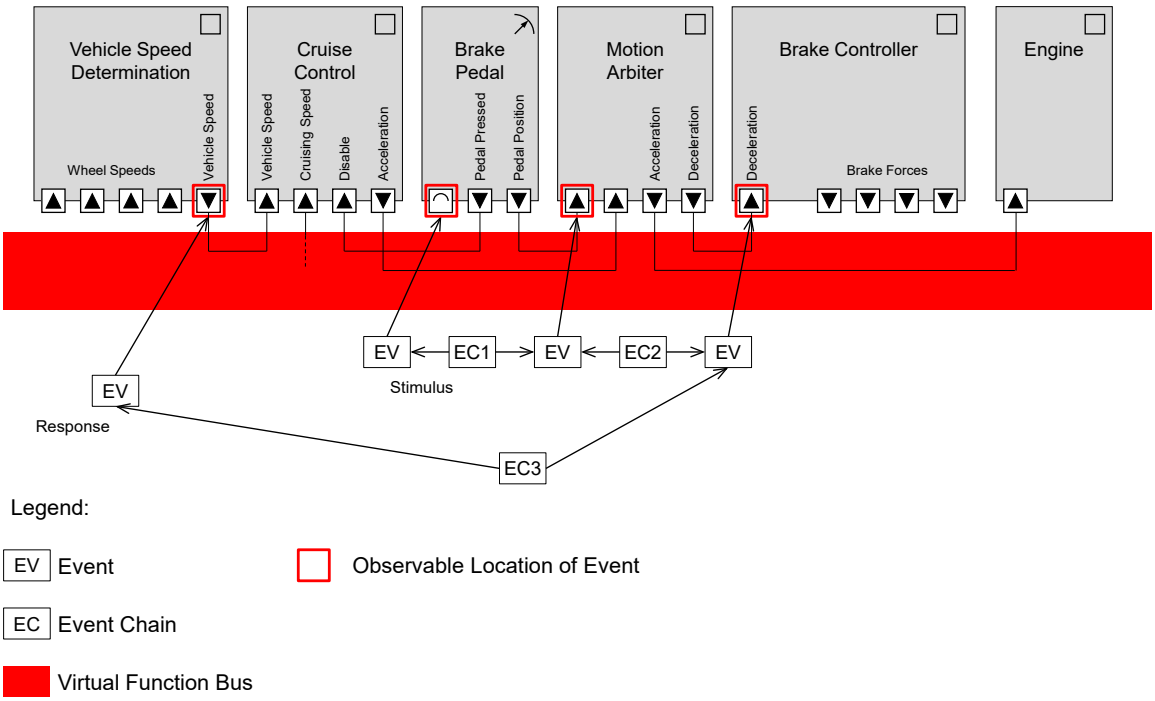


Figure 3.17: Example of the ”Sequence” pattern

An example for this pattern is depicted in Figure 3.17. The event chains *EC1* through *EC3* define a causal relationship of events observed at a port of the SWC called *Brake Pedal* and a port of the SWC called *Vehicle Speed Determination*.

3.3.1.3.2 Fork

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

The pattern is illustrated in Figure 3.18, which shows a path that forks because the SWC *Brake Controller* calculates the brake force value for each wheel (*EC5* through *EC8*).

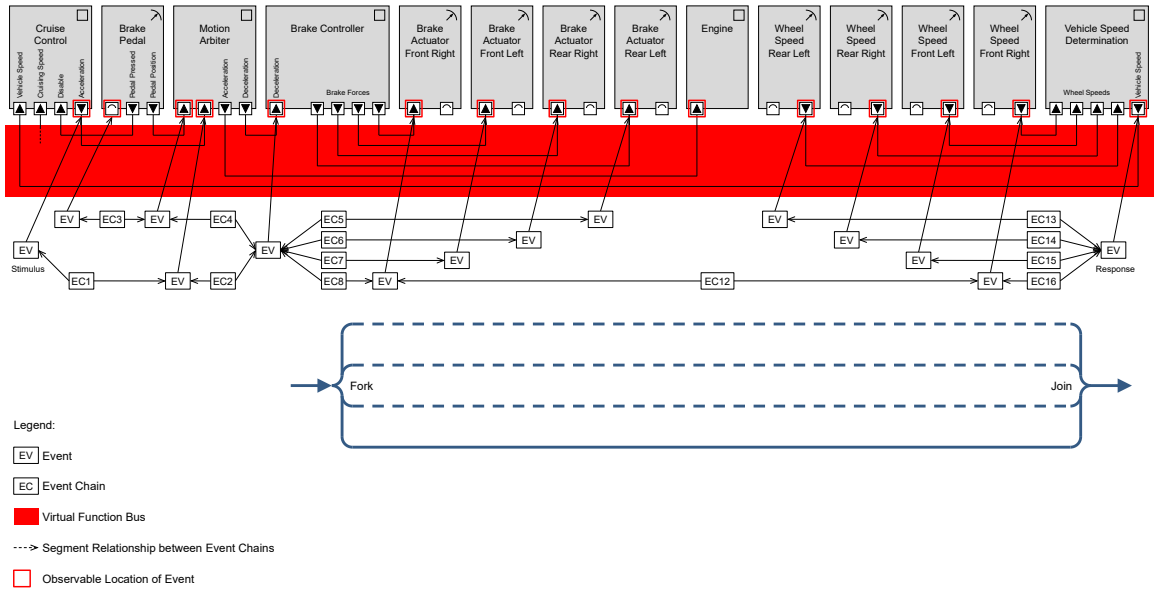


Figure 3.18: Example of the ”Fork” and ”Join” pattern

3.3.1.3.3 Join

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

The pattern is illustrated in Figure 3.18 which shows a path that joins because the SWC *Vehicle Speed Determination* aggregates the wheel speed values from individual wheels (*EC13* through *EC16*).

3.3.1.3.4 Alternative

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

The pattern is illustrated in Figure 3.19 which shows that an event observed at a required port of the SWC *Motion Arbiter* leads to an occurrence of an event either at the port called *Deceleration* of the SWC *Brake Controller*, or at the port called *Acceleration* of the SW-C *Engine*. These alternative causal relationships are described by the event chains *EC2* and *EC4* in this figure. In either case, the deceleration or acceleration of the vehicle leads to the occurrence of an event at the provided port called *Vehicle Speed* of the SW-C *Vehicle Speed Determination* reporting the vehicle’s speed. These alternative causal relationships are described by the event chains *EC3* and *EC5* which both reference the same response event. To fulfill the overall event chain, only one of the alternative paths shall have been occurred.

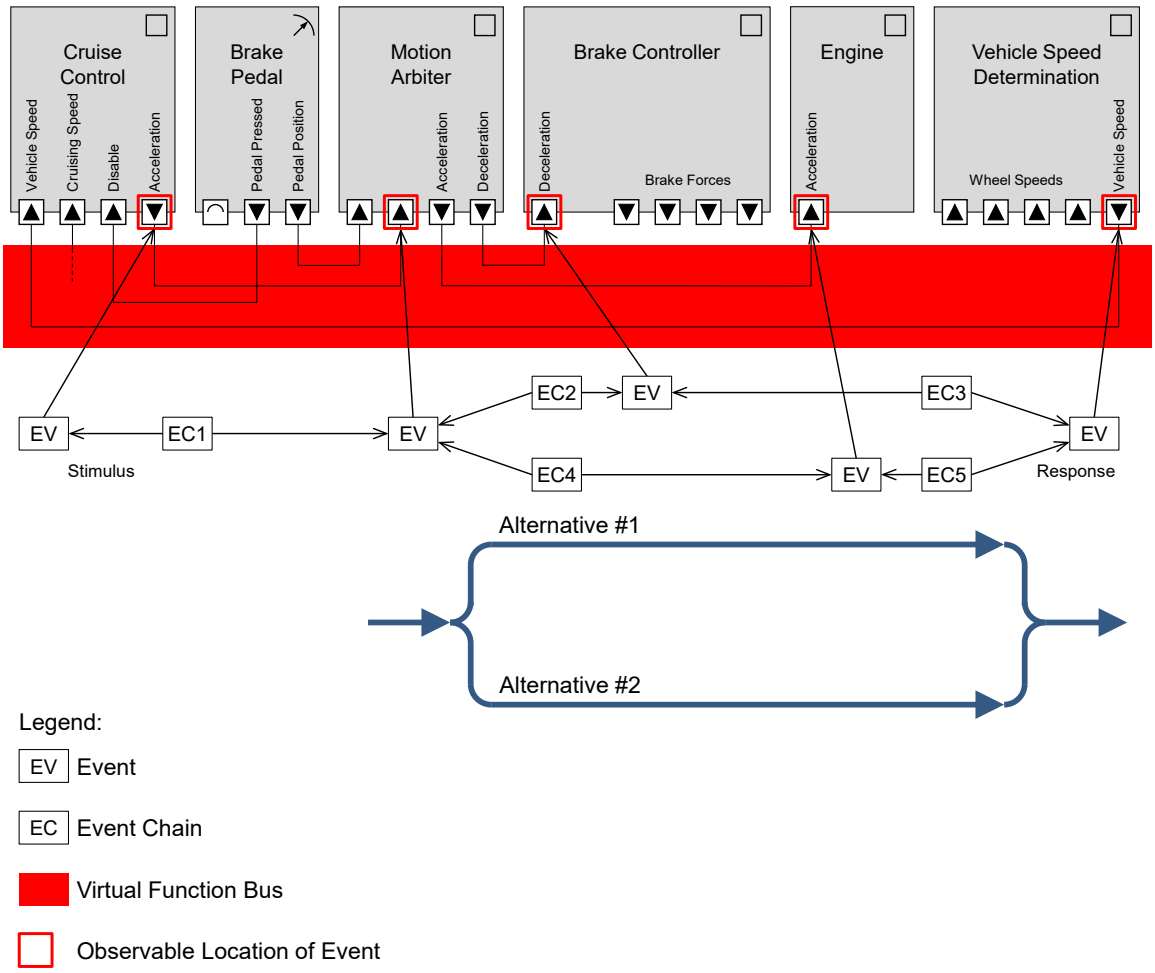


Figure 3.19: Example of the ”Alternative” pattern

3.3.1.3.5 Cycle

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

The pattern is illustrated in Figure 3.20 which shows three event chains *EC8*, *EC12* and *EC17* forming a cycle. The stimulus event of event chain *EC8* is the response event of event chain *EC17*; and the response event of event chain *EC12* is the stimulus event of event chain *EC17*. Event chain *EC8* and *EC12* reference the same event in different roles, namely response event from event chain *EC8* perspective and stimulus event from the event chain *EC12* perspective.

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

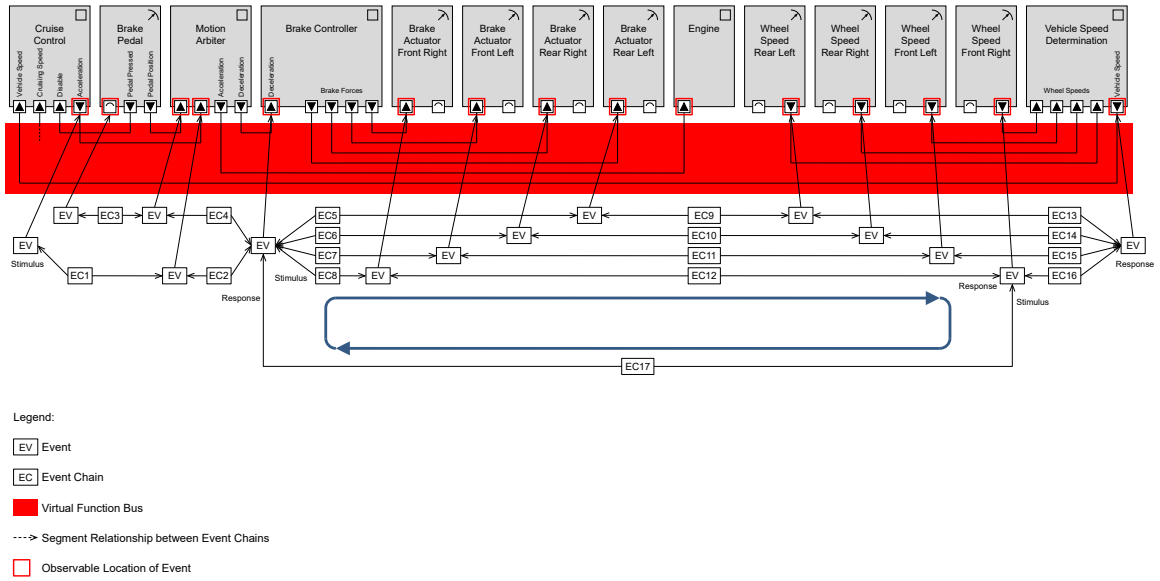


Figure 3.20: Example of the ”Cycle” pattern

3.3.2 TimingDescriptionEvent

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

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

An overview of the different event types is given in Figure 3.21. These are described in more detail in the following sub-sections.

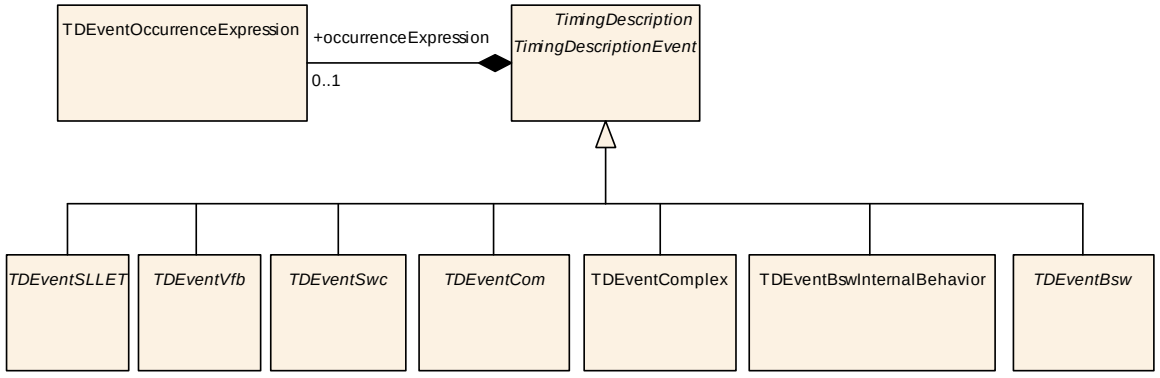


Figure 3.21: Overview of the different types of timing events

Depending on the value of the `category` of the `TimingDescriptionEvent`, it may be used in different use-cases.

[TPS_TIMEX_00056] Standardized `category`s of `TimingDescriptionEvent` in Classic Platform ⌈AUTOSAR standardizes the following `category`s of `TimingDescriptionEvent` and their semantics:

- `undefined`: as per `STANDARD`
- `STANDARD`: No specific semantics are imposed on the `TimingDescriptionEvent`. It indicates the standard behavior.
- `LET_RELEASE`: The `TimingDescriptionEvent` represents the release/start point of an LET interval
- `LET_TERMINATE`: The `TimingDescriptionEvent` represents the termination/end point of an LET interval
- `SL_LET_RELEASE`: The `TimingDescriptionEvent` represents the release/start point of an SL-LET interval
- `SL_LET_TERMINATE`: The `TimingDescriptionEvent` represents the termination/end point of an SL-LET interval
- `DISPATCH_ENTRY_POINT`: The `TimingDescriptionEvent` marks the *dispatcher entry point* of a timing description event chain describing a dispatcher associated with a software cluster
- `DISPATCH_EXIT_POINT`: The `TimingDescriptionEvent` marks the *dispatcher exit point* of a timing description event chain describing a dispatcher associated with a software cluster

⌋*()*

[constr_4559] Restriction of `TimingDescriptionEvent.category` ⌈Any `TimingDescriptionEvent.category` not in the list in [TPS_TIMEX_00056] shall be ignored.⌋*()*

Also note that information regarding the occurrence of a `TimingDescriptionEvent` is described separately in 3.4.2.2.

3.3.2.1 TDEventVfb

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

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

- `VfbTiming` 3.1.1
- `SystemTiming` 3.1.3
- `SwcTiming` 3.1.2

- EcuTiming 3.1.5

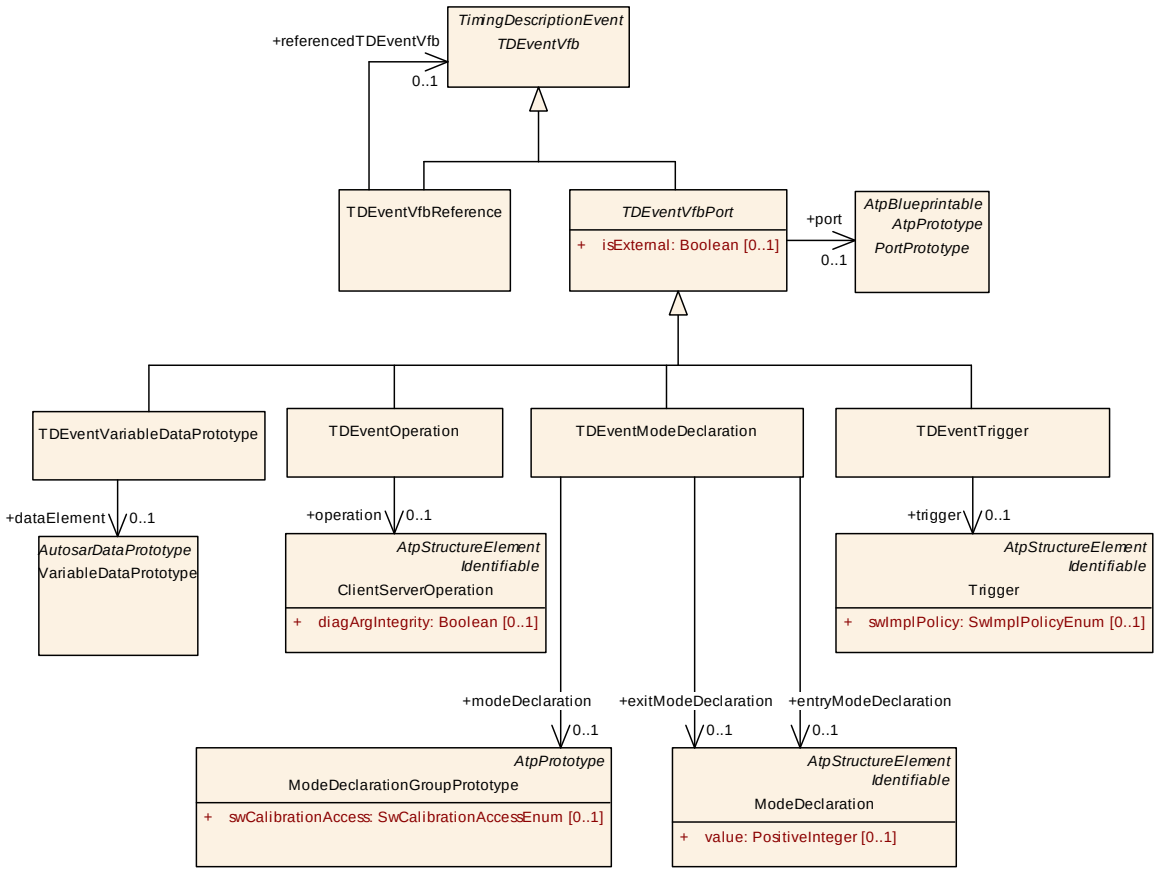


Figure 3.22: VFB events

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

Table 3.14: TDEventVfb

3.3.2.1.1 TDEventVfbReference

[TPS_TIMEX_00043] Purpose of `TDEventVfbReference` ⌈The element `TDEventVfbReference` is used to reference timing description events already specified

in other timing views. In other words, it enables one to re-use existing timing models.⌋ *(RS_TIMEX_00001, RS_TIMEX_00019)*

Class	**TDEventVfbReference**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	This is used to reference timing description events related to the Virtual Functional Bus (VFB) view which are specified in other timing views.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
referenced TDEventVfb	TDEventVfb	0..1	ref	The referenced timing description event.

Table 3.15: TDEventVfbReference

[constr_6886] Existence of `TDEventVfbReference.referencedTDEventVfb` ⌈For each `TDEventVfbReference`, the reference to `TDEventVfb` in the role `referencedTDEventVfb` shall exist **`at the time when the VFB Timing Description is complete`**.⌋*()*

3.3.2.1.2 TDEventVfbPort

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

Class	***TDEventVfbPort*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	This is the abstract parent class to describe specific timing event types at Virtual Functional Bus (VFB) level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	TDEventModeDeclaration, TDEventOperation, TDEventTrigger, TDEventVariableDataPrototype			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
isExternal	Boolean	0..1	attr	This attribute is used to refer to external events that are related to hardware I/O, like physical sensors and actuators, at Virtual Functional Bus (VFB) level.
port	PortPrototype	0..1	ref	port on which the TimingEvent shall apply
portPrototype Blueprint	PortPrototypeBlueprint	0..1	ref	port on which the TimingEvent shall apply (in the context of an AUTOSAR blueprint)

Table 3.16: TDEventVfbPort

[constr_6885] Existence of `TDEventVfbPort.isExternal` ⌈For each `TDEventVfbPort`, the attribute `isExternal` shall exist **`at the time when the VFB Timing Description is complete`**.⌋*()*

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

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

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

3.3.2.1.2.1 TDEventVariableDataPrototype

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

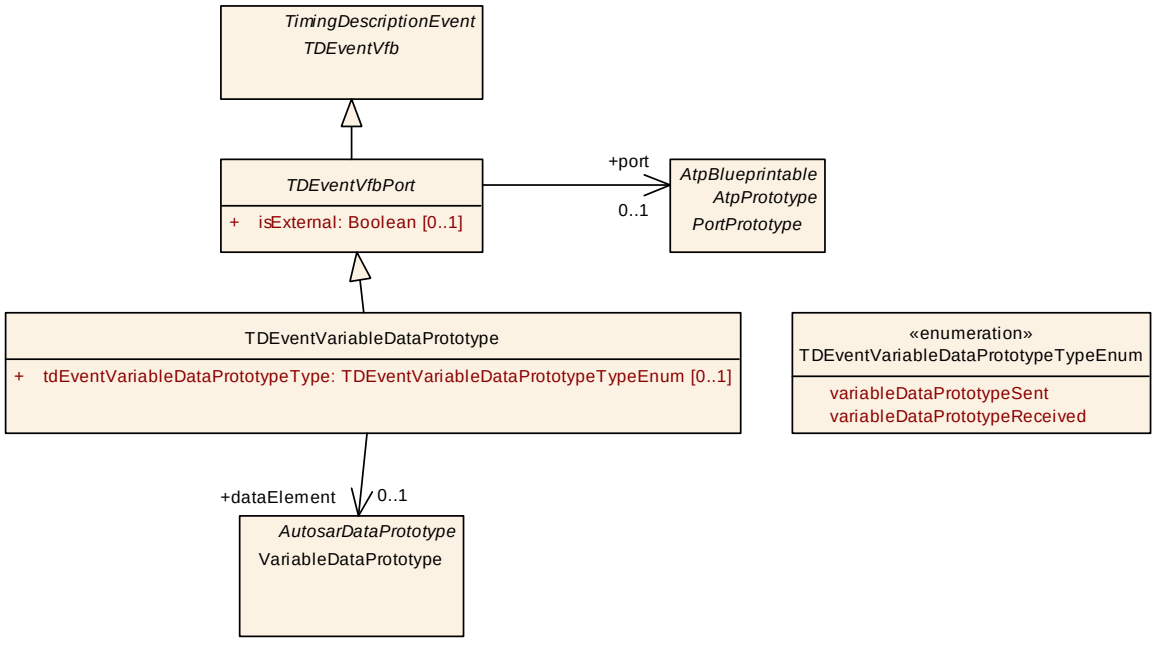


Figure 3.23: Variable Data Prototype

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

▽

△

Class	TDEventVariableDataPrototype			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataElement	VariableDataPrototype	0..1	ref	The referenced VariableDataPrototype
tdEventVariable DataPrototype Type	TDEventVariableData PrototypeTypeEnum	0..1	attr	The specific type of this timing event.

Table 3.17: TDEventVariableDataPrototype

[constr_6887] Existence of `TDEventVariableDataPrototype.tdEventVariableDataPrototypeType` ⌈For each `TDEventVariableDataPrototype`, the attribute `tdEventVariableDataPrototypeType` shall exist **at the time when the VFB Timing Description is complete**.⌋*()*

[constr_6888] Existence of `TDEventVariableDataPrototype.dataElement` ⌈For each `TDEventVariableDataPrototype`, the reference to `VariableDataPrototype` in the role `dataElement` shall exist **at the time when the VFB Timing Description is complete**.⌋*()*

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

Table 3.18: TDEventVariableDataPrototypeTypeEnum

3.3.2.1.2.2 TDEventOperation

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

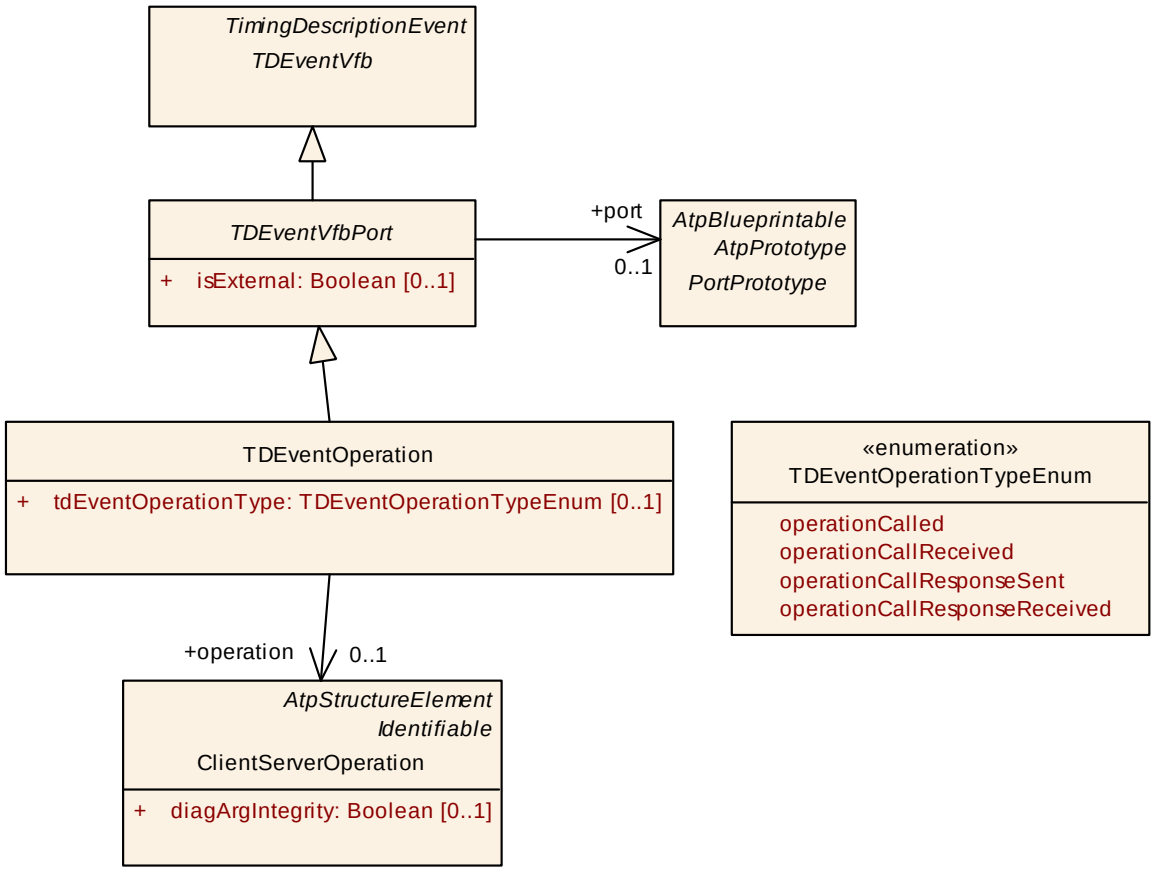


Figure 3.24: Operation

Class	**TDEventOperation**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	This is used to describe timing events related to client-server communication at VFB level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TDEventVfbPort*, *Timing Description*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
operation	ClientServerOperation	0..1	ref	The referenced operation.
tdEvent OperationType	TDEventOperationType Enum	0..1	attr	The specific type of this timing event.

Table 3.19: TDEventOperation

[constr_6889] Existence of `TDEventOperation.tdEventOperationType` ⌈For each `TDEventOperation`, the attribute `tdEventOperationType` shall exist **`at the time when the VFB Timing Description is complete`**.⌋*()*

[constr_6890] Existence of `TDEventOperation.operation` ⌈For each `TDEventOperation`, the reference to `ClientServerOperation` in the role `operation` shall exist **`at the time when the VFB Timing Description is complete`**.⌋*()*

Enumeration	**TDEventOperationTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb
Note	This is used to describe the specific event type of a TDEventOperation.
Aggregated by	TDEventOperation.tdEventOperationType
Literal	***Description***
operationCalled	A point in time where the referenced operation is called by the client SWC. **Tags:** atp.EnumerationLiteralIndex=0
operationCall Received	A point in time where the call of the referenced operation is received by the server SWC. **Tags:** atp.EnumerationLiteralIndex=1
operationCall ResponseReceived	A point in time where the client SWC has received the response of the referenced operation call. **Tags:** atp.EnumerationLiteralIndex=2
operationCall ResponseSent	A point in time where the server SWC has terminated with the execution of the referenced operation, and has sent out a response. **Tags:** atp.EnumerationLiteralIndex=3

Table 3.20: TDEventOperationTypeEnum

3.3.2.1.2.3 TDEventModeDeclaration

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

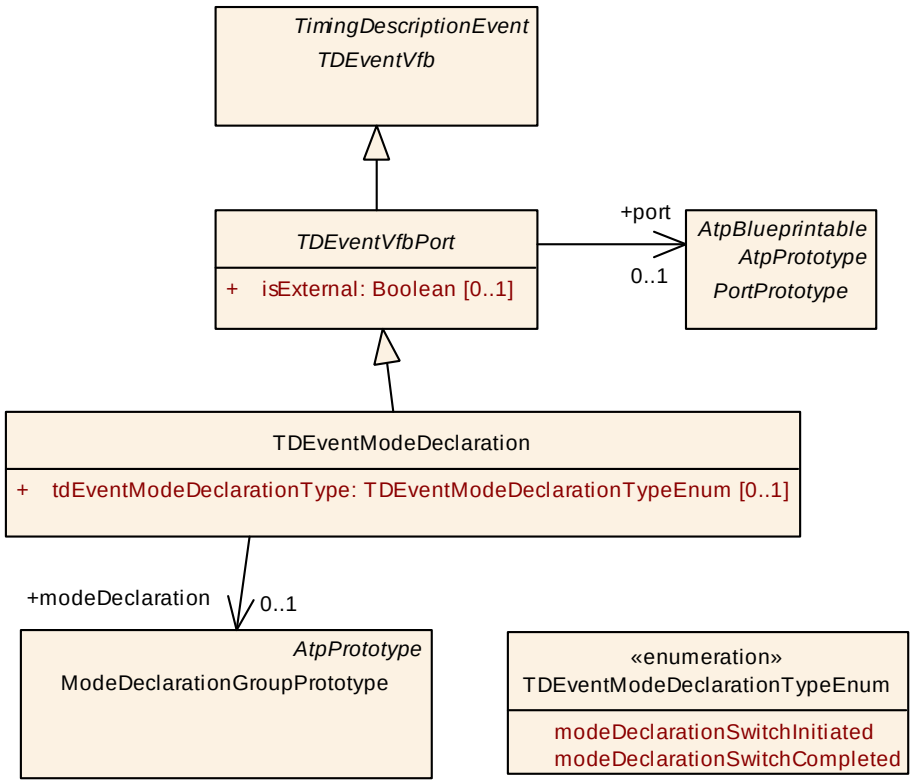


Figure 3.25: Mode Declaration

Class	**TDEventModeDeclaration**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	This is used to describe timing events related to mode switch communication at VFB level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TDEventVfbPort*, *Timing Description*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
entryMode Declaration	ModeDeclaration	0..1	ref	Optional parameter which refines the scope of the TDEventModeDeclaration. If the parameter is set, the event occurs only if the mode declaration group prototype instance shall enter into the referenced ModeDeclaration.
exitMode Declaration	ModeDeclaration	0..1	ref	Optional parameter which refines the scope of the TDEventModeDeclaration. If the parameter is set, the event occurs only if the mode declaration group prototype instance shall exit from the referenced ModeDeclaration.
mode Declaration	ModeDeclarationGroup Prototype	0..1	ref	The referenced mode declaration group prototype.
tdEventMode DeclarationType	TDEventMode DeclarationTypeEnum	0..1	attr	The specific type of this timing event.

Table 3.21: TDEventModeDeclaration

[constr_6891] Existence of `TDEventModeDeclaration.tdEventModeDeclarationType` ⌈For each `TDEventModeDeclaration`, the attribute `tdEventModeDeclarationType` shall exist **`at the time when the VFB Timing Description is complete`**.⌋*()*

[constr_6892] Existence of `TDEventModeDeclaration.modeDeclaration` ⌈For each `TDEventModeDeclaration`, the reference to `ModeDeclarationGroupPrototype` in the role `modeDeclaration` shall exist **`at the time when the VFB Timing Description is complete`**.⌋*()*

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

Table 3.22: TDEventModeDeclarationTypeEnum

3.3.2.1.2.4 TDEventTrigger

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

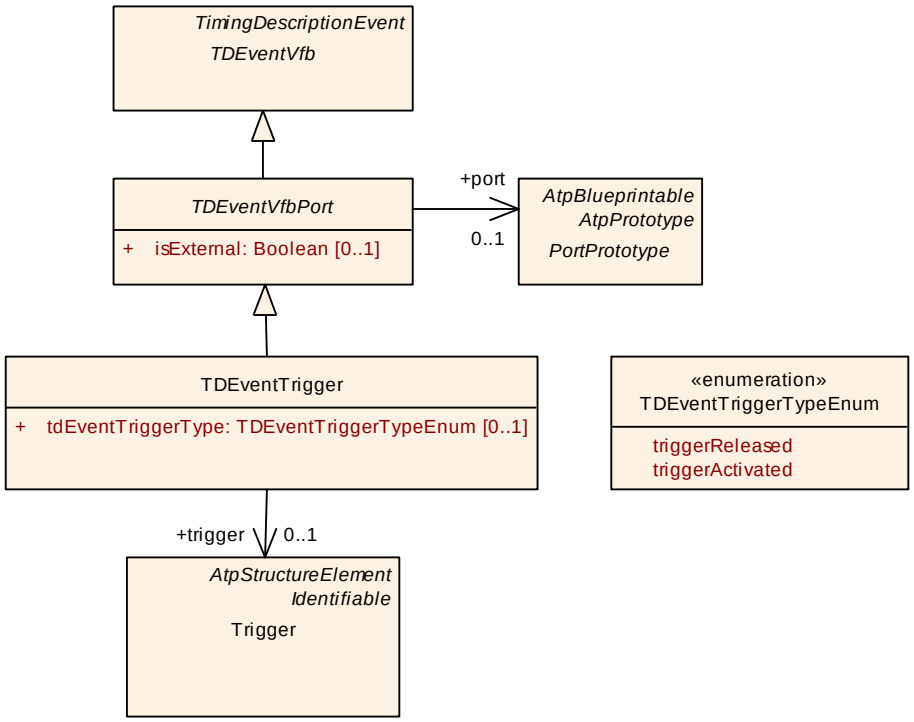


Figure 3.26: Trigger

Class	**TDEventTrigger**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	This is used to describe timing events related to triggers at VFB level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TDEventVfbPort*, *Timing Description*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tdEventTrigger Type	TDEventTriggerType Enum	0..1	attr	The specific type of this timing event.
trigger	Trigger	0..1	ref	The trigger which is provided (released) or required (activate) in the given context.

Table 3.23: TDEventTrigger

[constr_6893] **Existence of `TDEventTrigger.tdEventTriggerType`** ⌈For each `TDEventTrigger`, the attribute `tdEventTriggerType` shall exist **`at the time when the VFB Timing Description is complete`**.⌋ *()*

[constr_6894] Existence of `TDEventTrigger.trigger` ⌈For each `TDEventTrigger`, the reference to `Trigger` in the role `trigger` shall exist **at the time when the VFB Timing Description is complete**.⌋ *()*

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

Table 3.24: TDEventTriggerTypeEnum

3.3.2.1.2.5 Blueprinting TDEventVfbPort

The primary purpose of blueprinting `VfbTiming` is for use with a "AUTOSAR Application Interface" [5]. In a blueprint of an "AUTOSAR Application Interface", a `TimingConstraint` can be applied to `TDEventVfb` to annotate a high-level timing goal for an blueprinted `PortPrototype`. The reference `portPrototypeBlueprint` refers to a `PortPrototypeBlueprint` which effectively wraps a `PortPrototype` used in the context of an AUTOSAR Blueprint, but, permits certain extra aspects over a `PortPrototype` as explained in "Blueprinting PortPrototype" [3].

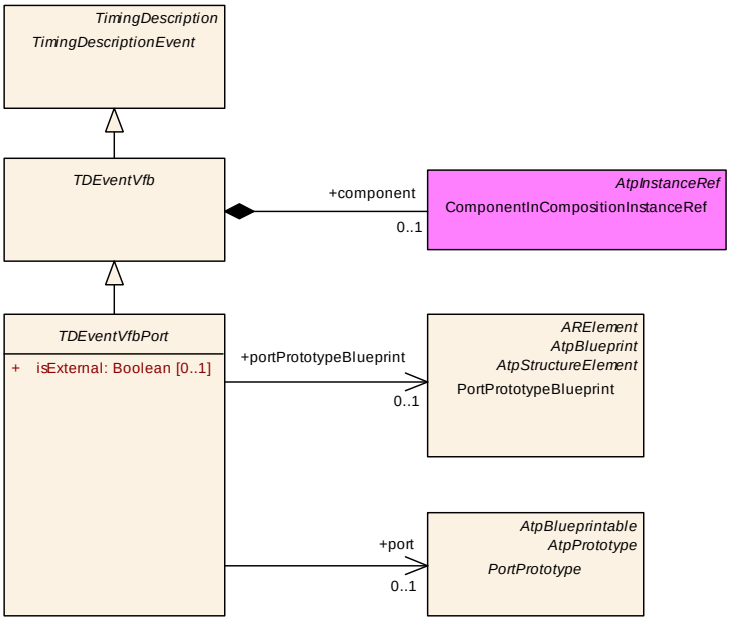


Figure 3.27: `TDEventVfb` Blueprint

[constr_4508] Existence of TDEventVfbPort.portPrototypeBlueprint ⌈The reference TDEventVfbPort.portPrototypeBlueprint shall exist only if the immediate parent is ARPackage.category==BLUEPRINT, **at the time when the VFB Timing Description is complete**.⌋*()*

[constr_6900]{DRAFT} **Dual existence of TDEventVfb.port and TDEventVfb.portPrototypeBlueprint** ⌈The reference TDEventVfbPort.port and TDEventVfbPort.portPrototypeBlueprint shall not co-exist in a model, **at the time when the VFB Timing Description is complete**.⌋*()*

3.3.2.2 TDEventSwc

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

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

Table 3.25: TDEventSwc

3.3.2.2.1 TDEventSwcInternalBehavior

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

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

- SwcTiming 3.1.2
- SystemTiming 3.1.3
- EcuTiming 3.1.5

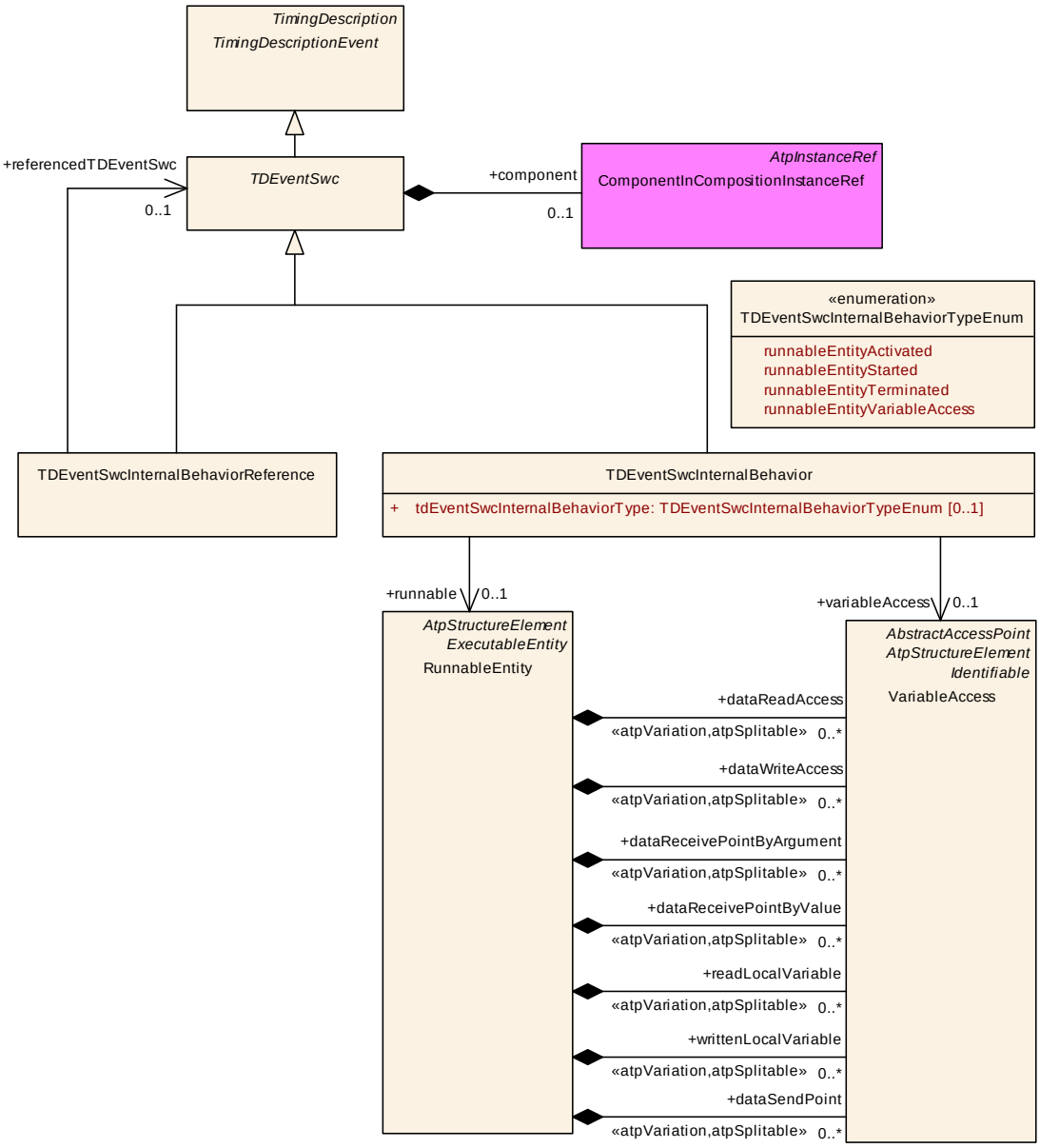


Figure 3.28: Event of type ”SwcInternalBehavior”

Class	**TDEventSwcInternalBehavior**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventSwcInternalBehavior			
Note	This is used to describe timing events related to the SwcInternalBehavior of an AtomicSwComponent Type.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventSwc*, *TimingDescription*, *Timing DescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
runnable	RunnableEntity	0..1	ref	The scope of this timing event.

△

Class	**TDEventSwcInternalBehavior**			
tdEventSwc Internal BehaviorType	TDEventSwcInternal BehaviorTypeEnum	0..1	attr	The specific type of this timing event.
variableAccess	VariableAccess	0..1	ref	The scope of this timing event.

Table 3.26: TDEventSwcInternalBehavior

[constr_6882] Existence of `TDEventSwcInternalBehavior.tdEventSwcInternalBehaviorType` ⌈For each `TDEventSwcInternalBehavior`, the attribute `tdEventSwcInternalBehaviorType` shall exist **at the time when the Swc Timing Description is complete**.⌋*()*

[constr_6883] Existence of `TDEventSwcInternalBehavior.runnable` ⌈For each `TDEventSwcInternalBehavior`, the reference to `RunnableEntity` in the role `runnable` shall exist **at the time when the Swc Timing Description is complete**.⌋*()*

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

Table 3.27: TDEventSwcInternalBehaviorTypeEnum

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

[constr_4511] Validity of referencing `RunnableEntity` ⌈A `RunnableEntity` shall be referenced if and only if the value of `tdEventSwcInternalBehaviorType` is either:

- `runnableEntityActivated`
- `runnableEntityStarted`

- runnableEntityTerminated
- runnableEntityVariableAccess

⌋()

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

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

Class	**TDEventSwcInternalBehaviorReference**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventSwcInternalBehavior			
Note	This is used to reference timing description events related to the Software Component (SW-C) view which are specified in other timing views.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventSwc*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
referenced TDEventSwc	TDEventSwc	0..1	ref	The referenced timing description event.

Table 3.28: TDEventSwcInternalBehaviorReference

[constr_6884] Existence of TDEventSwcInternalBehaviorReference.referencedTDEventSwc ⌈For each TDEventSwcInternalBehaviorReference, the reference to TDEventSwc in the role referencedTDEventSwc shall exist **at the time when the Swc Timing Description is complete**.⌋()

3.3.2.3 TDEventCom

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

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

- SystemTiming 3.1.2
- EcuTiming 3.1.5

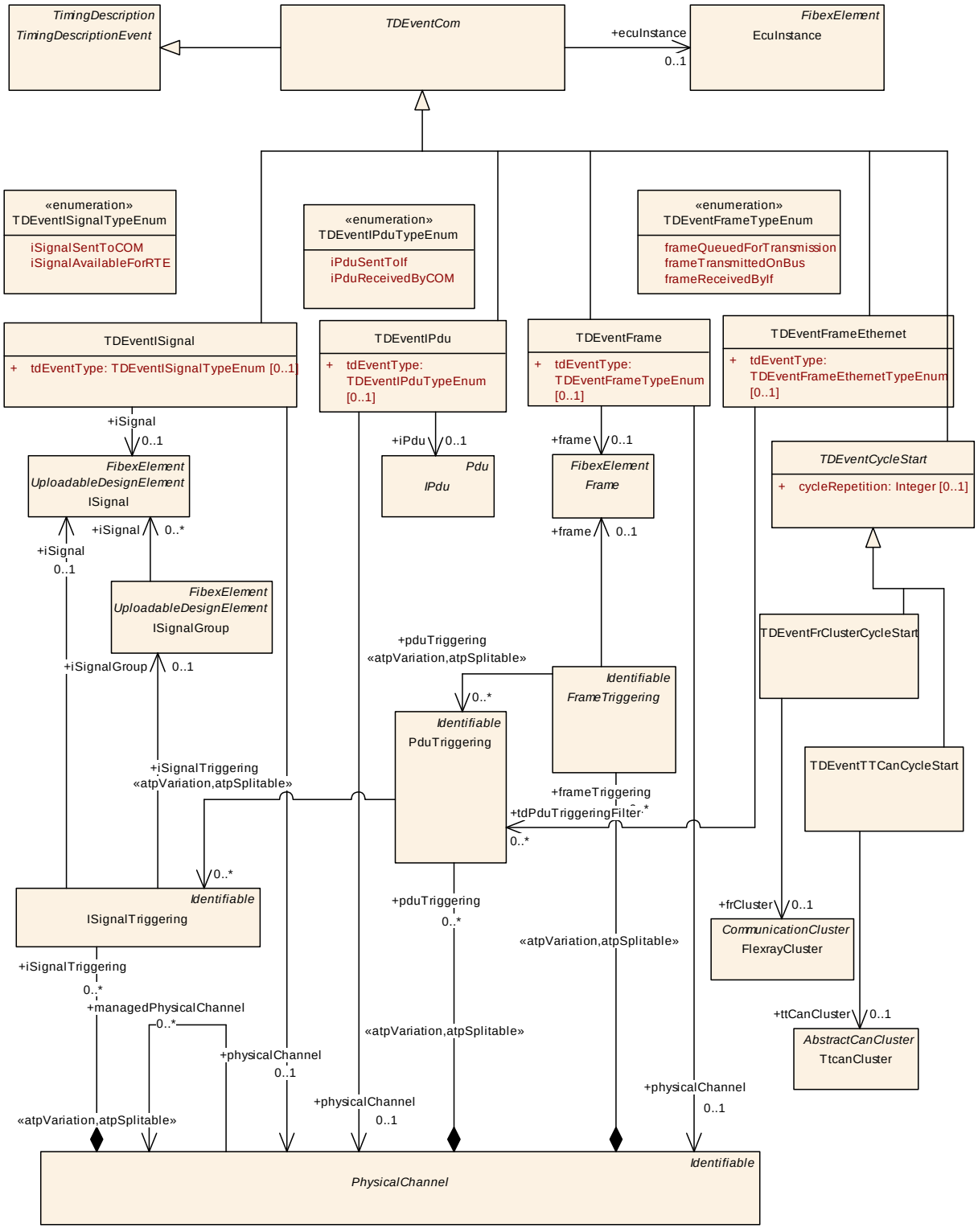


Figure 3.29: Events regarding communication

Class	**TDEventCom** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is the abstract parent class to describe timing events related to communication including the physical layer.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	*TDEventCycleStart*, TDEventFrame, TDEventFrameEthernet, TDEventIPdu, TDEventISignal			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	**Type**	**Mult.**	**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 3.29: TDEventCom

3.3.2.3.1 TDEventISignal

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

Class	**TDEventISignal**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is used to describe timing events related to the exchange of I-Signals between COM and RTE.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
iSignal	ISignal	0..1	ref	The scope of this timing event.
physical Channel	PhysicalChannel	0..1	ref	The PhysicalChannel on which the ISignal is transmitted.
tdEventType	TDEventISignalType Enum	0..1	attr	The specific type of this timing event.

Table 3.30: TDEventISignal

[constr_6915]{DRAFT} **Affinity of `ISignal` in `TDEventISignal`** ⌈The referenced `ISignal` in the role `TDEventISignal.iSignal` shall exist also in the list of `iSignal`s aggregated by `TDEventISignal.physicalChannel.iSignalTriggering` **at the time when the Ecu Timing Description is complete**.⌋*()*

[constr_6864] Existence of `TDEventISignal.tdEventType` ⌈For each `TDEventISignal`, the attribute `tdEventType` shall exist **at the time when the Ecu Timing Description is complete**.⌋*()*

[constr_6865] Existence of `TDEventISignal.iSignal` ⌈For each `TDEventISignal`, the reference to `ISignal` in the role `iSignal` shall exist **at the time when the Ecu Timing Description is complete**.⌋*()*

[constr_6866] Existence of `TDEventISignal.physicalChannel` ⌈For each `TDEventISignal`, the reference to `PhysicalChannel` in the role `physicalChannel` shall exist **`at the time when the Ecu Timing Description is complete`**.⌋*()*

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

Table 3.31: TDEventISignalTypeEnum

3.3.2.3.2 TDEventIPdu

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

Class	**TDEventIPdu**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventCom			
Note	This is used to describe timing events related to the exchange of I-PDUs between the bus specific (Flex Ray / CAN / LIN) Interface BSW module and COM.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iPdu	IPdu	0..1	ref	The scope of this timing event.
physical Channel	PhysicalChannel	0..1	ref	The PhysicalChannel on which the IPdu is transmitted.
tdEventType	TDEventIPduTypeEnum	0..1	attr	The specific type of this timing event.

Table 3.32: TDEventIPdu

[constr_6917]{DRAFT} **Affinity of `IPdu` in `TDEventIPdu`** ⌈The referenced `IPdu` in the role `TDEventIPdu.iPdu` shall exist also in the list of `iPdu`s aggregated by `TDEventIPdu.physicalChannel.pduTriggering` **`at the time when the Ecu Timing Description is complete`**.⌋*()*

[constr_6867] Existence of `TDEventIPdu.tdEventType` ⌈For each `TDEventIPdu`, the attribute `tdEventType` shall exist **at the time when the Ecu Timing Description is complete**.⌋*()*

[constr_6868] Existence of `TDEventIPdu.iPdu` ⌈For each `TDEventIPdu`, the reference to `IPdu` in the role `iPdu` shall exist **at the time when the Ecu Timing Description is complete**.⌋*()*

[constr_6869] Existence of `TDEventIPdu.physicalChannel` ⌈For each `TDEventIPdu`, the reference to `PhysicalChannel` in the role `physicalChannel` shall exist **at the time when the Ecu Timing Description is complete**.⌋*()*

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

Table 3.33: TDEventIPduTypeEnum

3.3.2.3.3 TDEventFrame

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

Class	**TDEventFrame**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is used to describe timing events related to the exchange of frames between the communication controller and the bus specific (FlexRay / CAN / LIN) Interface BSW module.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
frame	Frame	0..1	ref	The scope of this timing event.

▽

△

Class	**TDEventFrame**			
physical Channel	PhysicalChannel	0..1	ref	The PhysicalChannel on which the Frame is transmitted.
tdEventType	TDEventFrameType Enum	0..1	attr	The specific type of this timing event.

Table 3.34: TDEventFrame

[constr_6916]{DRAFT} **Affinity of `Frame` in `TDEventFrame`** ⌈The referenced `Frame` in the role `TDEventFrame.frame` shall exist also in the list of `frame`s aggregated by `TDEventFrame.physicalChannel.frameTriggering` **at the time when the Ecu Timing Description is complete**.⌋*()*

[constr_6870] Existence of `TDEventFrame.tdEventType` ⌈For each `TDEventFrame`, the attribute `tdEventType` shall exist **at the time when the Ecu Timing Description is complete**.⌋*()*

[constr_6871] Existence of `TDEventFrame.frame` ⌈For each `TDEventFrame`, the reference to `Frame` in the role `frame` shall exist **at the time when the Ecu Timing Description is complete**.⌋*()*

[constr_6872] Existence of `TDEventFrame.physicalChannel` ⌈For each `TDEventFrame`, the reference to `PhysicalChannel` in the role `physicalChannel` shall exist **at the time when the Ecu Timing Description is complete**.⌋*()*

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

Table 3.35: TDEventFrameTypeEnum

3.3.2.3.4 TDEventComEthernet

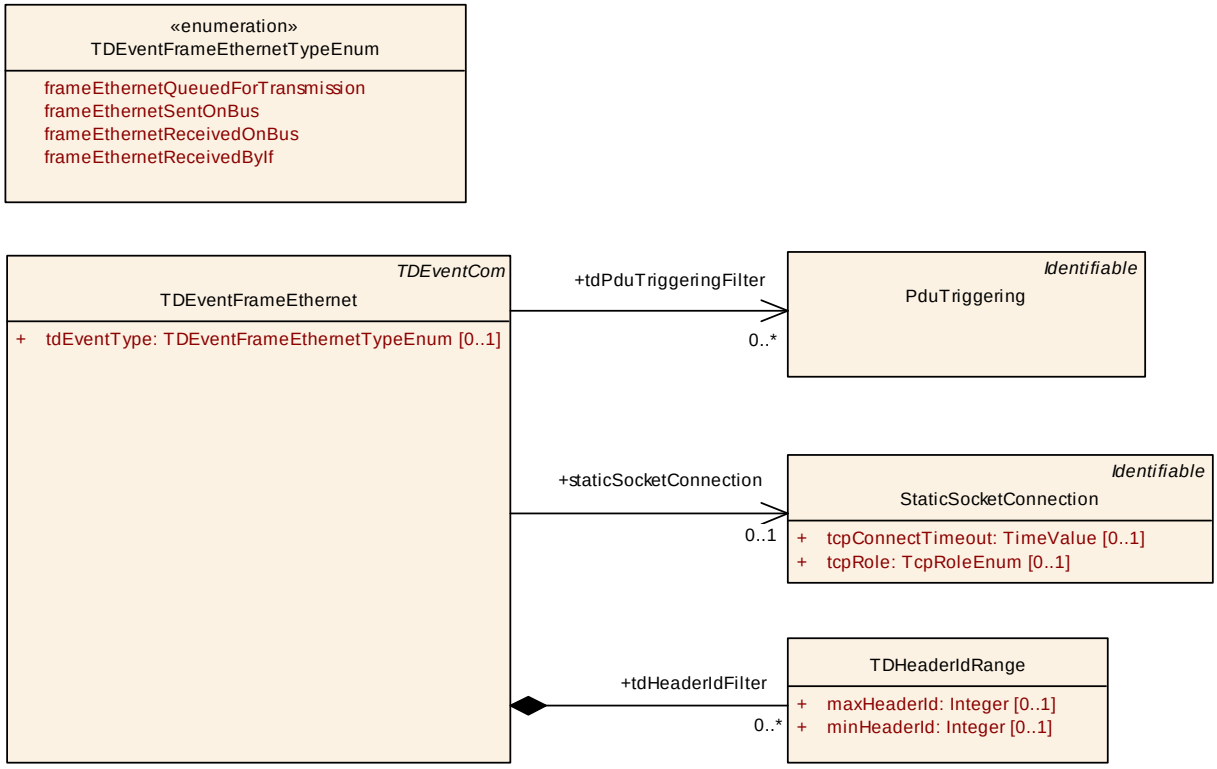


Figure 3.30: Events regarding Ethernet communication

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

Class	**TDEventFrameEthernet**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is used to describe timing description events related to the exchange of Ethernet frames between an Ethernet communication controller and the BSW Ethernet interface and driver module.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
staticSocket Connection	StaticSocketConnection	0..1	ref	Specifies the SocketConnection by the means of which Physical Data Units (PDU) are transmitted or received within an Ethernet Frame.
tdEventType	TDEventFrameEthernet TypeEnum	0..1	attr	This is used to describe the specific event type of a TDEventFrameEthernet.
tdHeaderIdFilter	TDHeaderIdRange	*	aggr	Specifies the header identifier or a range of header identifiers that if contained in the Ethernet frame let the TDEventFrameEthernet occur.
tdPduTriggering Filter	PduTriggering	*	ref	Specifies the PDU that if contained in the Ethernet frame let the TDEventFrameEthernet occur.

Table 3.36: TDEventFrameEthernet

[constr_6873] Existence of TDEventFrameEthernet.tdEventType ⌈For each TDEventFrameEthernet, the attribute tdEventType shall exist **at the time when the Ecu Timing Description is complete**.⌋()

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

Table 3.37: TDEventFrameEthernetTypeEnum

Class	**TDHeaderIdRange**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	Specifies a range of PDU header identifiers. This range is specified by a minimum and maximum header identifier; and the maximum header identifier shall be greater than or equal the minimum header identifier.			
Base	*ARObject*			
Aggregated by	TDEventFrameEthernet.tdHeaderIdFilter			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
maxHeaderId	Integer	0..1	attr	Specifies the maximum PDU header identifier, in other words the upper bound of a range of PDU header identifiers.
minHeaderId	Integer	0..1	attr	Specifies the minimum PDU header identifier, in other words the lower bound of a range of PDU header identifiers.

Table 3.38: TDHeaderIdRange

[constr_6874] Existence of TDHeaderIdRange.maxHeaderId ⌈For each TDHeaderIdRange, the attribute maxHeaderId shall exist **at the time when the Ecu Timing Description is complete**.⌋()

[constr_6875] Existence of TDHeaderIdRange.minHeaderId ⌈For each TDHeaderIdRange, the attribute minHeaderId shall exist **at the time when the Ecu Timing Description is complete**.⌋()

3.3.2.3.5 TDEventCycleStart

[constr_6876] Existence of `TDEventCycleStart.cycleRepetition` ⌈For each `TDEventCycleStart`, the attribute `cycleRepetition` shall exist **`at the time when the Ecu Timing Description is complete`**.⌋*()*

Class	***TDEventCycleStart*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is the abstract parent class to describe timing events related to a point in time where a communication cycle starts. Via the attribute "cycleRepetition", a filtered view to the cycle start can be defined.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	TDEventFrClusterCycleStart, TDEventTTCanCycleStart			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cycleRepetition	Integer	0..1	attr	The start of every <cycleRepetition> cycle is targeted by this event.

Table 3.39: TDEventCycleStart

3.3.2.3.5.1 TDEventFrClusterCycleStart

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

Class	**TDEventFrClusterCycleStart**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is used to describe the timing event related to a point in time where a communication cycle starts on a FlexRay cluster.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TDEventCycleStart*, *Timing Description*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
frCluster	FlexrayCluster	0..1	ref	The scope of this timing event.

Table 3.40: TDEventFrClusterCycleStart

[constr_6877] Existence of `TDEventFrClusterCycleStart.frCluster` ⌈For each `TDEventFrClusterCycleStart`, the attribute `frCluster` shall exist **`at the time when the Ecu Timing Description is complete`**.⌋*()*

3.3.2.3.5.2 TDEventTTCanCycleStart

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

Class	**TDEventTTCanCycleStart**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventCom			
Note	This is used to describe the timing event related to a point in time where a communication cycle starts on a TTCAN cluster.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventCom*, *TDEventCycleStart*, *Timing Description*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ttCanCluster	TtcanCluster	0..1	ref	The scope of this timing event.

Table 3.41: TDEventTTCanCycleStart

[constr_6878] Existence of `TDEventTTCanCycleStart.ttCanCluster` ⌈For each `TDEventTTCanCycleStart`, the attribute `ttCanCluster` shall exist **`at the time when the Ecu Timing Description is complete`**.⌋*()*

3.3.2.4 TDEventBswInternalBehavior

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

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

- `BswModuleTiming` 3.1.4.1
- `EcuTiming` 3.1.5

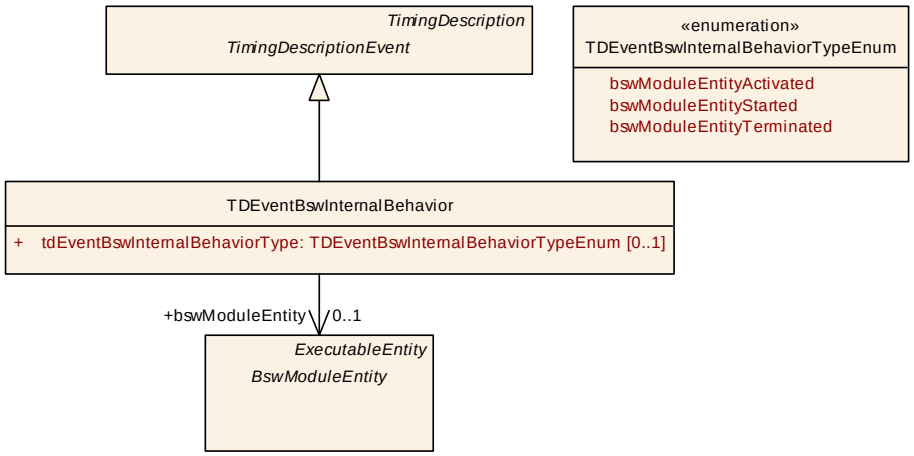


Figure 3.31: Events related to the internal structure of a BSW module

Class	**TDEventBswInternalBehavior**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventBswInternalBehavior			
Note	This is used to describe timing events related to the BswInternalBehavior of a BSW module.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bswModuleEntity	BswModuleEntity	0..1	ref	The scope of this timing event.
tdEventBswInternalBehaviorType	TDEventBswInternalBehaviorTypeEnum	0..1	attr	The specific type of this timing event.

Table 3.42: TDEventBswInternalBehavior

[constr_6858] Existence of `TDEventBswInternalBehavior.tdEventBswInternalBehaviorType` ⌈For each `TDEventBswInternalBehavior`, the attribute `tdEventBswInternalBehaviorType` shall exist **`at the time when the Bsw Timing Description is complete`**.⌋*()*

[constr_6859] Existence of `TDEventBswInternalBehavior.bswModuleEntity` ⌈For each `TDEventBswInternalBehavior`, the reference to `BswModuleEntity` in the role `bswModuleEntity` shall exist **`at the time when the Bsw Timing Description is complete`**.⌋*()*

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

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

Table 3.43: TDEventBswInternalBehaviorTypeEnum

3.3.2.5 TDEventBsw

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

[constr_6901]{DRAFT} **Existence of `TDEventBsw.bswModuleDescription`** ⌈For each `BswModuleTiming`, the reference to a `BswModuleDescription` in the role `bswModuleDescription` shall exist **`at the time when the Bsw Timing Description is complete`**.⌋ *()*

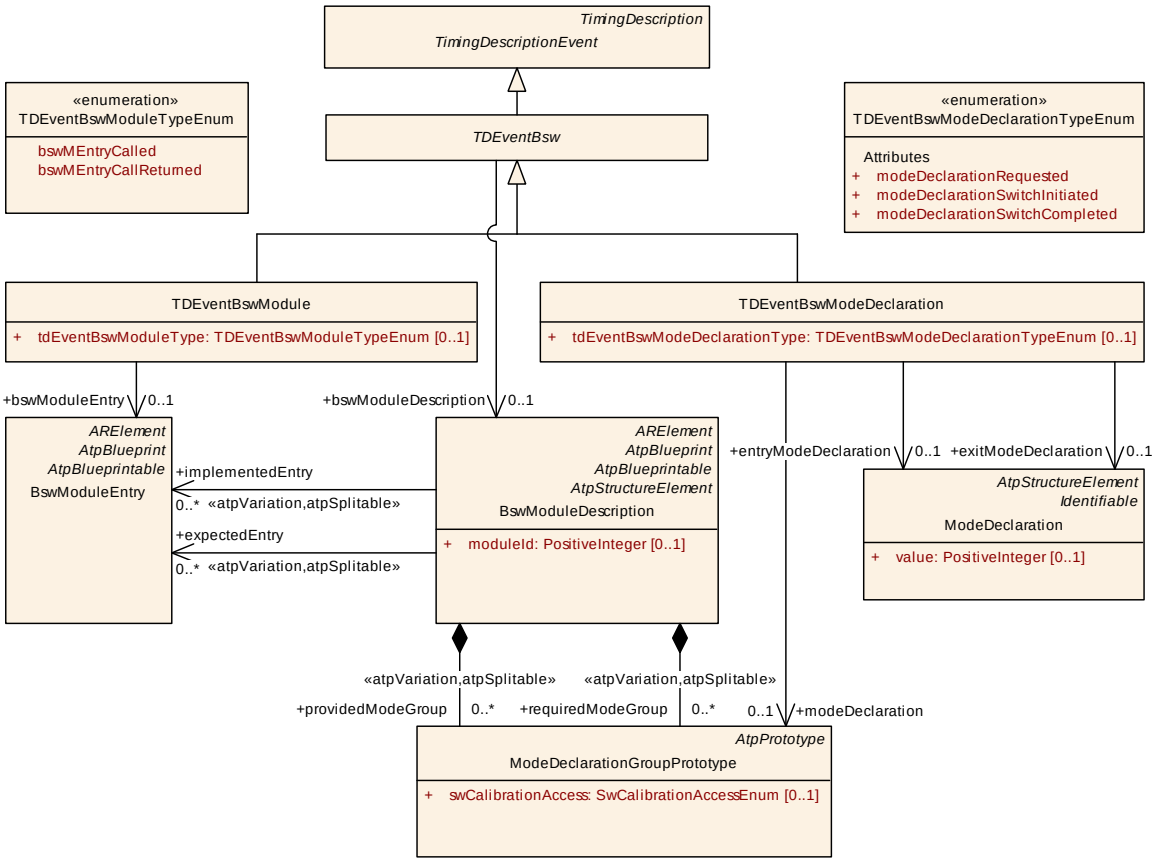


Figure 3.32: Events dealing with inter BSW module relations and mode communications on BSW level

3.3.2.5.1 TDEventBswModule

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

Class	**TDEventBswModule**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventBsw			
Note	This is used to describe timing events related to the interaction between BSW modules.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventBsw*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bswModuleEntry	BswModuleEntry	0..1	ref	The scope of this timing event.
tdEventBswModuleType	TDEventBswModuleTypeEnum	0..1	attr	The specific type of this timing event.

Table 3.44: TDEventBswModule

[constr_6860] Existence of `TDEventBswModule.tdEventBswModuleType` ⌈For each `TDEventBswModule`, the attribute `tdEventBswModuleType` shall exist **`at the time when the Bsw Timing Description is complete`**.⌋*()*

[constr_6861] Existence of `TDEventBswModule.bswModuleEntry` ⌈For each `TDEventBswModule`, the reference to `BswModuleEntry` in the role `bswModuleEntry` shall exist **`at the time when the Bsw Timing Description is complete`**.⌋*()*

Enumeration	**TDEventBswModuleTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventBsw
Note	This is used to describe the specific event type of a TDEventBswModule.
Aggregated by	TDEventBswModule.tdEventBswModuleType
Literal	***Description***
bswMEntryCalled	A point in time where the associated BswModuleEntry has been called. **Tags:** atp.EnumerationLiteralIndex=0
bswMEntryCall Returned	A point in time where the call of the associated BswModuleEntry has returned. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.45: TDEventBswModuleTypeEnum

3.3.2.5.2 TDEventBswModeDeclaration

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

Class	**TDEventBswModeDeclaration**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventBsw			
Note	This is used to describe timing events related to the mode communication on BSW level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventBsw*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
entryMode Declaration	ModeDeclaration	0..1	ref	Optional parameter which refines the scope of the TDEventBswModeDeclaration. If the parameter is set, the event occurs only if the mode declaration group prototype instance shall enter into the referenced ModeDeclaration.
exitMode Declaration	ModeDeclaration	0..1	ref	Optional parameter which refines the scope of the TDEventBswModeDeclaration. If the parameter is set, the event occurs only if the mode declaration group prototype instance shall exit from the referenced ModeDeclaration.
mode Declaration	ModeDeclarationGroup Prototype	0..1	ref	The scope of this timing event.

▽

△

Class	TDEventBswModeDeclaration			
tdEventBsw Mode DeclarationType	TDEventBswMode DeclarationTypeEnum	0..1	attr	The specific type of this timing event.

Table 3.46: TDEventBswModeDeclaration

[constr_6862] Existence of `TDEventBswModeDeclaration.tdEventBswModeDeclarationType` ⌈For each `TDEventBswModeDeclaration`, the attribute `tdEventBswModeDeclarationType` shall exist **`at the time when the Bsw Timing Description is complete`**.⌋*()*

[constr_6863] Existence of `TDEventBswModeDeclaration.modeDeclaration` ⌈For each `TDEventBswModeDeclaration`, the reference to `ModeDeclarationGroupPrototype` in the role `modeDeclaration` shall exist **`at the time when the Bsw Timing Description is complete`**.⌋*()*

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

Table 3.47: TDEventBswModeDeclarationTypeEnum

3.3.2.6 TDEventComplex

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

Complex timing events can be used during the specification of:

- `VfbTiming` 3.1.1
- `SystemTiming` 3.1.3
- `SwcTiming` 3.1.2
- `BswModuleTiming` 3.1.4.1

- EcuTiming 3.1.5

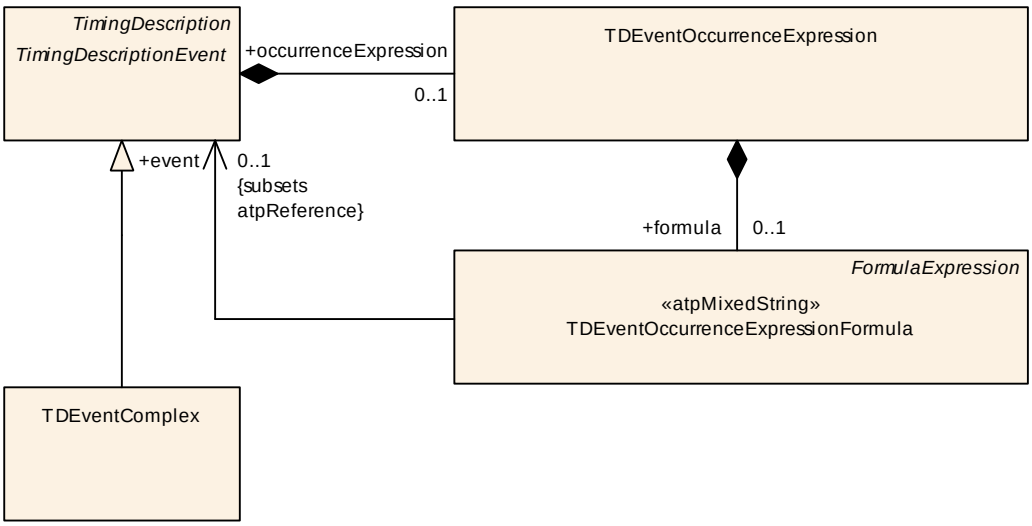


Figure 3.33: Complex timing event

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

Table 3.48: TDEventComplex

A complex timing event is a special observable event. In comparison to the ”atomic” events described above a complex event does not contain information about the context it references, like VariableDataPrototype in TDEventVariableDataPrototype. Instead, a complex event uses the occurrence expression to specify the context with regard to occurrences of TimingDescriptionEvents as describe in the following section.

3.3.2.7 TDEventSLLET

SL-LET timing events can be used during the specification of:

- VfbTiming 3.1.1
- SwcTiming 3.1.2
- SystemTiming 3.1.3

- EcuTiming 3.1.5

[TPS_TIMEX_00120]{DRAFT} **Purpose of TDEventSLLET** ⌈The meta-class TDEventSLLET provides an abstract class for modelling types of SL-LET timing events.⌋ *()*

3.3.2.7.1 TDEventSLLETPort

One sub-class of TDEventSLLET shall be used to specify timing events on SWC ports within the context of SL-LET.

[TPS_TIMEX_00124]{DRAFT} **Purpose of TDEventSLLETPort** ⌈The meta-class TDEventSLLETPort describes a SL-LET event originating from a referenced PortPrototype.⌋ *()*

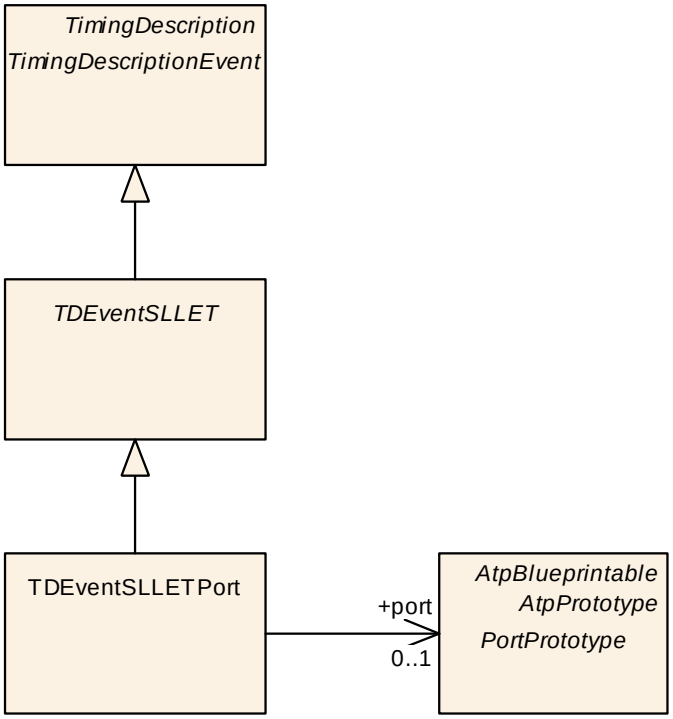


Figure 3.34: SL-LET events

Class	**TDEventSLLETPort**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventSLLET			
Note	Used to describe SL-LET timing events on the level of a SWC port. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventSLLET*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
port	PortPrototype	0..1	ref	The originating port of the timing event

Table 3.49: TDEventSLLETPort

3.3.2.8 Occurrence Expression Language for Timing Events

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

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

- Content Filter: filters occurrences of an atomic event based on the *value* of exchanged data or operation arguments.
- Complex Event: combines any number of atomic and complex events to specify a new TimingDescriptionEvent.

3.3.2.8.1 Specifying an Occurrence Expression

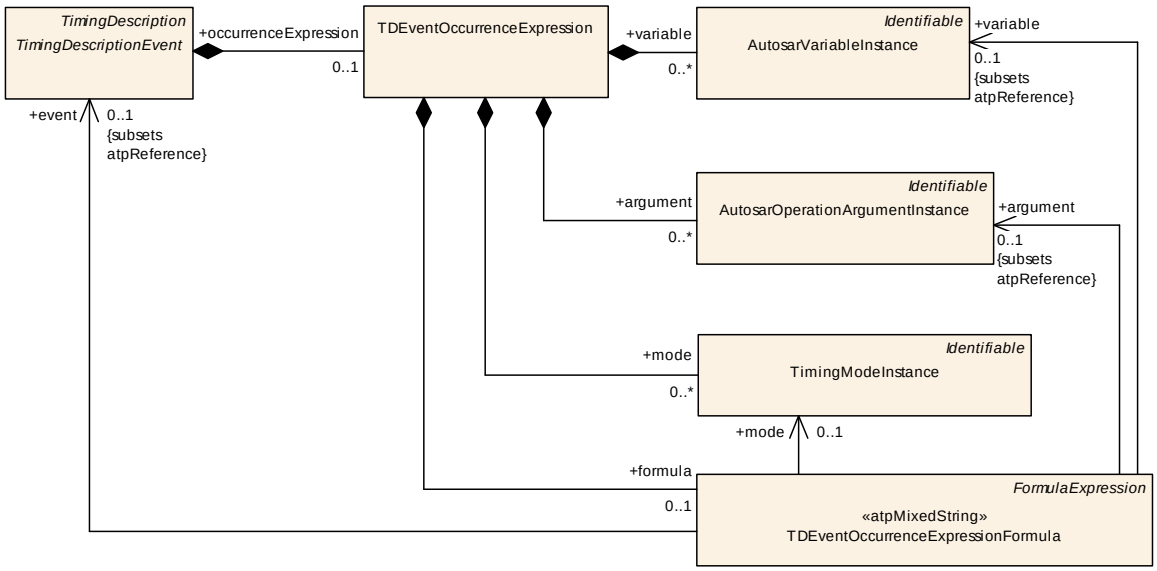


Figure 3.35: The occurrence expression

As shown in Figure 3.35, each TimingDescriptionEvent aggregates a TDEventOccurrenceExpression as an 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 [6]). The TDEventOccurrenceExpressionFormula uses the capabilities of the FormulaExpression and adds the following functions to the expression language:

- TIMEX_value:
  - operand:

    * a AutosarVariableInstance or
    * a AutosarOperationArgumentInstance
  - returns: Numerical (see [constr_4551]).

- TIMEX_occurs:
  - operand: a TimingDescriptionEvent whose occurrence shall be evaluated
  - returns: Boolean, TRUE if the referenced TimingDescriptionEvent occurs at the point in time the expression is evaluated, otherwise FALSE.
- TIMEX_hasOccurred:
  - operand: a TimingDescriptionEvent whose occurrence shall be evaluated
  - returns: Boolean, TRUE if the referenced TimingDescriptionEvent has occurred *at least once* before or at the same point in time the expression is evaluated, otherwise FALSE.
- TIMEX_timeSinceLastOccurrence:
  - operand: a TimingDescriptionEvent whose occurrence shall be evaluated
  - returns: Float and the unit is seconds. It returns the time difference between the point in time of the last occurrence of the referenced event and the point in time the expression is evaluated.
- TIMEX_angleSinceLastOccurrence:
  - operand: a TimingDescriptionEvent whose occurrence shall be evaluated
  - returns: Float and the unit is degree. It returns the angle of the crank shaft between the point in time of the last occurrence of the referenced event and the point in time the expression is evaluated.
- TIMEX_modeActive:
  - operand: a TimingModeInstance whose occurrence shall be evaluated
  - returns: Boolean, TRUE if the specified mode declaration is *active* at the point in time the expression is evaluated, otherwise FALSE.

The starting point of the time interval considered by the TIMEX functions is the point in time the measurement of the event occurrences has been started.

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 TimingModeInstance. Due to the

atpMixedString nature of the TDEventOccurrenceExpressionFormula several references can be used within the occurrence expression.

[constr_4500] Restricted usage of Occurence Expression functions ⌈The functions:

- *TIMEX_occurs*,
- *TIMEX_hasOccurred*,
- *TIMEX_timeSinceLastOccurrence*,
- *TIMEX_angleSinceLastOccurrence*,
- *TIMEX_modeActive*

shall only be used for an TDEventOccurrenceExpressionFormula applied to a TDEventComplex.⌋*()*

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

The TDEventOccurrenceExpressionFormula is specified such that it describes an *event* rather than a state. As a consequence the TDEventOccurrenceExpressionFormula shall ensure that a complex TimingDescriptionEvent *could* only occur at the occurrence time of one of the referenced TimingDescriptionEvents.

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

The example given below shows how to combine the functions introduced above in order to specify a TDEventOccurrenceExpressionFormula for a complex event called *ECX*. This complex event expresses that the two dataElements *DE1* and *DE2* are becoming available at the required port *RequiredPort* of *Swc1* within an interval of maximum 0.5 ms *and* that the value of the dataElement *DE3* is larger than three (3) at this point in time.

Figure 3.36 sketches the AUTOSAR software component model of this example.

An SWC *Swc1* has a RPortPrototype called *RequiredPort*, and a PPortPrototype called *ProvidedPort*. Both ports are typed by a SenderReceiverInterface called *SenderReceiverInterface1*, and consist of three dataElements: *DE1*, *DE2* and *DE3* [1].

[1] one could also instead define three discrete RPortPrototypes typed by a SenderReceiverInterface, each with a single dataElement

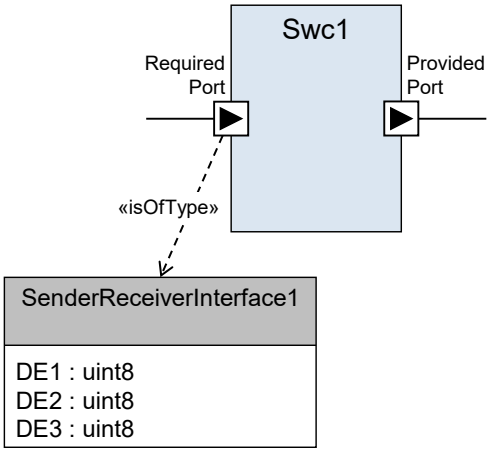


Figure 3.36: The SWC used by the Occurrence Expression Example

The corresponding timing model for the given example is shown in listing C.14. Since the timing is described for a software component in the Virtual Functional Bus view, the `VfbTiming` is used for specifying the corresponding timing model, namely the Virtual Functional Bus Timing View. This timing model shall only contain timing description events related to the Virtual Functional Bus as described in section 3.3.2.1.

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

Condition1 Either atomic timing event *E1* or *E2* shall occur. In this example, *E1* and *E2* are atomic timing events `TDEventVariableDataPrototype` which occur when the `VariableDataPrototype`s called *DE1* and *DE2* are received on `PortPrototype` called *Required Port* of the component called *Swc1*.

Condition2 The value of the `VariableDataPrototype` called *DE3* shall be greater than 3.

Condition3 The `VariableDataPrototype`s called *DE1* and *DE2* shall become available at the *required* `PortPrototype` called *RequiredPort* within a time interval of maximum 0.5 milliseconds.

The complex event *ECX* would be described by the following occurrence expression in C.5.1

Due to the first condition the complex event *ECX* can only occur when one of the atomic timing events *E1* or *E2* occurs at the point in time of evaluation and thus satisfies the principle that the occurence expresses the semantics of an event and not a state.

Figure 3.39 shows a measurement of the event occurrences.

The corresponding AUTOSAR ARXML file fragment for the complex event *ECX* has the following appearance:

Class	TDEventOccurrenceExpression			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventOccurrenceExpression			
Note	This is used to specify a filter on the occurrences of TimingDescriptionEvents by means of a 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*			
Aggregated by	*TimingDescriptionEvent*.occurrenceExpression			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
argument	AutosarOperation ArgumentInstance	*	aggr	An occurrence expression can reference an arbitrary number of OperationArgumentPrototypes in its expression. This association aggregates instance references to OperationArgumentPrototypes which can be referenced in the expression.
formula	TDEventOccurrence ExpressionFormula	0..1	aggr	This is the expression formula which is used to describe the occurrence expression.
mode	TimingModeInstance	*	aggr	An occurrence expression can reference an arbitrary number of TimingModeInstances in its expression. This association aggregates instance references to Mode Declaration which can be referenced in the expression.
variable	AutosarVariable Instance	*	aggr	An occurrence expression can reference an arbitrary number of VariableDataPrototypes in its expression. This association aggregates instance references to Variable DataPrototypes which can be referenced in the expression.

Table 3.50: TDEventOccurrenceExpression

[constr_6879] Existence of TDEventOccurrenceExpression.formula ⌈For each TDEventOccurrenceExpression, the attribute formula shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent.**⌋*()*

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

Table 3.51: TDEventOccurrenceExpressionFormula

Class	AutosarVariableInstance			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventOccurrenceExpression			
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*			
Aggregated by	TDEventOccurrenceExpression.variable, TimingExtensionResource.timingVariable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
variableInstance	DataPrototype	0..1	iref	This is the reference to the instanceRef definition. **InstanceRef implemented by:** VariableInComponent InstanceRef

Table 3.52: AutosarVariableInstance

[constr_6880] Existence of `AutosarVariableInstance.variableInstance` ⌈For each `AutosarVariableInstance`, the reference in the role `variableInstance` shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋ *()*

Class	AutosarOperationArgumentInstance			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventOccurrenceExpression			
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 ClientServer Interface			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	TDEventOccurrenceExpression.argument, TimingExtensionResource.timingArgument			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
operation Argument Instance	DataPrototype	0..1	iref	This is the reference to the instanceRef definition. **InstanceRef implemented by:** OperationArgumentIn ComponentInstanceRef

Table 3.53: AutosarOperationArgumentInstance

[constr_6881] Existence of `AutosarOperationArgumentInstance.operationArgumentInstance` ⌈For each `AutosarOperationArgumentInstance`, the reference in the role `operationArgumentInstance` shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋ *()*

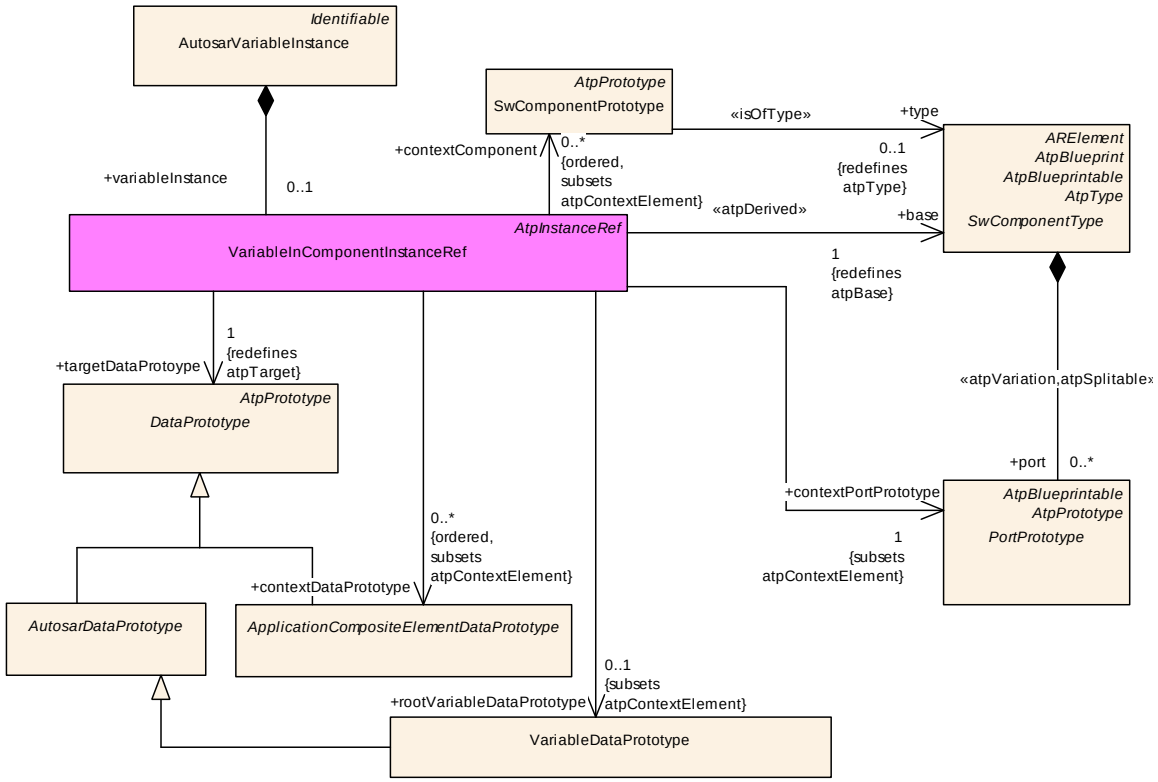


Figure 3.37: The required context information to reference a variable instance within AUTOSAR.

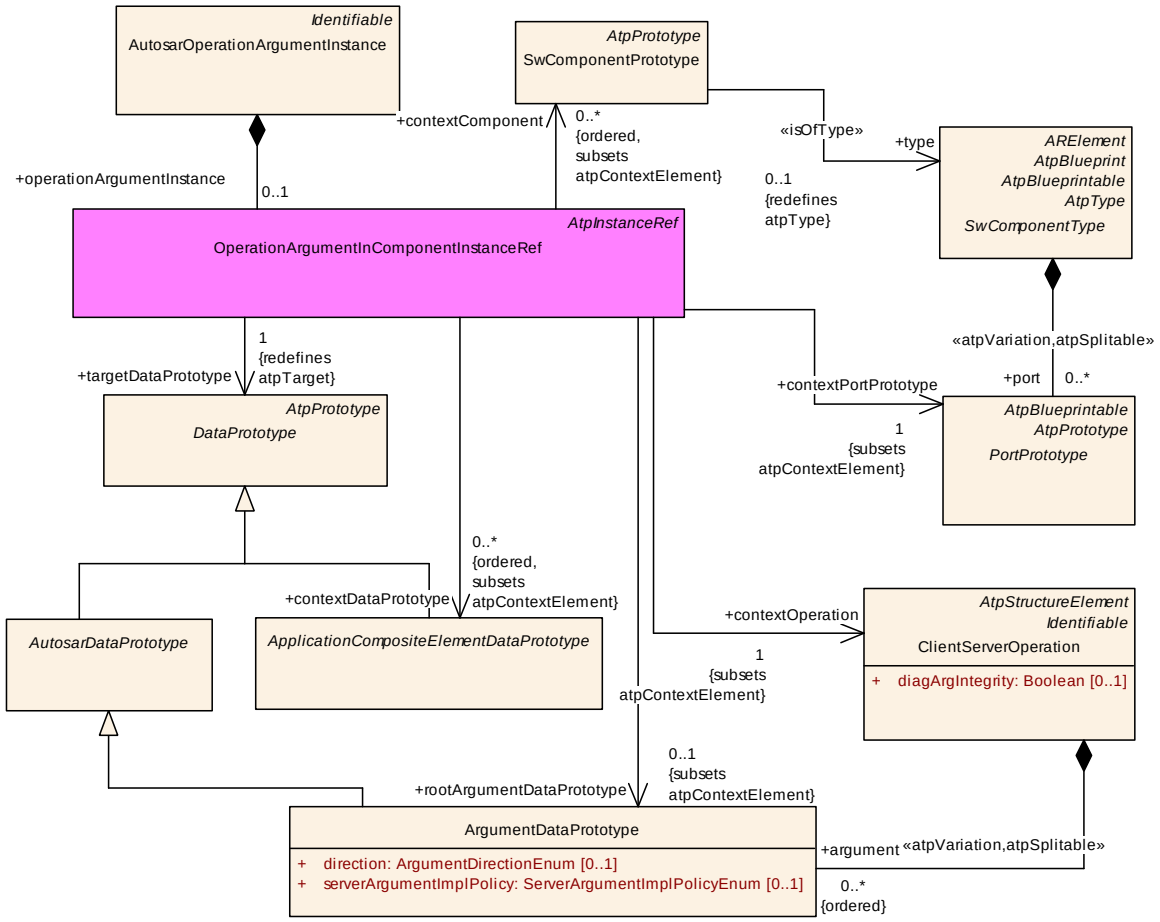


Figure 3.38: The required context information to reference an operation argument instance within AUTOSAR.

3.3.2.9 Occurrence Expression Language Syntax

The occurrence expression language is based on the syntax of the formula language defined in the Generic Structure Template [6]. It extends the language by additional functions and additional references to model elements. In the following, the implications of the extensions to the syntax are presented based on the grammar definition.

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

3.3.2.9.1 Interpreting an Occurrence Expression

Based on the specification mechanism described in the previous sections it is possible to use the occurrence expression formula to refine the timing specification to the intended precision. This section describes how such an occurrence expression has to be interpreted. The duty of the interpreter is to determine the occurrences of the `Tim-`

ingDescriptionEvent for which the occurrence expression is defined. This is done in two ways, depending on whether the occurrence expression is used as a content filter or as a complex event.

3.3.2.9.1.1 Interpreting a Content Filter

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

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

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

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

[constr_4503] Restricted usage of AutosarOperationArgumentInstance for Content Filter ⌈If a content filter is defined for an atomic event then references to AutosarOperationArgumentInstances are only allowed if the atomic event is of type TDEventOperation. Only if such an atomic event occurs, the value of the operation arguments can be evaluated. Thus, also the scope of the atomic event shall be the same as the AutosarOperationArgumentInstance, meaning that they shall point to the same ClientServerOperation. Finally, references to an AutosarOperationArgumentInstance with argument direction ”out” are only allowed, if the atomic event of type TDEventOperation refers either to the point in time when the operation call response has been sent (TD-EVENT-OPERATION-TYPE=OPERATION-CALL-RESPONSE-SENT) or to the point in time when the operation call response has been received (TD-EVENT-OPERATION-TYPE=OPERATION-CALL-RESPONSE-RECEIVED).⌋*()*

3.3.2.9.1.2 Interpreting a Complex Event

In this case, the occurrence expression is defined for a complex event. All features of the occurrence expression language can be used for this expression type. At a specific

point in time *t*, the interpreter evaluates the expression to determine if the complex event has occurred.

Considering the occurrence expression defined for the example given in Section 3.3.2.8.1, the interpreter ”implements” a function *ECX(t)* which returns TRUE, if the complex event ECX occurs at time *t*:

```
ECX(t) =
( TIMEX_occurs( t, /Example/Expression/E1 )
   || TIMEX_occurs( t, /Example/Expression/E2 ) )
&& TIMEX_value( t, /Example/Expression/ECX/avi_DE3 ) > 3
&& abs( TIMEX_timeSinceLastOccurrence( t, /Example/Expression/E1 ) -
   TIMEX_timeSinceLastOccurrence( t, /Example/Expression/E2 ) ) <= 0.0005
```

The expression shall only be evaluated at occurrence times of *E1* or *E2*, because only then the complex event *ECX* can occur and the expression can return TRUE.

As shown in the sketched trace in Figure 3.39 the timing description events called *E1* and *E2* occur at different times. On the left hand side of this figure the two events occur within a time interval of 0.0005 seconds. The point in time the given occurrence expression is evaluated is the point in time the event *E2* occurs. The result of the occurrence expression at this point in time, $t_{evaluate}$ respectively t_{E2}, is TRUE. On the right hand side of this figure the two events do not occur within a time interval of 0.0005 seconds. The point in time the given occurrence expression is evaluated is the point in time the event *E1* occurs. The result of the occurrence expression at this point in time, $t_{evaluate}$ respectively t_{E1}, is FALSE.

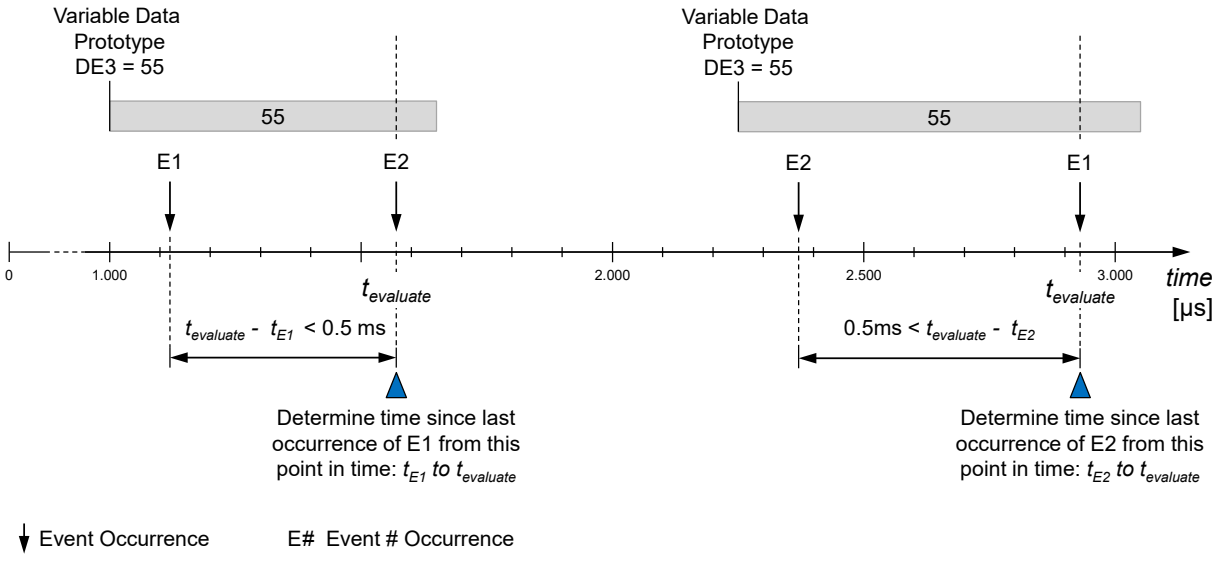


Figure 3.39: Trace showing various occurrences of the timing description events *E1* and *E2*, as well as the value of the variable *DE3*.

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

- the value of a referenced `AutosarOperationArgumentInstance` at time *t*.

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

There are different ways to gather the required information:

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

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

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

3.3.2.10 Time Base Referencing for Timing Description Events

In certain use-cases, a TimingDescriptionEvent may reference an optional TimingDescriptionEvent.clockReference to provide a relation to an *actual* time base, this concept is explained in section 4.2.

3.4 TimingConstraint

TIMEX TimingConstraints may be used as functional requirements and thus can be used as a basis for testing. Or, could be used as timing guarantees, for example, in the context of a performance specification.

The application of TimingConstraints are contextual and are applied to constrain TimingDescriptionEvents, TimingDescriptionEventChains or ExecutableEntitys.

3.4.1 TimingConstraints on TimingDescriptionEventChains

In 3.40 differing sub-classes of TimingConstraints on TimingDescriptionEventChains are shown. Generally a TimingConstraint, when applied to TimingDescriptionEventChain, refers to a given scope/chain context with a temporal restriction, present in most sub-classes. The remaining attributes are context specific. As the TimingDescriptionEventChain has a semantic of a directed acyclic graph, the direction is obvious, but it matters whether a single TimingDescriptionEventChain or a group of TimingDescriptionEventChain are constrained.

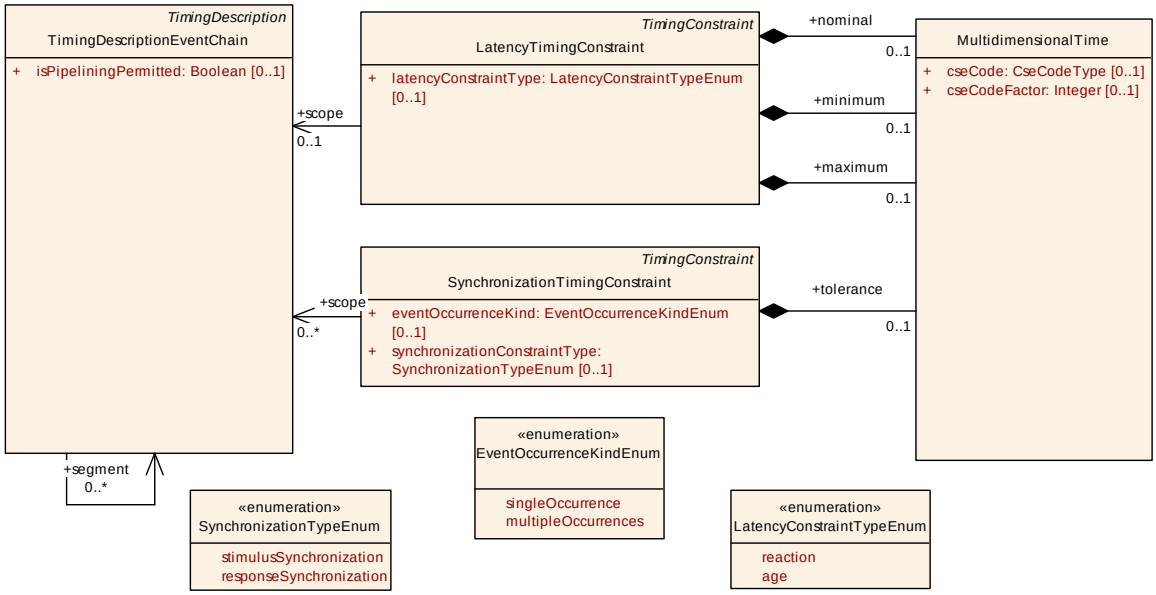


Figure 3.40: TimingConstraints on TimingDescriptionEventChains

3.4.1.1 SynchronizationTimingConstraint

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

necessity for synchronization needs to be expressed at design level. For this purpose, synchronization constraints are used.

A SynchronizationTimingConstraint imposes a synchronization restriction among either the stimulus or response event occurrences of two or more TimingDescriptionEventChains. In the former case (stimulus synchronization) the referenced TimingDescriptionEventChains shall have the same response event (join), or in the latter case (response synchronization) they shall have the same stimulus event (fork).

Class	**SynchronizationTimingConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::SynchronizationTiming Constraint			
Note	This constraint is used to restrict the timing behavior of different, but correlated events or event chains, with regard to synchronization. Two scenarios are supported: • If (synchronizationConstraintType==responseSynchronization) – TimingDescriptionEvents: An arbitrary number of correlated events which play the role of responses shall occur synchronously with respect to a predefined tolerance. – TimingDescriptionEventChains: 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. • If (synchronizationConstraintType==stimulusSynchronization) – TimingDescriptionEvents:An arbitrary number of correlated events which play the role of stimuli shall occur synchronously with respect to a predefined tolerance. – TimingDescriptionEventChains: An arbitrary number of correlated event chains with a common response, but different stimuli, where the stimuli shall occur synchronously with respect to a predefined tolerance. In case the constraint is imposed on events the following two scenarios are supported: • If (eventOccurrenceKind==singleOccurrence): any of the events shall occur only once in the given time interval. • If (eventOccurrenceKind==multipleOccurrences): any of the events may occur more than once in the given time interval. In other words multiple occurrences of an event within the given time interval are permitted.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventOccurrenceKind	EventOccurrenceKindEnum	0..1	attr	Indicates whether the referenced events shall occur only once (single occurrence) or multiple times (multiple occurrences) in the given time interval.
scope	TimingDescriptionEventChain	*	ref	The event chains that are in the scope of the constraint. Mutually exclusive to scopeEvent, see ([constr_4522]).
scopeEvent	TimingDescriptionEvent	*	ref	The events that are in the scope of the constraint. Mutually exclusive to scope, see ([constr_4522])
synchronizationConstraintType	SynchronizationTypeEnum	0..1	attr	Indicates whether the associated events of the SynchronizationTimingConstraint have a common stimulus or response.
tolerance	MultidimensionalTime	0..1	aggr	The maximum time interval, within which the synchronized events shall occur. The events may occur in any order within this time interval. The time interval starts at the point-in-time when one of the referenced events occurs.

Table 3.54: SynchronizationTimingConstraint

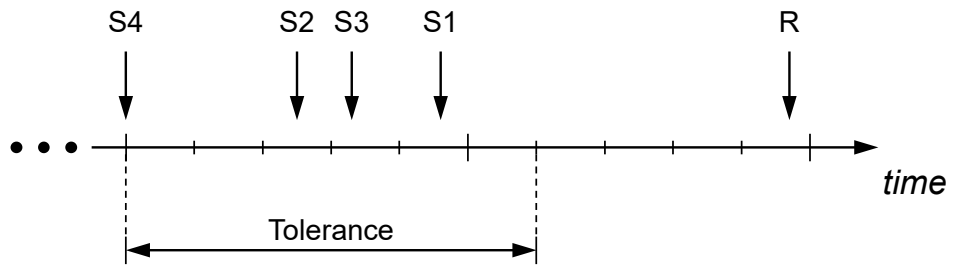
Enumeration	**SynchronizationTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::SynchronizationTiming Constraint
Note	Specifies the `synchronizationConstraintType` for a `SynchronizationTimingConstraint`.
Aggregated by	SynchronizationTimingConstraint.synchronizationConstraintType
Literal	***Description***
response Synchronization	In case that the Synchronization Timing Constraint is specified for event chains, the response events of the associated event chains shall occur synchronously with respect to the specified tolerance. All associated event chains shall have the same stimulus event. In case that the Synchronization Timing Constraint is specified for events, the associated events shall occur synchronously with respect to the specified tolerance. All associated events represent the response events of a common stimulus event, even such a stimulus event is not known yet or not available in the scope of the model. **Tags:** atp.EnumerationLiteralIndex=0
stimulus Synchronization	In case that the Synchronization Timing Constraint is specified for event chains, the stimulus events of the associated event chains shall occur synchronously with respect to the specified tolerance. All associated event chains shall have the same response event. In case that the Synchronization Timing Constraint is specified for events, the associated events shall occur synchronously with respect to the specified tolerance. All associated events represent the stimulus events of a common response event, even such a response event is not known yet or not available in the scope of the model. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.55: SynchronizationTypeEnum

Enumeration	**EventOccurrenceKindEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::SynchronizationTiming Constraint
Note	Specifies the `eventOccurrenceKind` for a `SynchronizationTimingConstraint`.
Aggregated by	SynchronizationTimingConstraint.eventOccurrenceKind
Literal	***Description***
multiple Occurrences	Specifies that an event may occur more than once in a given time interval. **Tags:** atp.EnumerationLiteralIndex=0
singleOccurrence	The referenced event shall occur only once in a given time interval. Indicates whether the referenced events shall occur only once (single occurrence) or multiple times (multiple occurrences) in the given time interval. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.56: EventOccurrenceKindEnum

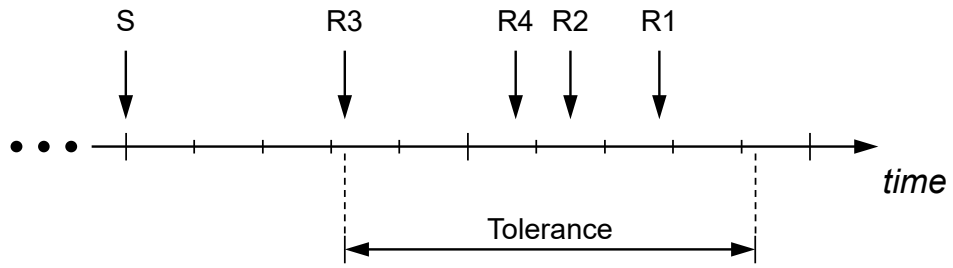
The parameters are shown in use in Figure 3.41 and Figure 3.42.



S# Stimulus Event
R Response Event

Event Occurrence

Figure 3.41: `SynchronizationTimingConstraint` imposed on `TimingDescriptionEventChain.stimulus`



S Stimulus Event
R# Response Event

Event Occurrence

Figure 3.42: `SynchronizationTimingConstraint` imposed on `TimingDescriptionEventChain.response`

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

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

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

[constr_4514] `SynchronizationTimingConstraint` shall reference at least two event chains ⌈In the case, that the `SynchronizationTimingConstraint` is

imposed on TimingDescriptionEventChains then at least two (2) TimingDescriptionEventChains shall be referenced.⌋()

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

3.4.1.2 LatencyTimingConstraint

A LatencyTimingConstraint specifies a minimum and/or maximum time duration between the occurrence of the stimulus and the occurrence of the corresponding response of that TimingDescriptionEventChain. For example, this could 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 function/task to be executed on a processor.

[TPS_TIMEX_00004] LatencyTimingConstraint specifies latency constraints ⌈The element LatencyTimingConstraint[2] is used to specify the amount of time that elapses between the occurrence of any two TimingDescriptionEvents.⌋(*RS_TIMEX_00001*, *RS_TIMEX_00002*, *RS_TIMEX_00012*, *RS_TIMEX_00015*)

Class	**LatencyTimingConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::LatencyTimingConstraint			
Note	Constrains the time duration between the occurrence of the stimulus and the occurrence of the corresponding response of that scope. In contrast to scope, a causal dependency between the stimulus and the corresponding response of the scope is required.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
latency ConstraintType	LatencyConstraintType Enum	0..1	attr	The specific type of this latency constraint.
maximum	MultidimensionalTime	0..1	aggr	The maximum latency between the occurrence of the stimulus and the occurrence of the corresponding response of the associated event chain. **Tags:** xml.sequenceOffset=20
minimum	MultidimensionalTime	0..1	aggr	The minimum latency between the occurrence of the stimulus and the occurrence of the corresponding response of the associated event chain. **Tags:** xml.sequenceOffset=10
nominal	MultidimensionalTime	0..1	aggr	The nominal latency between the occurrence of the stimulus and the occurrence of the corresponding response of the associated event chain. **Tags:** xml.sequenceOffset=30
scope	TimingDescriptionEvent Chain	0..1	ref	The event chain that defines the scope of the constraint.

Table 3.57: LatencyTimingConstraint

[2] A synonym for delay

Enumeration	**LatencyConstraintTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::LatencyTimingConstraint
Note	Specifies the `latencyConstraintType` for a `LatencyTimingConstraint`.
Aggregated by	LatencyTimingConstraint.latencyConstraintType
Literal	***Description***
age	The `LatencyTimingConstraint` is seen from the perspective of the `response` event of the `scope`. Given a certain `response` event, the age interval of the latest `stimulus` is constrained. **Tags:** atp.EnumerationLiteralIndex=0
reaction	The `LatencyTimingConstraint` is seen from the perspective of the `stimulus` event of the `scope`. Given a certain `stimulus` event, the reaction interval of the first `response` is constrained. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.58: 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 `TimingDescriptionEventChain` in the `scope`.

`Example`: 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 (under-sampling). Accordingly, data duplication occurs, if the consumer's period is smaller than the producer's period (oversampling). This is depicted in figure 3.43.

The application of `LatencyTimingConstraint`s leads to some interesting observations:

- In systems without over- and under-sampling, *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.

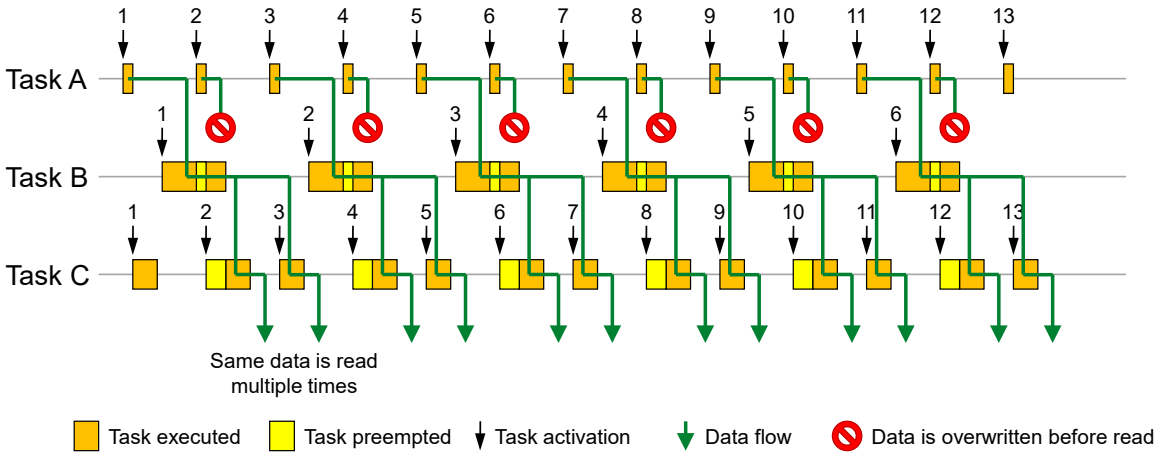


Figure 3.43: Loss and duplication of data due to under- and oversampling.

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

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

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

[constr_6837] Existence of `LatencyTimingConstraint.latencyConstraintType` ⌈For each `LatencyTimingConstraint`, the attribute `latencyConstraintType` shall exist **`at the imposition time associated with the concrete subclass of TimingDescriptionEvent`**.⌋*()*

[constr_6838] Existence of `LatencyTimingConstraint.maximum` ⌈For each `LatencyTimingConstraint`, the attribute `maximum` shall exist **`at the imposition time associated with the concrete subclass of TimingDescriptionEvent`**.⌋*()*

[constr_6839] Existence of `LatencyTimingConstraint.minimum` ⌈For each `LatencyTimingConstraint`, the attribute `minimum` shall exist **`at the imposition time associated with the concrete subclass of TimingDescriptionEvent`**.⌋*()*

[constr_6841] Existence of `LatencyTimingConstraint.scope` ⌈For each `LatencyTimingConstraint`, the reference in the role `scope` shall exist at least once **`at the imposition time associated with the concrete subclass of TimingDescriptionEvent`**.⌋*()*

3.4.2 TimingConstraints on TimingDescriptionEvents

In 3.44 differing sub-classes of `TimingConstraint`s on `TimingDescriptionEvent`s are shown. Generally a `TimingConstraint`, when applied to `TimingDescriptionEvent`, refers to a given scope/context with a temporal restriction, present in most sub-classes. The remaining attributes are context specific.

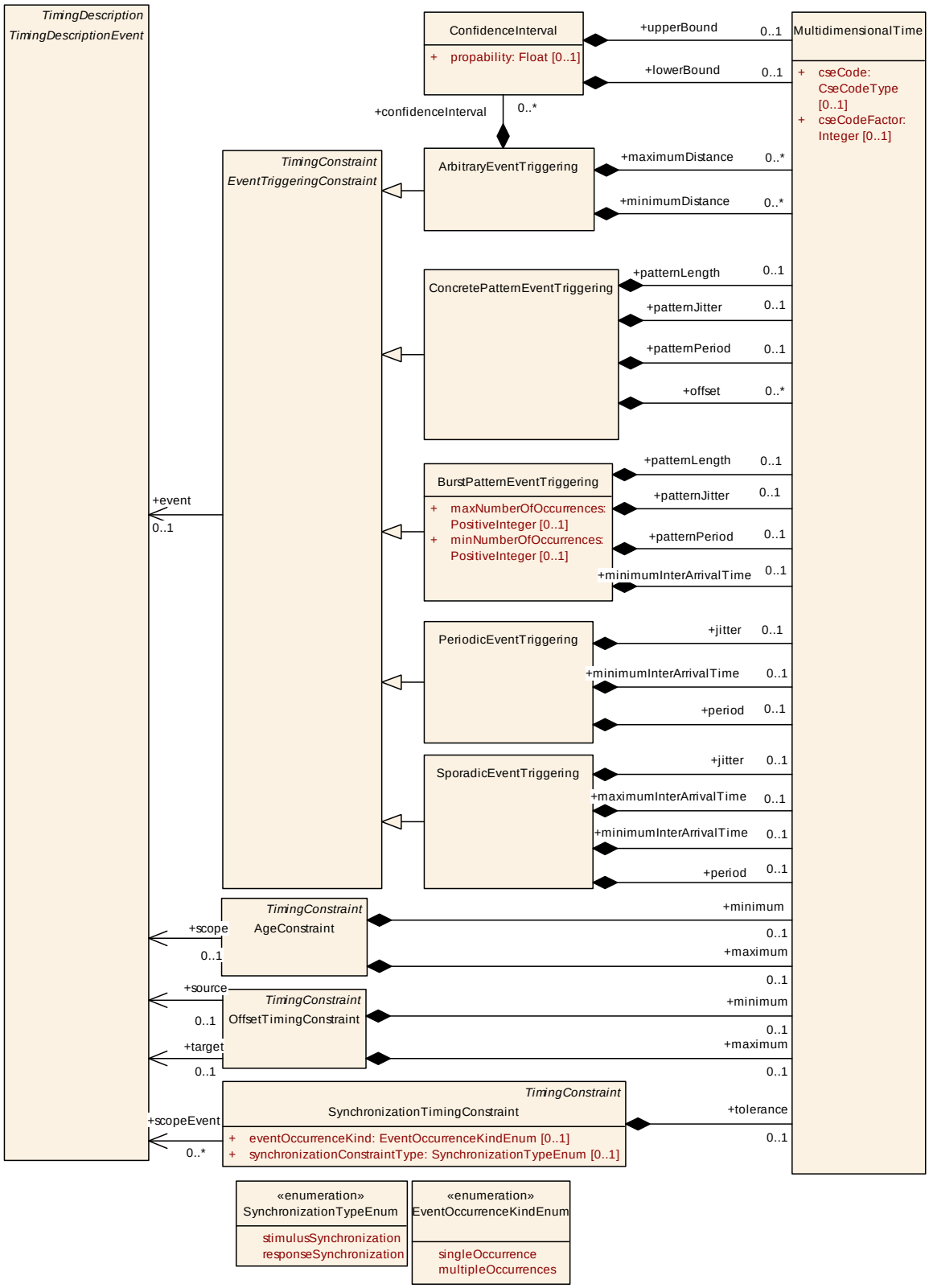


Figure 3.44: TimingConstraints on TimingDescriptionEvents

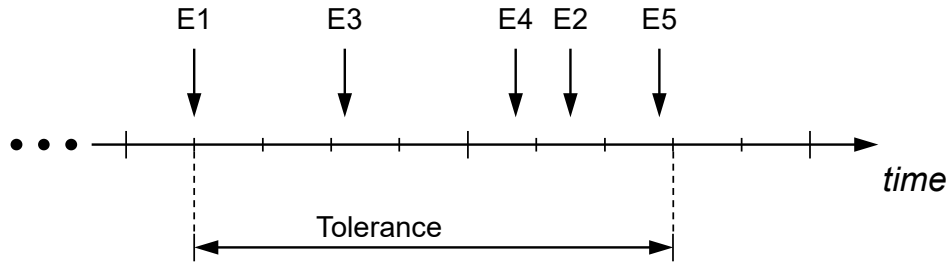
3.4.2.1 SynchronizationTimingConstraint

In some cases the complete event chains are not entirely known, or not available in the scope of the model, at the point in time the timing constraint shall be specified. For this purpose, TIMEX allows the specification of synchronization constraints on events. In this case, the events referenced by the constraint are related implicitly, because they have a common stimulus (in case of constraint type `responseSynchronization` or a common response (in case of constraint type `stimulusSynchronization` not known yet, or not available in the scope of the model.

At a later stage during the development, when the refined software architecture exposes the complete event chains (e.g. because the common stimulus gets known), the respective event chains shall be specified and associated with a `SynchronizationTimingConstraint` on event chains (see 3.4.1.1) in order to refine the previously defined `SynchronizationTimingConstraint` on events.

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

The parameters are shown in use in Figure 3.45.



E# Event #

Event Occurrence

Figure 3.45: Parameter characterizing the Synchronization Constraint

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

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

[constr_6846] Existence of `SynchronizationTimingConstraint.synchronizationConstraintType` ⌈For each `SynchronizationTimingConstraint`,

the attribute synchronizationConstraintType shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋()

[constr_6847] Existence of SynchronizationTimingConstraint.tolerance ⌈For each SynchronizationTimingConstraint, the attribute tolerance shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋()

3.4.2.2 EventTriggeringConstraint

The EventTriggeringConstraint is an abstract base class representing a number sub-classes of concrete types (patterns) of event reporting. Regardless of the concrete meta-class, the EventTriggeringConstraint needs an event as it's scope.

[TPS_TIMEX_00003] EventTriggeringConstraint specifies occurrence behavior respectively model ⌈The element EventTriggeringConstraint is an abstract meta-class to specify the particular occurrences of a given TimingDescriptionEvent.⌋(*RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008*)

Class	***EventTriggeringConstraint*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	Describes the occurrence behavior of the referenced timing event. The occurrence behavior can only be determined when a mapping from the timing events to the implementation can be obtained. However, such an occurrence behavior can also be described by the modeler as an assumption or as a requirement about the occurrence of the event.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Subclasses	ArbitraryEventTriggering, BurstPatternEventTriggering, ConcretePatternEventTriggering, PeriodicEventTriggering, SporadicEventTriggering			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event	TimingDescriptionEvent	0..1	ref	The referenced timing event

Table 3.59: EventTriggeringConstraint

[constr_6818] Existence of EventTriggeringConstraint.event ⌈For each EventTriggeringConstraint, the reference in the role event shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋()

3.4.2.2.1 PeriodicEventTriggering

[TPS_TIMEX_00010] PeriodicEventTriggering specifies periodic occurrences of events ⌈The element PeriodicEventTriggering is used to specify

the characteristics of a TimingDescriptionEvent which occurs periodically.⌋ *(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008, RS_TIMEX_00015)*

Class	**PeriodicEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	Describes the behavior of an event with a strict periodic occurrence pattern, given by period. 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*			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
jitter	MultidimensionalTime	0..1	aggr	The maximum deviation of the periodic event occurrence. **Tags:** xml.sequenceOffset=20
minimumInterArrivalTime	MultidimensionalTime	0..1	aggr	The minimum time distance between subsequent consecutive occurrences of the associated event. If the minimumInterArrivalTime is less than the period minus the jitter, then the minimumInterArrivalTime has no effect on the properties of the constraint. **Tags:** xml.sequenceOffset=10
period	MultidimensionalTime	0..1	aggr	The periodic distance between subsequent occurrences of the event. **Tags:** xml.sequenceOffset=30

Table 3.60: PeriodicEventTriggering

Let t_n be the point-in-time of the n-th occurrence of the event. A PeriodicEventTriggering Constraint is satisfied if, and only if, at least one reference point-in-time $t_{reference}$ exists such that for every occurrence of the event at t_n the following holds true:

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

and for all of those event occurrences the minimum distance shall be less than or equal to minimumInterArrivalTime.

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

Figure 3.46 illustrates the parameters of the PeriodicEventTriggering. The upper part of this figure shows the case that the value of jitter is less than the value of the parameter period; whereas the lower part of this figure shows the jitter $\geq$ period.

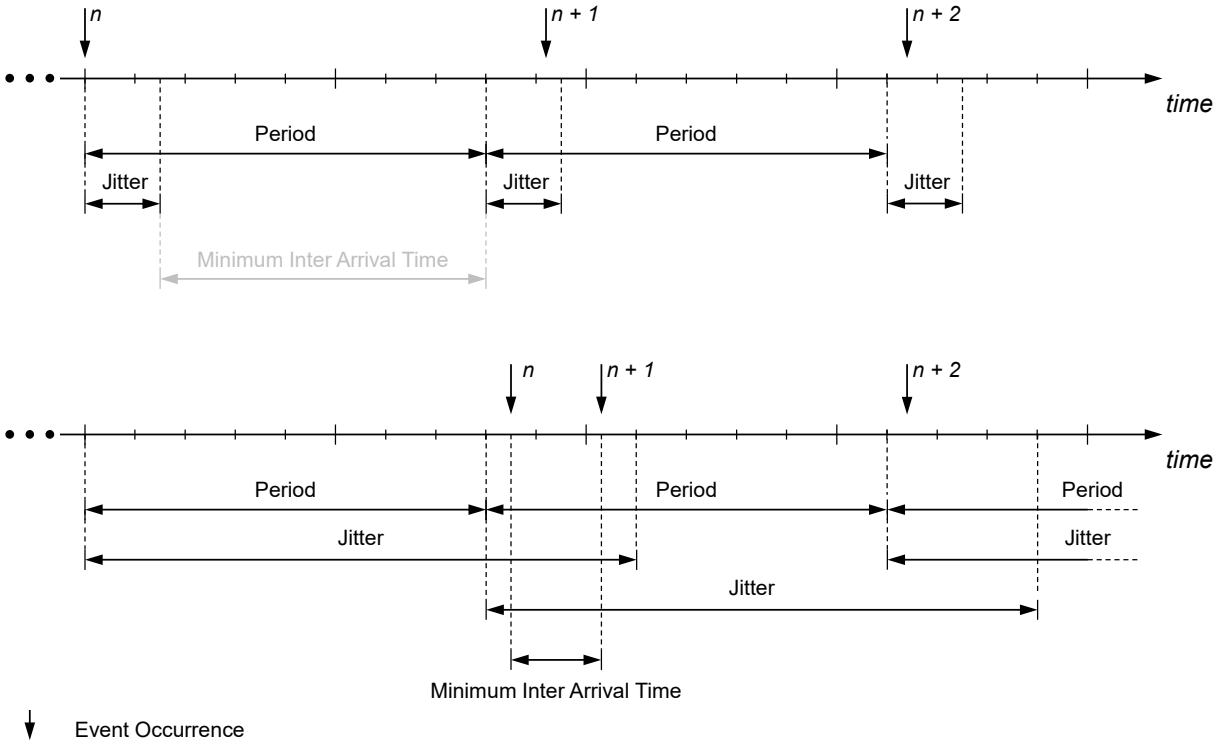


Figure 3.46: Parameters characterizing the `PeriodicEventTriggering`

`Example`: A `PeriodicEventTriggering` Constraint is specified with the following parameters: `period`=`6ms` and `jitter`=`2ms`. In other words, one imposes a timing constraint on an event to occur every `6ms` and specifies that a deviation of `2ms` is tolerable.

In addition, it is assumed that the `minimumInterArrivalTime`=`1ms` and therefore has no impact on the timing of the event's occurrences. This timing constraint is shown in Figure 3.47. The repeating gray-colored rectangles in this figure indicate the time intervals during which the event may occur; i.e. they demarcate the subsequent time intervals the event is expected to occur.

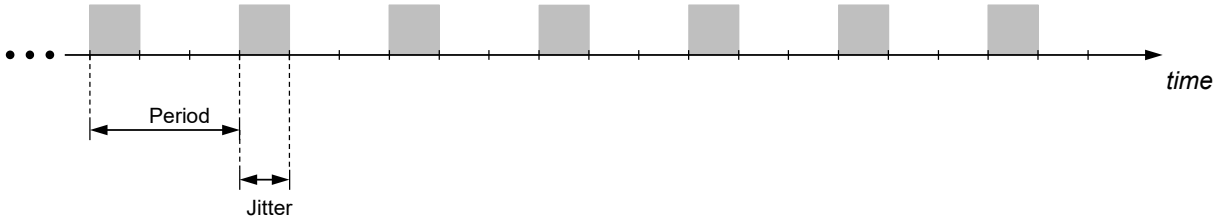


Figure 3.47: Example of a `PeriodicEventTriggering` Constraint

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

The subsequent event occurrences shown in Figure 3.48 satisfy the given `PeriodicEventTriggering` constraint, because all occurrences of the event observed

during the observation time interval happen in their corresponding time interval given by `period` and `jitter`.

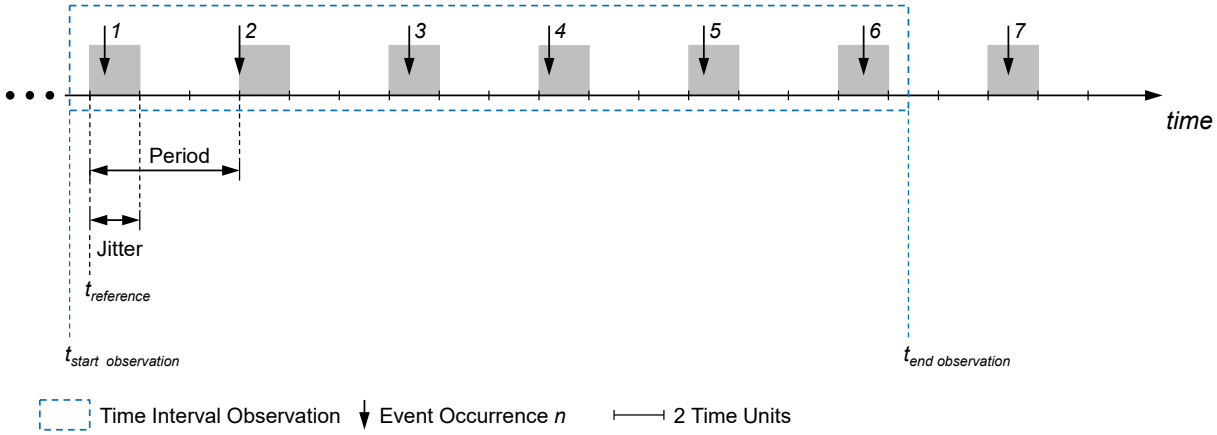


Figure 3.48: Event occurrences satisfying the given `PeriodicEventTriggering` Constraint shown in the `example`

The subsequent event occurrences shown in Figure 3.49 satisfy the given periodic event triggering constraint, because all occurrences of the event observed during the observation time interval happen in their corresponding time interval given by `period` and `jitter`. In contrast to the example shown in Figure 3.48 the reference point-in-time is another one.

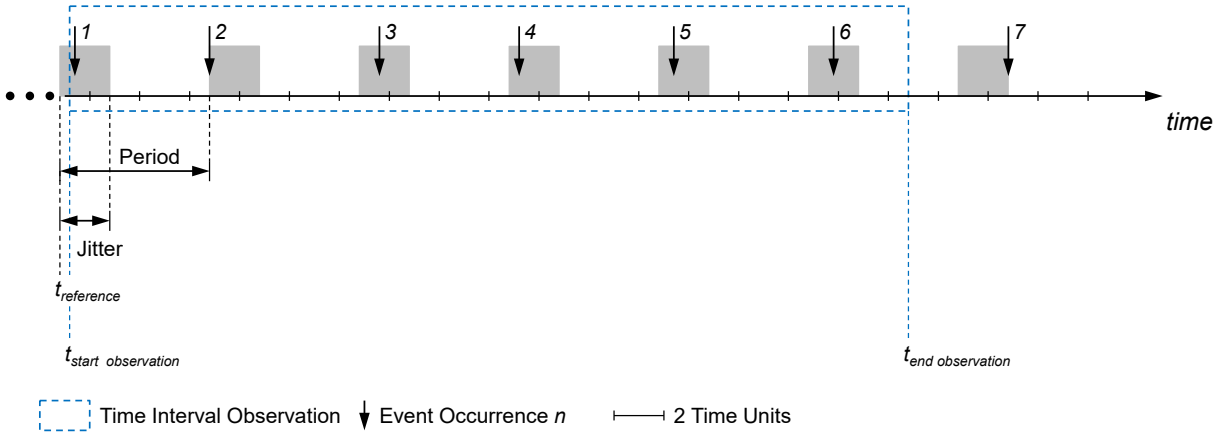


Figure 3.49: Event occurrences satisfying the given `PeriodicEventTriggering` Constraint shown in the `example`, but with another reference point-in-time $t_{reference}$.

The subsequent event occurrences shown in Figure 3.50 violate the given `PeriodicEventTriggering` constraint, because the fifth occurrence of the event does not happen in its corresponding time interval given by `period` and `jitter`. In other words, there does not exist a reference point-in-time that ensures that all occurrences of the event observed during the observation time interval happen in their corresponding time interval given by `period` and `jitter`. This results in a violation of the parameters `period` and `jitter`.

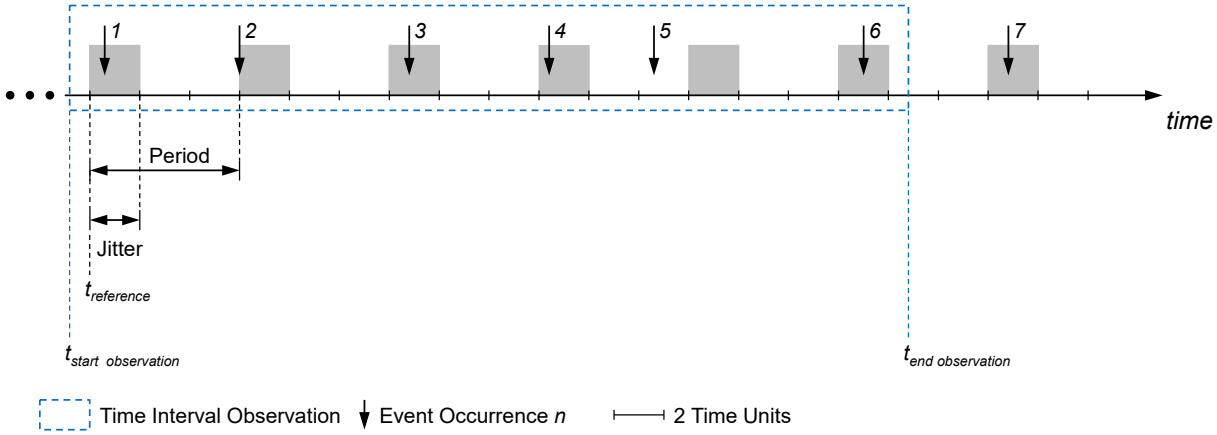


Figure 3.50: Event occurrences violating the given `PeriodicEventTriggering` Constraint shown in the example at the beginning of this subsection.

The subsequent event occurrences shown in Figure 3.51 violate the given periodic event triggering constraint, because the fourth occurrence of the event does not happen in its corresponding time interval given by `period` and `jitter`. In other words, the fourth occurrence of the event happens in the time interval the fifth occurrence of the event happens and therefore violates the specified `jitter`.

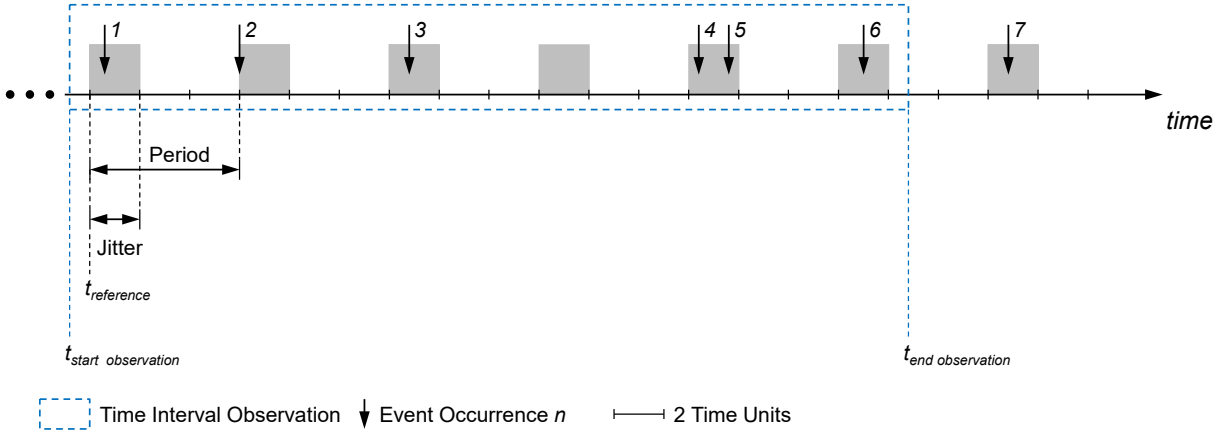


Figure 3.51: Event occurrences satisfying the given `PeriodicEventTriggering` Constraint shown in the `example`

[constr_4543] Maximum value of `minimumInterArrivalTime` ⌈The `minimumInterArrivalTime` shall be ≤ the `period`.⌋*()*

[constr_6819] Existence of `PeriodicEventTriggering.jitter` ⌈For each `PeriodicEventTriggering`, the attribute `jitter` shall exist **`at the imposition time associated with the concrete subclass of TimingDescriptionEvent`**.⌋*()*

[constr_6820] Existence of `PeriodicEventTriggering.minimumInterArrivalTime` ⌈For each `PeriodicEventTriggering`, the attribute `minimumInterArrivalTime` shall exist **`at the imposition time associated with the concrete subclass of TimingDescriptionEvent`**.⌋*()*

[constr_6821] Existence of `PeriodicEventTriggering.period` ⌈For each `PeriodicEventTriggering`, the attribute `period` shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent.**⌋*()*

3.4.2.2.2 SporadicEventTriggering

In contrast to a `PeriodicEventTriggering`, a `SporadicEventTriggering` permits events which may not necessarily occur. It adds one additional parameter: the `maximumInterArrivalTime` to specify the largest possible time distance between two `TimingDescriptionEvent` occurrences.

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

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

Table 3.61: SporadicEventTriggering

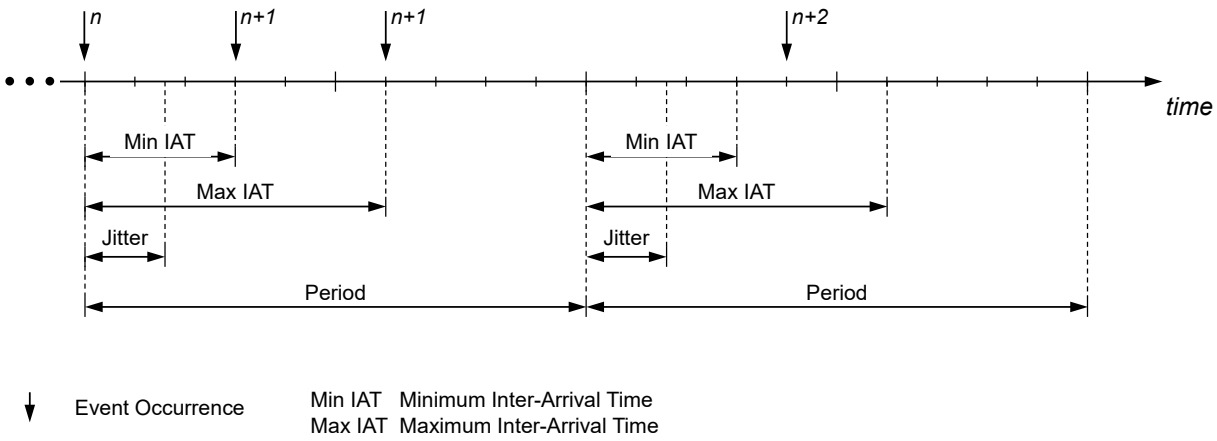


Figure 3.52: Parameters characterizing the `SporadicEventTriggering` Constraint

[constr_6822] Existence of `SporadicEventTriggering.maximumInterArrivalTime` ⌈For each `SporadicEventTriggering`, the attribute `maximumInterArrivalTime` shall exist **`at the imposition time associated with the concrete subclass of TimingDescriptionEvent`**.⌋*()*

[constr_6823] Existence of `SporadicEventTriggering.minimumInterArrivalTime` ⌈For each `SporadicEventTriggering`, the attribute `minimumInterArrivalTime` shall exist **`at the imposition time associated with the concrete subclass of TimingDescriptionEvent`**.⌋*()*

3.4.2.2.3 ConcretePatternEventTriggering

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

Class	**ConcretePatternEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	Describes the behavior of an event that occurs according to a precisely known pattern.			
Base	*ARObject*, *EventTriggeringConstraint*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ConcretePatternEventTriggering**			
offset	MultidimensionalTime	*	aggr	The offset for each occurrence of the event in the specified time interval. A list of point-in-times in the time interval given by the parameter patternLength at which the event occurs. **Tags:** xml.name=TIME-VALUE xml.roleElement=true xml.sequenceOffset=10 xml.typeElement=false
patternJitter	MultidimensionalTime	0..1	aggr	The maximum deviation of the time interval's starting point from the beginning of the given period. This parameter is only applicable in conjunction with the parameter `patternPeriod`.
patternLength	MultidimensionalTime	0..1	aggr	The duration of the time interval within which the event repeatedly occurs. The event occurs at concrete points in time within the given time interval. **Tags:** xml.sequenceOffset=20
patternPeriod	MultidimensionalTime	0..1	aggr	The time distance between the beginnings of subsequent repetitions of the given concrete pattern.

Table 3.62: ConcretePatternEventTriggering

Figure 3.53 and Figure 3.54 illustrate the parameters of the `ConcretePatternEventTriggering` in use.

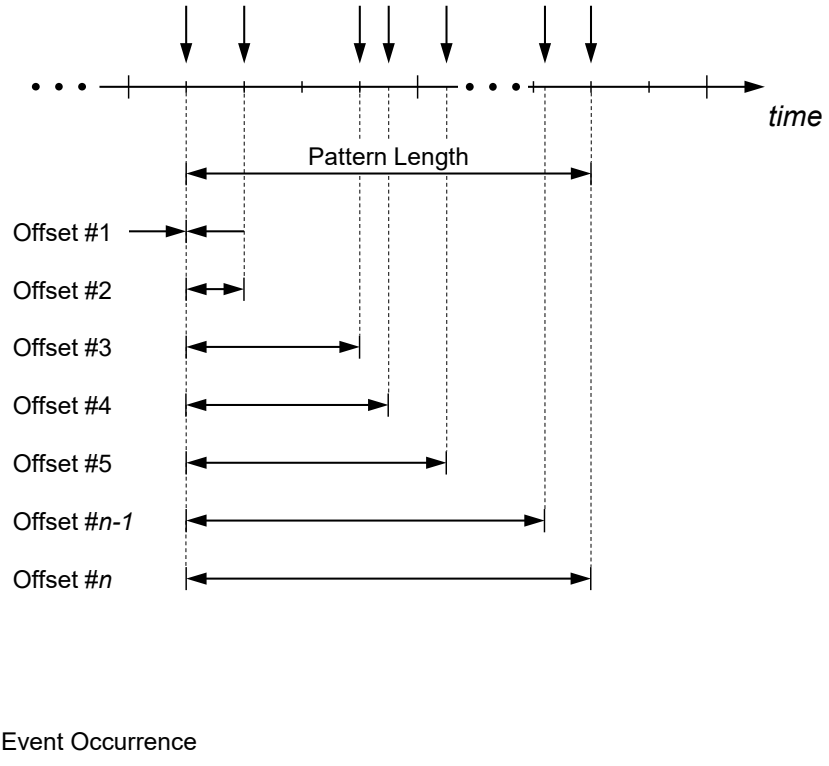


Figure 3.53: Parameters characterizing the `ConcretePatternEventTriggering` Constraint

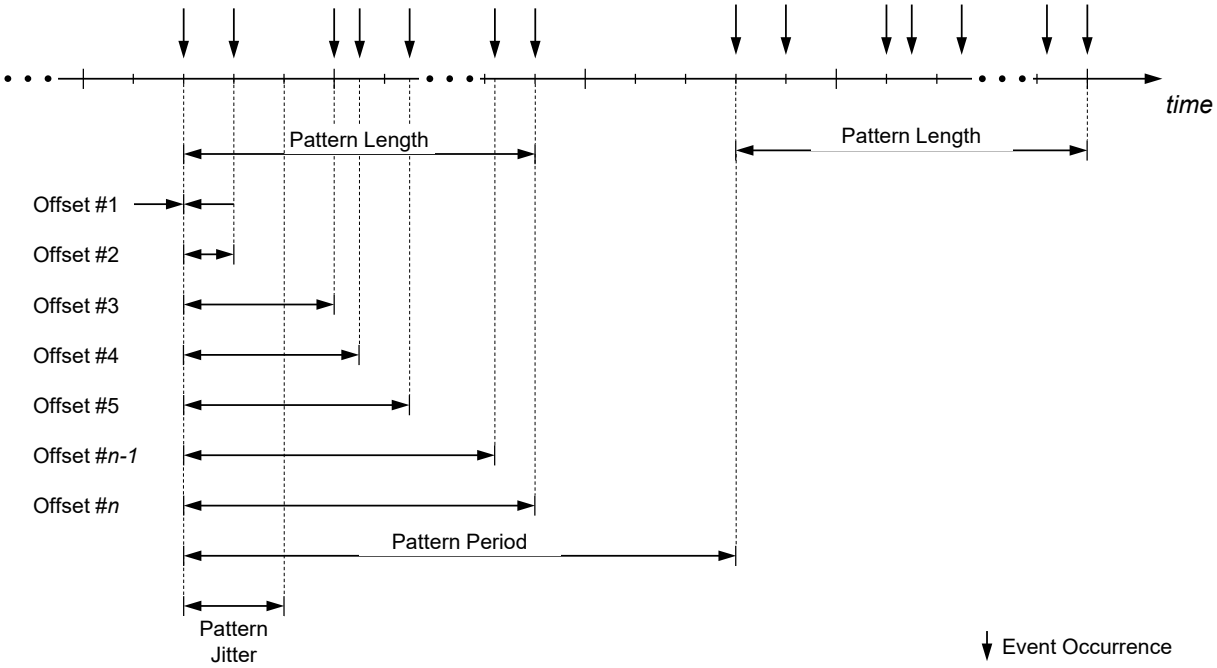


Figure 3.54: Parameters characterizing the `ConcretePatternEventTriggering` Constraint when periodically repeated

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

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

[constr_6824] Existence of `ConcretePatternEventTriggering.patternLength` ⌈For each `ConcretePatternEventTriggering`, the attribute `patternLength` shall exist **at the imposition time associated with the concrete subclass of `TimingDescriptionEvent`**.⌋*()*

[constr_6898] Existence of `ConcretePatternEventTriggering.offset` ⌈For each `ConcretePatternEventTriggering`, the attribute `offset` shall exist **at the imposition time associated with the concrete subclass of `TimingDescriptionEvent`**.⌋*()*

3.4.2.2.4 BurstPatternEventTriggering

[TPS_TIMEX_00013] `BurstPatternEventTriggering` specifies burst of occurrences of events ⌈The element `BurstPatternEventTriggering` is used to specify the characteristics of a burst of occurrences of one and the same `TimingDescriptionEvent`.⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008)*

Class	**BurstPatternEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	Describes the maximum number of occurrences of the same event in a given time interval. Typically used to model a worst case activation scenario.			
Base	*ARObject*, *EventTriggeringConstraint*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maxNumberOfOccurrences	PositiveInteger	0..1	attr	The maximum number of event occurrences within the given time interval. The event may never occur, or may occur N times between 1 and `maxNumberOfOccurrences`. If the parameter `minNumberOfOccurrences` is specified then the event occurs at least the number of times specified by `minNumberOfOccurrences` and at maximum by `maxNumberOfOccurrences`.
minimumInterArrivalTime	MultidimensionalTime	0..1	aggr	Specifies the minimum distance between subsequent occurrences of the event 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
patternJitter	MultidimensionalTime	0..1	aggr	The maximum deviation of the time interval's starting point from the beginning of the given period. This parameter is only applicable in conjunction with the parameter `patternPeriod`
patternLength	MultidimensionalTime	0..1	aggr	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 time distance between the beginnings of subsequent repetitions of the given burst pattern.

Table 3.63: BurstPatternEventTriggering

Figure 3.55 and Figure 3.56 illustrate the parameters of the `BurstPatternEvent-Triggering` in use.

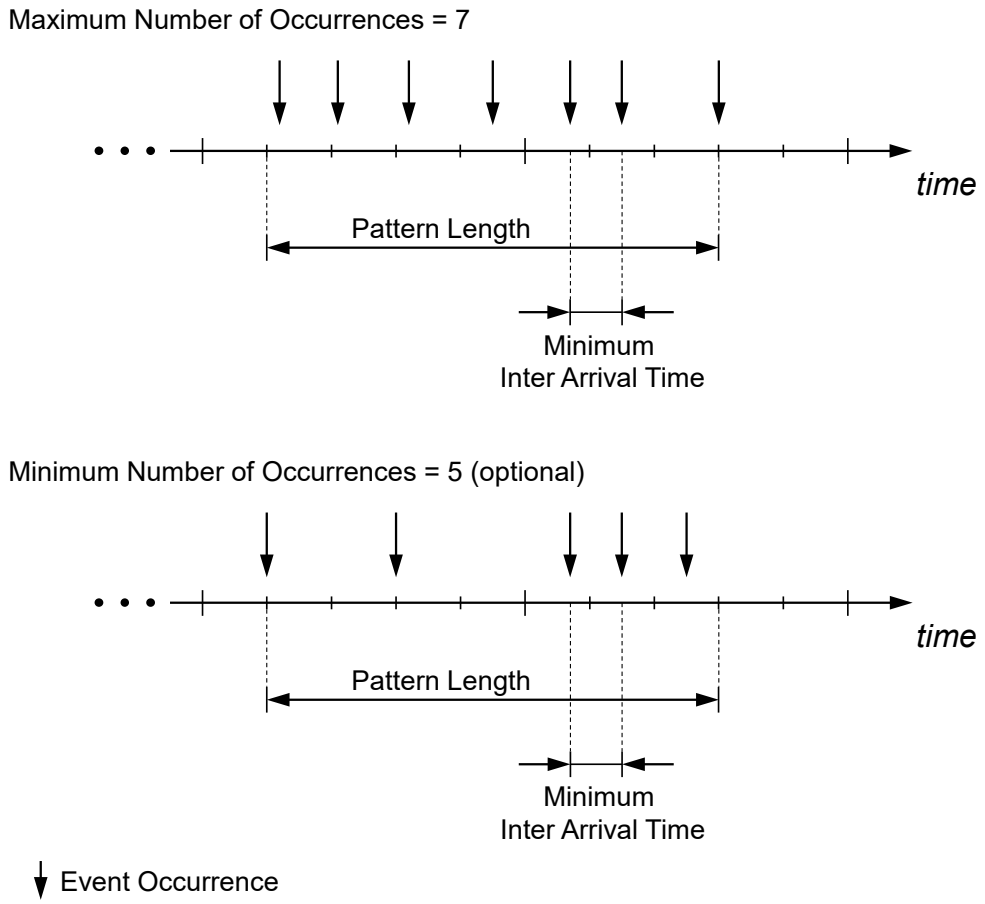


Figure 3.55: Parameters characterizing the `BurstPatternEventTriggering` Constraint

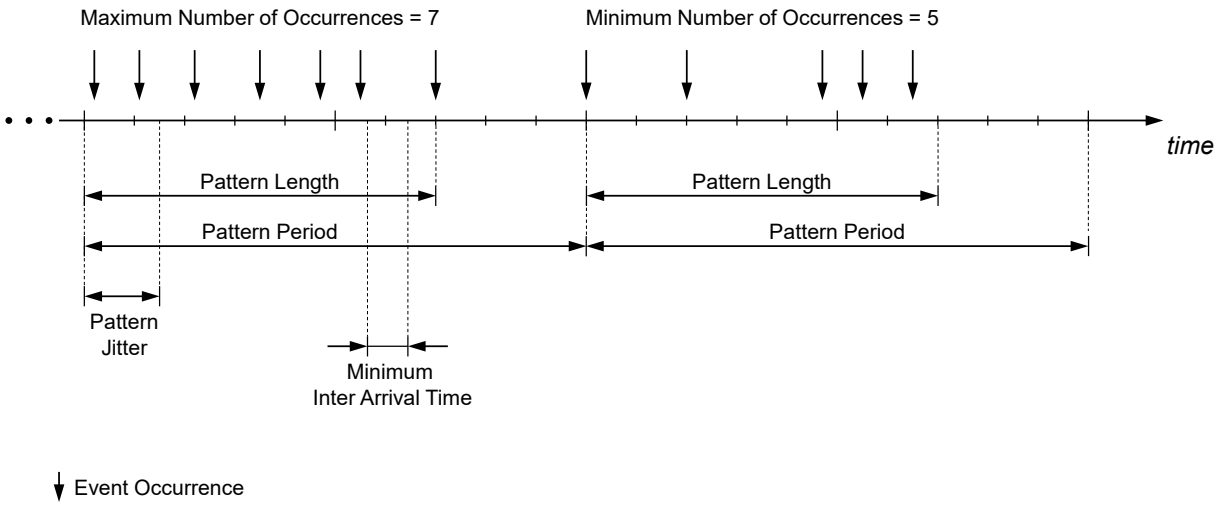


Figure 3.56: Parameters characterizing the `BurstPatternEventTriggering` Constraint when periodically repeated

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

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

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

[constr_6825] Existence of BurstPatternEventTriggering.maxNumberOfOccurrences ⌈For each BurstPatternEventTriggering, the attribute maxNumberOfOccurrences shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋*()*

[constr_6826] Existence of BurstPatternEventTriggering.minimumInterArrivalTime ⌈For each BurstPatternEventTriggering, the attribute minimumInterArrivalTime shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋*()*

[constr_6827] Existence of BurstPatternEventTriggering.patternLength ⌈For each BurstPatternEventTriggering, the attribute patternLength shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋*()*

3.4.2.2.5 ArbitraryEventTriggering

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

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

Class	**ArbitraryEventTriggering**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::EventTriggeringConstraint			
Note	Describes that an event occurs occasionally, singly, irregularly or randomly. The primary purpose of this event triggering is to abstract event occurrences captured by data acquisition tools (background debugger, trace analyzer, etc.) during system runtime.			
Base	*ARObject*, *EventTriggeringConstraint*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
confidenceInterval	ConfidenceInterval	*	aggr	List of confidence intervals. **Tags:** xml.sequenceOffset=30

▽

△

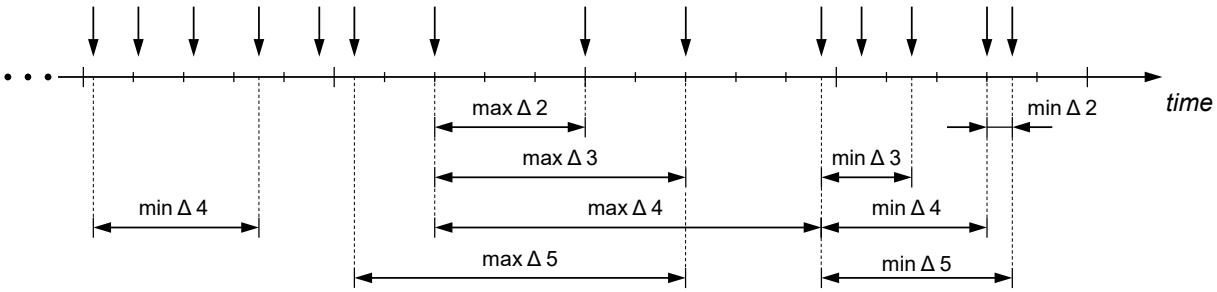
Class	**ArbitraryEventTriggering**			
maximum Distance	MultidimensionalTime	*	aggr	The nth array element describes the maximum distance that can be observed for a sample of n+1 event occurrences. This is an array with an identical number of elements as for the minimumDistance. **Tags:** xml.name=TIME-VALUE xml.roleElement=true xml.sequenceOffset=20 xml.typeElement=false
minimum Distance	MultidimensionalTime	*	aggr	The nth array element describes the minimum distance that can be observed for a sample of n+1 event occurrences. This is an array with an identical number of elements as for the maximumDistance. **Tags:** xml.name=TIME-VALUE xml.roleElement=true xml.sequenceOffset=10 xml.typeElement=false

Table 3.64: 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*			
Aggregated by	ArbitraryEventTriggering.confidenceInterval			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
lowerBound	MultidimensionalTime	0..1	aggr	The lower bound of the expected distance of two consecutive event occurrences.
propability	Float	0..1	attr	The probability for the measured lower and upper bound of the confidence interval.
upperBound	MultidimensionalTime	0..1	aggr	The upper bound of the expected distance of two consecutive event occurrences.

Table 3.65: ConfidenceInterval

Figure 3.57 illustrates the parameters of the `ArbitraryEventTriggering` in use.



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

Event Occurrence

Figure 3.57: Parameters characterizing the `ArbitraryEventTriggering` Constraint

[constr_6828] Existence of `ArbitraryEventTriggering.minimumDistance` ⌈For each `ArbitraryEventTriggering`, the reference in the role `minimumDistance` shall exist at least once **at the imposition time associated with the concrete subclass of `TimingDescriptionEvent`.**⌋*()*

[constr_6829] Existence of `ArbitraryEventTriggering.maximumDistance` ⌈For each `ArbitraryEventTriggering`, the reference in the role `maximumDistance` shall exist at least once **at the imposition time associated with the concrete subclass of `TimingDescriptionEvent`.**⌋*()*

[constr_6830] Existence of `ConfidenceInterval.lowerBound` ⌈For each `ConfidenceInterval`, the attribute `lowerBound` shall exist **at the imposition time associated with the concrete subclass of `TimingDescriptionEvent`.**⌋*()*

[constr_6831] Existence of `ConfidenceInterval.propability` ⌈For each `ConfidenceInterval`, the attribute `propability` shall exist **at the imposition time associated with the concrete subclass of `TimingDescriptionEvent`.**⌋*()*

[constr_6832] Existence of `ConfidenceInterval.upperBound` ⌈For each `ConfidenceInterval`, the attribute `upperBound` shall exist **at the imposition time associated with the concrete subclass of `TimingDescriptionEvent`.**⌋*()*

3.4.2.3 OffsetTimingConstraint

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.

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

Class	**OffsetTimingConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::OffsetConstraint			
Note	Bounds the time offset between the occurrence of two timing events, without requiring a direct functional dependency between the source and the target. If the target event occurs, it is expected to occur earliest with the minimum, and latest with the maximum offset relatively after the occurrence of the source event. Note: not every source event occurrence shall be followed by a target event occurrence. In contrast to LatencyTimingConstraint, there shall not necessarily be a causal dependency between the source and target event.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maximum	MultidimensionalTime	0..1	aggr	The maximum offset the target event occurs relatively after the occurrence of the source event. **Tags:** xml.sequenceOffset=20
minimum	MultidimensionalTime	0..1	aggr	The mimum offset the target event occurs relatively after the occurrence of the source event. **Tags:** xml.sequenceOffset=10
source	TimingDescriptionEvent	0..1	ref	The timing event that the target event is to be synchronized with.
target	TimingDescriptionEvent	0..1	ref	The timing event which is expected to occur timely after the source event.

Table 3.66: OffsetTimingConstraint

[constr_6842] Existence of OffsetTimingConstraint.maximum ⌈For each OffsetTimingConstraint, the attribute maximum shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋*()*

[constr_6843] Existence of OffsetTimingConstraint.minimum ⌈For each OffsetTimingConstraint, the attribute minimum shall exist **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋*()*

[constr_6844] Existence of OffsetTimingConstraint.source ⌈For each OffsetTimingConstraint, the reference in the role source shall exist at least once **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋*()*

[constr_6845] Existence of OffsetTimingConstraint.target ⌈For each OffsetTimingConstraint, the reference in the role target shall exist

at least once **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋()

3.4.2.4 AgeConstraint

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

Typically, the AgeConstraint restricts the time interval between the physical creation of data by the corresponding sender and the availability of the data in the receiver. The scope of an AgeConstraint is a TDEventVariableDataPrototype. Every time the scoped event occurs, the VariableDataPrototype shall have the specified data age.

At a later stage during the development, when the refined software architecture exposes the relation between the actual sender of the data and the receiver, an event chain between the sending and receiving point in time shall be defined and associated with a LatencyTimingConstraint (see 3.4.1.2) in order to refine the previous defined age constraint.

[TPS_TIMEX_00005] Semantics of an AgeConstraint ⌈AgeConstraint is used to specify a lower (minimum) and upper (maximum) bounded tolerance for a received VariableDataPrototype or Trigger.⌋(*RS_TIMEX_00001*, *RS_TIMEX_00009*)

Class	AgeConstraint			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::AgeConstraint			
Note	Constrains the scope by a minimum and maximum time boundary.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maximum	MultidimensionalTime	0..1	aggr	The received event referenced by scope should not exceed this upper bound.
minimum	MultidimensionalTime	0..1	aggr	The received event referenced by scope should not precede this lower bound.
scope	TimingDescriptionEvent	0..1	ref	TimingDescriptionEvent to be constrained.

Table 3.67: AgeConstraint

[constr_4504] Restriction of the scope of an AgeConstraint ⌈An AgeConstraint may only reference either a:

- TDEventVariableDataPrototype.tdEventVariableDataPrototypeType==variableDataPrototypeReceived
- TDEventTrigger.tdEventTriggerType==triggerActivated

in the role scope, **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋()

Since an AgeConstraint is intended to be used only in those scenarios without a sending context it is prudent to restrict it to the receiving side context of the intended sub-class of TimingDescriptionEvent and further constrain the referenced port context to receiver side PortPrototypes.

[constr_6914]{DRAFT} **Restriction of the port context of an AgeConstraint** ⌈An AgeConstraint.scope.port shall reference only sub-classes of AbstractRequiredPortPrototype **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋()

3.4.3 TimingConstraints on ExecutableEntities

TIMEX provides the means to specify constraints on ExecutableEntitys around execution order or an execution time etc. In contrast to the constraints on single events or event chains, the approach is a little different and is tied together with the use-cases e.g. 4.1, where several types of constraints are applied depending on the configuration. In this chapter, the general modeling is explained.

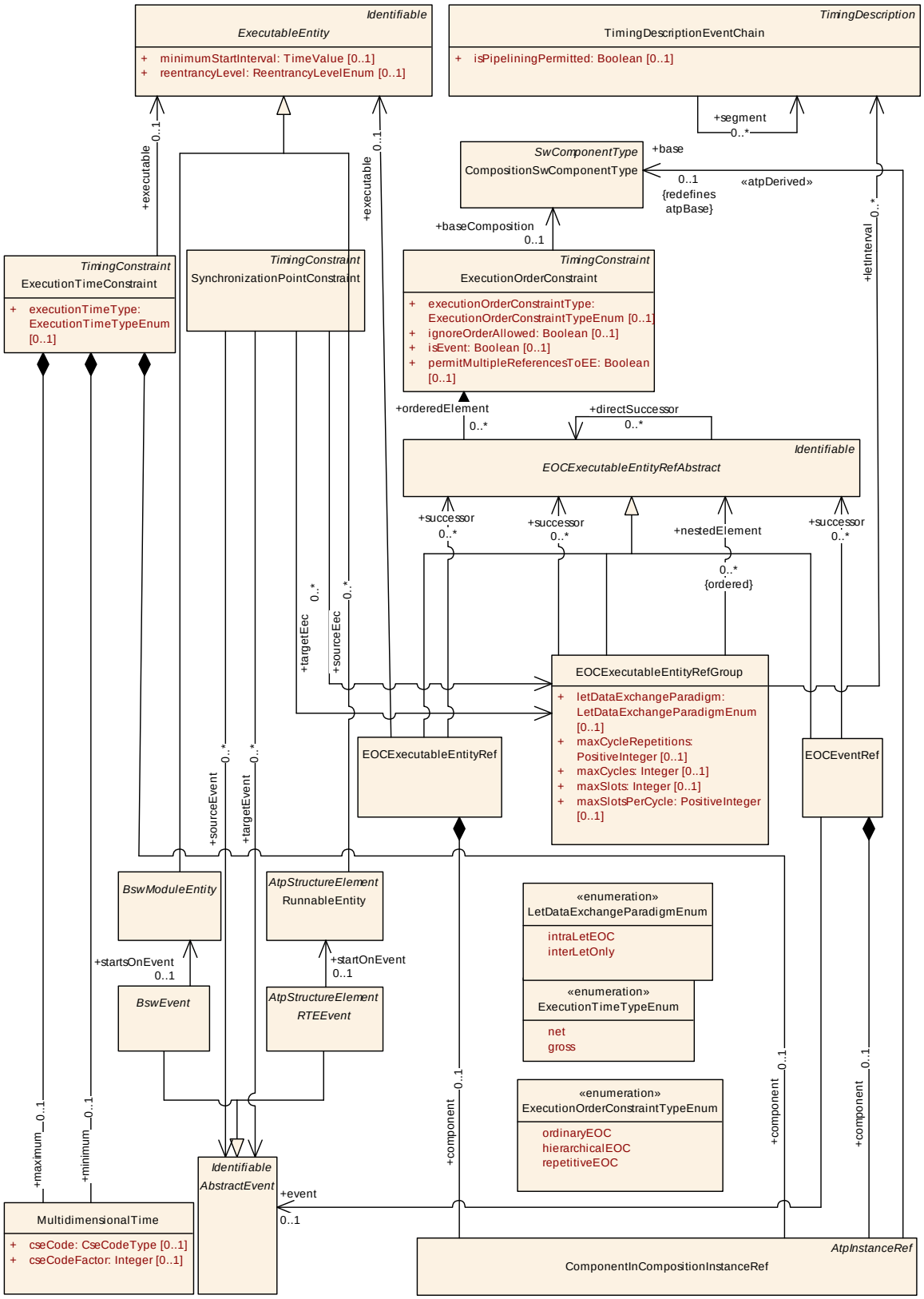


Figure 3.58: TimingConstraints on ExecutableEntitys

3.4.3.1 ExecutionOrderConstraint

An ExecutionOrderConstraint can be used in any TIMEX view, as long as the ExecutableEntitys and/or the AbstractEvents to be referenced are available in other AUTOSAR models, namely the [7] and [8].

[TPS_TIMEX_00007] ExecutionOrderConstraint constrains the ordering of sequences of ExecutableEntitys ⌈The element ExecutionOrderConstraint is used to specify the order of execution of a number of ExecutableEntitys.⌋ *(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00014)*

Class	**ExecutionOrderConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint			
Note	This constraint is used to restrict the order of execution for a set of ExecutableEntitys. The ExecutionOrderConstraint can be used in any timing view. The various scopes for ExecutionOrderConstraint are described below. Generally, each ExecutionOrder Constraint has a scope of software components and can reference all ExecutableEntitys available in the corresponding internal behavior (RunnableEntity and BswModuleEntity) either directly or by the events activating respectively starting them (RteEvent and BswEvent). On VFB level an ExecutionOrderConstraint can be specified for RunnableEntities part of the composition hierarchy referenced by the VfbTiming. On SW-C level an ExecutionOrderConstraint can be specified for RunnableEntities part of the Internal Behavior referenced by the SwcTiming. On System level an ExecutionOrderConstraint can be specified for RunnableEntities part of the composition hierarchy of the system referenced by the SystemTiming. On BSW Module level, an ExectionOrderConstraint can be specified for BswModuleEntities part of an BswInternalBehavior referenced by the BswModuleTiming. On ECU level an ExecutionOrderConstraint can be specified for all ExecutableEntitys and Events available via the EcucValueCollection, covering ECU Extract and BSW Module Configuration, referenced by the EcuTiming.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingConstraint*, *Traceable*			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base Composition	CompositionSw ComponentType	0..1	ref	Specifies the composition SW-C type playing the role of a SW-C containing further SW-Cs and represents the scope of the Execution Order Constraint.
executionOrder ConstraintType	ExecutionOrder ConstraintTypeEnum	0..1	attr	Specifies the specific type of ExecutionOrderConstraint.
ignoreOrder Allowed	Boolean	0..1	attr	Controls whether the order of execution specified by this constraint can be intentionally ignored (TRUE), or shall be respected (FALSE). **Tags:** atp.Status=obsolete
isEvent	Boolean	0..1	attr	Indicates whether the ExecutionOrderConstraint is only referring to Executable Entities (FALSE) or only to RTE and/or BSW Events (TRUE).
orderedElement	EOCExecutableEntity RefAbstract	*	aggr	This aggregation represents an unordered collection of references to RunnableEntities which shall be considered in the ExecutionOrderConstraint. The role does not imply that the collection of references itself shall be ordered.
permitMultiple ReferencesTo EE	Boolean	0..1	attr	Indicates that the ExecutionOrderConstraints permits that an Executable Entity is referenced multiple times (TRUE) or only once (FALSE) in the constraint.

Table 3.68: ExecutionOrderConstraint

Enumeration	**ExecutionOrderConstraintTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint
Note	Specifies the type of the executionOrderConstraintType for a ExecutionOrderConstraint.
Aggregated by	ExecutionOrderConstraint.executionOrderConstraintType
Literal	***Description***
hierarchicalEOC	Specifies that the Execution Order Constraint specifies a hierarchical execution order constraint. **Tags:** atp.EnumerationLiteralIndex=0
ordinaryEOC	Specifies that the Execution Order Constraint specifies an ordinary execution order constraint. **Tags:** atp.EnumerationLiteralIndex=1
repetitiveEOC	Specifies that the Execution Order Constraint specifies a repetitive execution order constraint. **Tags:** atp.EnumerationLiteralIndex=2

Table 3.69: ExecutionOrderConstraintTypeEnum

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

Table 3.70: EOCExecutableEntityRefAbstract

Class	**EOCExecutableEntityRefGroup**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint			
Note	This is used to specify a group (composite) consisting of Execution Order Constraint Executable Entity References (leaves) and/or further Execution Order Constraint Executable Entity Reference Groups (composite).			
Base	*ARObject*, *EOCExecutableEntityRefAbstract*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ExecutionOrderConstraint.orderedElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
letData Exchange Paradigm	LetDataExchange ParadigmEnum	0..1	attr	Specifies the data exchange paradigm between ExecutableEntitys within a LET interval. **Tags:** atp.Status=draft
letInterval	TimingDescriptionEvent Chain	*	ref	This association references the TimingDescriptionEvent Chain that plays the role of a LET interval the executable entities in the group are assigned to.
maxCycle Repetitions	PositiveInteger	0..1	attr	**Repetitive Execution Order Constraint only:** The number of repetitions (cycles) of the event in the Repetitive Execution Order Constraint. **Tags:** atp.Status=draft

△

Class	EOCExecutableEntityRefGroup			
maxCycles	Integer	0..1	attr	In case of a Repetitive Execution Order Constraint this attribute specifies the number of cycles the Execution Order Constraint is considering. **Tags:** atp.Status=obsolete
maxSlots	Integer	0..1	attr	In case of a Repetitive Execution Order Constraint this attribute specifies the number of slots every cycle of the Execution Order Constraint is consisting of. **Tags:** atp.Status=obsolete
maxSlotsPer Cycle	PositiveInteger	0..1	attr	**Repetitive Execution Order Constraint only:** The number of `ExecutableEntity`s (slots) that are executed in a given order within a cycle, for the Repetitive Execution Order Constraint. **Tags:** atp.Status=draft
nestedElement (ordered)	EOCExecutableEntity RefAbstract	*	ref	This association is used to establish hierarchies of EOCEER Groups and References.
successor	EOCExecutableEntity RefAbstract	*	ref	The logical successor of an executable entity or a group of executable entities.
triggeringEvent	TimingDescriptionEvent	0..1	ref	In case of a Repetitive Execution Order Constraint this association references the timing description event triggering every cycle.

Table 3.71: EOCExecutableEntityRefGroup

Class	EOCExecutableEntityRef			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint			
Note	This is used to define a reference to an ExecutableEntity If the ExecutionOrderConstraint is defined on VFB, System or ECU level, a reference to the Sw ComponentPrototype, via the ComponentInCompositionInstanceRef, the referenced ExecutableEntity belongs to, shall be provided as context information.			
Base	*ARObject*, *EOCExecutableEntityRefAbstract*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ExecutionOrderConstraint.orderedElement			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
bswModule Instance	BswImplementation	0..1	ref	Specifies the BSW module instance the BSW module entity belongs to.
component	SwComponent Prototype	0..1	iref	This association references the specific instance of the SW-C prototype. **InstanceRef implemented by:** ComponentIn CompositionInstanceRef
executable	ExecutableEntity	0..1	ref	The ExecutableEntity whose execution order is restricted by the contraint.
successor	EOCExecutableEntity RefAbstract	*	ref	The logical successor of an executable entity or a group of executable entities.

Table 3.72: EOCExecutableEntityRef

Class	EOCEventRef
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint
Note	This is used to define a reference to an RTE or BSW Event.
Base	*ARObject*, *EOCExecutableEntityRefAbstract*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**EOCEventRef**			
Aggregated by	ExecutionOrderConstraint.orderedElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bswModule Instance	BswImplementation	0..1	ref	Specifies the BSW module instance the BSW event is related to.
component	SwComponent Prototype	0..1	iref	This association references the specific instance of the SW-C prototype. **InstanceRef implemented by:** ComponentIn CompositionInstanceRef
event	AbstractEvent	0..1	ref	The AbstractEvent (event) whose execution order is restricted by the contraint.
successor	EOCExecutableEntity RefAbstract	*	ref	The logical successor of an executable entity or a group of executable entities.

Table 3.73: EOCEventRef

[TPS_TIMEX_00038] Purpose of `EOCExecutableEntityRefAbstract` ⌈The element `EOCExecutableEntityRefAbstract` is an abstract class that represents:

- `EOCExecutableEntityRefGroup` (composite)
- `EOCExecutableEntityRef` (atomic)
- `EOCEventRef` (atomic)

and uses these to create structures representing part-whole hierarchies.⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00014)*

[TPS_TIMEX_00047] Purpose of `ExecutionOrderConstraintTypeEnum` ⌈The element `ExecutionOrderConstraintTypeEnum` specifies whether the structure in [TPS_TIMEX_00038] represents a:

- `ordinaryEOC`
- `hierarchicalEOC`
- `repetitiveEOC`

⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00014)*

[TPS_TIMEX_00041] Purpose of `EOCExecutableEntityRefGroup` ⌈The element `EOCExecutableEntityRefGroup` is used to define composites of:

- `EOCExecutableEntityRefGroup`s
- `EOCExecutableEntityRef`s
- `EOCEventRef`s

⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00014)*

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

[TPS_TIMEX_00048] Purpose of `EOCEventRef` ⌈The element `EOCEventRef` is used to reference:

- `RTEEvent`s
- `BswEvent`s

in order to specify an execution order of `ExecutableEntity`s.⌋*(RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00014)*

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

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

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

[constr_4533] Maximum number of successor relationships ⌈The maximum number of successor relationships, namely `successor` or `directSuccessor`:

- between two `EOCExecutableEntityRef`s
- between two `EOCEventRef`s
- between two `EOCExecutableEntityRefGroup`s
- between an `EOCExecutableEntityRef` and an `EOCExecutableEntityRefGroup`
- between an `EOCEventRef` and an `EOCExecutableEntityRefGroup`

is one (1).⌋*()*

[constr_4534] Maximum number of `directSuccessor` relationships ⌈The number of `directSuccessor` relationships of a:

- `EOCExecutableEntityRef`
- `EOCEventRef`
- `EOCExecutableEntityRefGroup`

shall not exceed the number of independent execution units available in a system.⌋*()*

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

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

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

[constr_4546] Setting the attribute isEvent ⌈The value of the attribute isEvent shall be set to:

- TRUE: when the ExecutionOrderConstraint refers only to AbstractEvents
- FALSE: when the ExecutionOrderConstraint refers only to ExecutableEntitys

as per [constr_4545]⌋*()*

[constr_4547] Restriction of ExecutionOrderConstraint.permitMultipleReferencesToEE ⌈The attribute permitMultipleReferencesToEE shall exist only if ExecutionOrderConstraint.isEvent==FALSE as per [constr_4546] **at the time when the Swc Timing Description is complete**.⌋*()*

Rationale for [constr_4547]: The same ExecutableEntity may only be referenced more than once in case the ExecutionOrderConstraint shall solely refer to ExecutableEntitys (and not also AbstractEvents).

[constr_6833] Existence of ExecutionOrderConstraint.orderedElement ⌈For each ExecutionOrderConstraint, the attribute orderedElement shall exist at least once **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋*()*

[constr_6834] Existence of EOCExecutableEntityRefGroup.nestedElement ⌈For each EOCExecutableEntityRefGroup, the reference in the role nestedElement shall exist at least once **at the imposition time associated with the concrete subclass of TimingDescriptionEvent**.⌋*()*

3.4.3.1.1 Ordinary Execution Order Constraint

A ExecutionOrderConstraint.executionOrderConstraintType==ordinaryEOC is used to specify an order of execution of ExecutableEntitys.

As shown in Figure 3.59 the ExecutionOrderConstraint contains a number of EOCExecutableEntityRefs which reference the ExecutableEntitys the execution order is imposed on. The associations successor and directSuccessor are used to specify the type of successor relationship and enforce the order of execution.

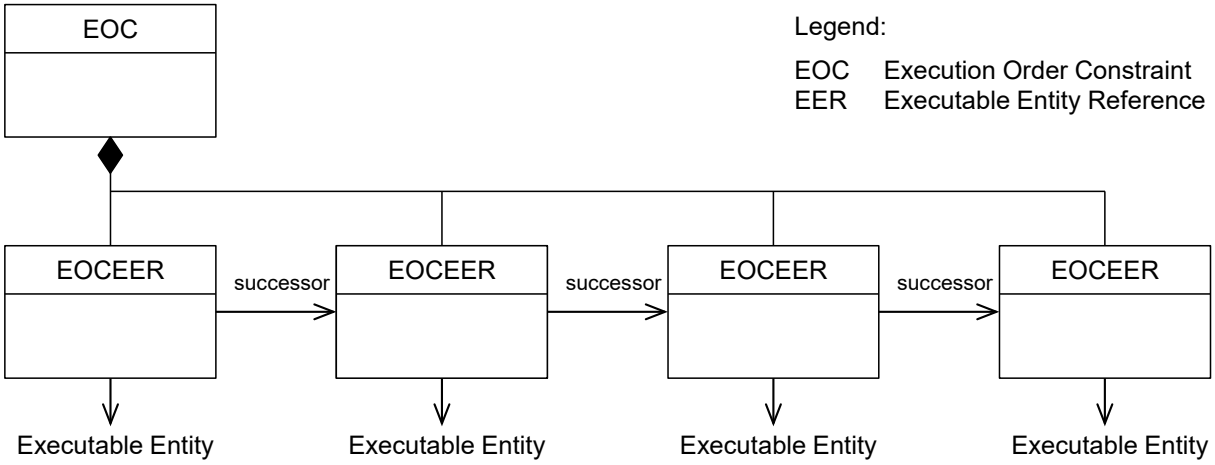


Figure 3.59: Example of an ExecutionOrderConstraint of type ordinaryEOC

In Figure 3.60, the ExecutionOrderConstraint contains a number of event references which refer to the specific RTEEvents/BswEvents the execution order is imposed on.

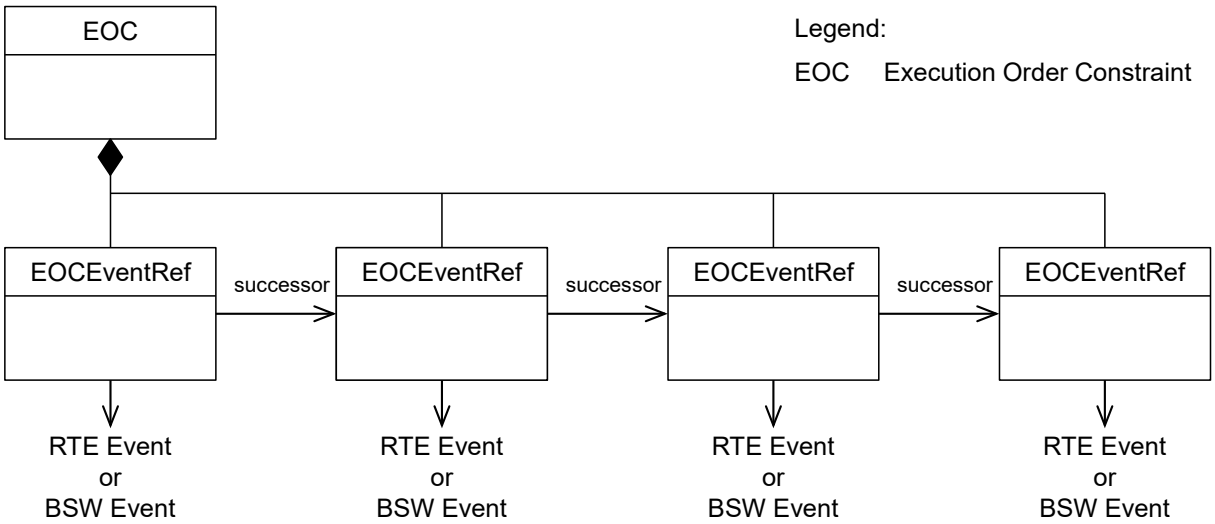


Figure 3.60: Example of an ExecutionOrderConstraint of type ordinaryEOC referencing RTEEvents/BswEvents

[constr_4541] `EOCExecutableEntityRef` shall reference `ExecutableEntity` in an `ExecutionOrderConstraint` of type `ordinaryEOC` ⌈In an ExecutionOrderConstraint.executionOrderConstraintType==ordinaryEOC all EOCExecutableEntityRefs shall reference an ExecutableEntity.⌋*()*

[constr_4548] `EOCEventRef` shall reference `AbstractEvent` in an `ExecutionOrderConstraint` of type `ordinaryEOC` ⌈In an ExecutionOrderConstraint.executionOrderConstraintType==ordinaryEOC all EOCEventRefs shall reference an AbstractEvent.⌋*()*

3.4.3.1.2 Hierarchical Execution Order Constraint

A ExecutionOrderConstraint.executionOrderConstraintType==hierarchicalEOC specifies an order of execution of ExecutableEntitys using the capability of creating groups of ExecutableEntitys. In other words, it enables to specify tree-like structures of EOCExecutableEntityRefGroups, EOCExecutableEntityRefs and EOCEventRefs.

As shown in Figure 3.61 the ExecutionOrderConstraint contains a number of EOCExecutableEntityRefs and one EOCExecutableEntityRefGroup, which in turn reference a number of EOCExecutableEntityRefs. The associations successor and directSuccessor between these EOCExecutableEntityRefs and the EOCExecutableEntityRefGroup are used to specify the type of successor relationship and enforce the order of execution.

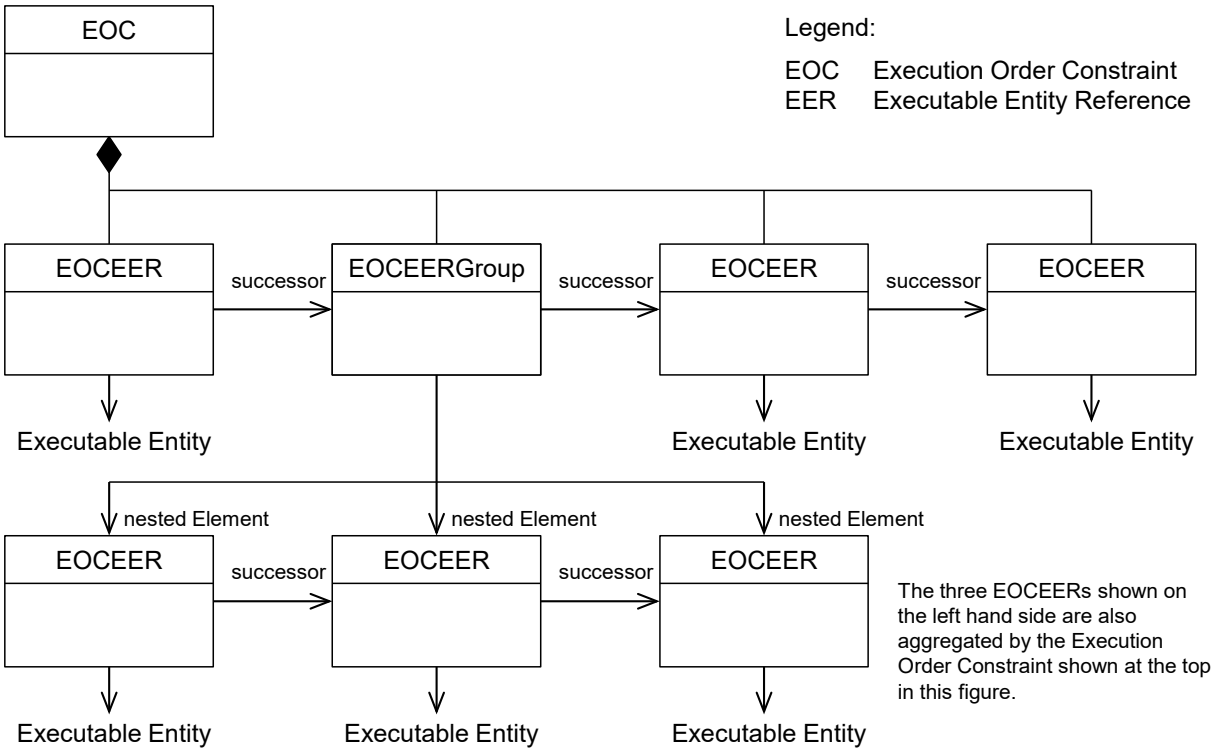


Figure 3.61: Example of a `ExecutionOrderConstraint` of type `hierarchicalEOC`

[constr_4523] Restriction of maxCycleRepetitions and maxSlotsPerCycle to Repetitive Execution Order Constraint ⌈The attributes

- EOCExecutableEntityRefGroup.maxCycleRepetitions
- EOCExecutableEntityRefGroup.maxSlotsPerCycle

shall exist only if *that* EOCExecutableEntityRefGroup is aggregated by an ExecutionOrderConstraint.executionOrderConstraintType==repetitiveEOC in the role orderedElement **at the time when the Swc Timing Description is complete**.⌋*()*

[constr_4538] Hierarchical Execution Order Constraint: EOCExecutableEntityRef, EOCEventRef, and EOCExecutableEntityRefGroup shall be target or source of a successor relationship ⌈In a ExecutionOrderConstraint.executionOrderConstraintType==hierarchicalEOC, each:

- EOCExecutableEntityRefGroup
- EOCExecutableEntityRef
- EOCEventRef

which is not part of an EOCExecutableEntityRefGroup shall be target or source of at least one successor relationship.⌋*()*

[constr_4542] EOCExecutableEntityRef shall reference ExecutableEntity in Hierarchical Execution Order Constraint ⌈In a ExecutionOrderConstraint.executionOrderConstraintType==hierarchicalEOC all EOCExecutableEntityRefs shall reference an ExecutableEntity.⌋*()*

[constr_4549] EOCEventRef shall reference AbstractEvent in Hierarchical Execution Order Constraint ⌈In a ExecutionOrderConstraint.executionOrderConstraintType==hierarchicalEOC all EOCEventRefs shall reference an AbstractEvent.⌋*()*

[constr_4550] A Hierarchical Execution Order Constraint shall have an unambiguous root EOCExecutableEntityRefGroup ⌈A ExecutionOrderConstraint.executionOrderConstraintType==hierarchicalEOC may contain multiple orderedElements, which may be any combination of any number of EOCExecutableEntityRefs respectively EOCEventRefs and EOCExecutableEntityRefGroups. Amongst these, there needs to be exactly one EOCExecutableEntityRefGroup being neither target nor source of any successor or directSuccessor relationship. This EOCExecutableEntityRefGroup is the *root* of the Hierarchical Execution Order Constraint.⌋*()*

[TPS_TIMEX_00129]{DRAFT} **Representation of the root in a Hierarchical Execution Order Constraint** ⌈In a ExecutionOrderConstraint.executionOrderConstraintType==hierarchicalEOC the root in the hierarchy shall be the EOCExecutableEntityRefGroup.category==ROOT_GROUP.⌋*()*

[constr_6909]{DRAFT} **Singleton ROOT_GROUP in a Hierarchical Execution Order Constraint** ⌈In a ExecutionOrderConstraint.executionOrderConstraintType==hierarchicalEOC, in the tree of orderedElements, there shall be only **one** EOCExecutableEntityRefGroup.category==ROOT_GROUP **at the time when the Swc Timing Description is complete**.⌋()

[constr_6910]{DRAFT} **Referencing from a ROOT_GROUP in a Hierarchical Execution Order Constraint** ⌈In a ExecutionOrderConstraint.executionOrderConstraintType==hierarchicalEOC, in the tree of orderedElements, in the singularly identifiable ROOT_GROUP (according to [constr_6909]):

- the successor shall not exist
- the directSuccessor shall not exist

at the time when the Swc Timing Description is complete.⌋()

[constr_6911]{DRAFT} **Referencing to a ROOT_GROUP in a Hierarchical Execution Order Constraint** ⌈In a ExecutionOrderConstraint.executionOrderConstraintType==hierarchicalEOC, in the tree of orderedElements, the singularly identifiable ROOT_GROUP (according to [constr_6909]) shall not be referenced in the role:

- successor by another EOCExecutableEntityRefGroup
- directSuccessor by another EOCExecutableEntityRefGroup
- nestedElement by another EOCExecutableEntityRefGroup

at the time when the Swc Timing Description is complete.⌋()

Note: [constr_4537] forbids that the same subclass of EOCExecutableEntityRefAbstract is used in multiple ExecutionOrderConstraints.

3.4.3.1.3 Repetitive Execution Order Constraint

The ExecutionOrderConstraint.executionOrderConstraintType==repetitiveEOC is used to specify *varying* ExecutionOrderConstraints depending on subsequent occurrences of a specific event. This enables one to specify that specific ExecutionOrderConstraints are imposed on a given number of ExecutableEntitys whenever the particular TimingDescriptionEvent occurs.

For example:

- if the event A occurs:
  - the 1st time; the ExecutableEntitys: one (1), two (2) and three (3)
  - the 2nd time; the ExecutableEntitys: one (1), four (4) and five (5)
  - the 3rd time; the ExecutableEntitys: one (1), two (2) and three (3)

  - the 4th time; the ExecutableEntitys: one (1), four (4) and five (5)
  - ...
- ...shall be executed in this given order

The occurrences of the specified event are called *cycles* and the order of the ExecutableEntitys within a cycle is arranged by *slots*.

As shown in Figure 3.62 the Repetitive Execution Order Constraints follows a specific pattern.

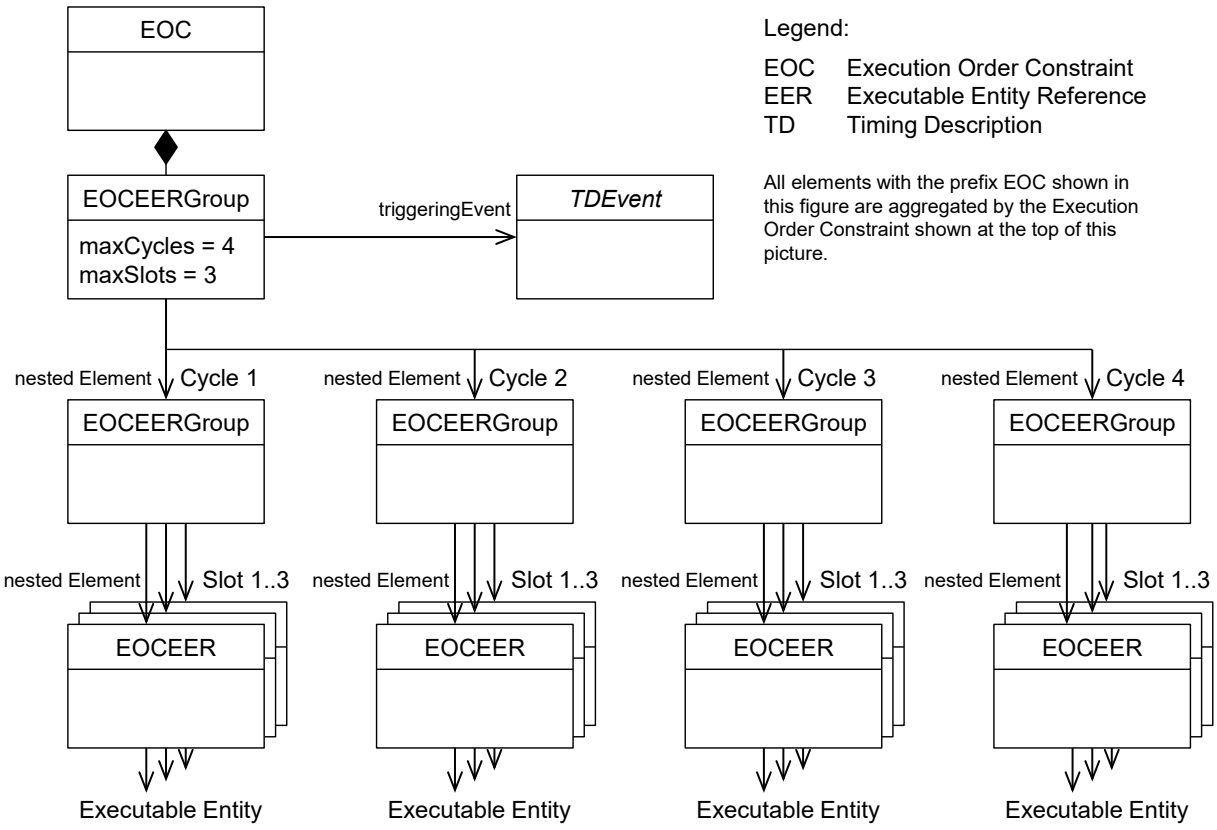


Figure 3.62: Example of a Repetitive Execution Order Constraint

The executionOrderConstraintType==repetitiveEOC follows the pattern of the Hierarchical Execution Order Constraint, but some restrictions apply to the use and structure of groups of executable entity references. The ExecutionOrderConstraint consists of one group of executable entity references, called the *root* group, which references only other groups of executable entity references.

These groups in turn reference executable entity references respectively event references which eventually reference the specific ExecutableEntitys respectively events. The *root* group specifies the maximum number of cycles maxCycles and the maximum number of slots maxSlots.

The maxCycles specifies the number of subsequent event occurrences after which the execution order constraint repeats, hence the name Repetitive Execution Order

Constraint; the maxSlots specifies the number of ExecutableEntitys that are executed in a given order within a cycle.

Note: maxCycles, respectively *cycle*, is a synonym for the term *maxRepetitions*, respectively *repetition*.

The table below presents the repetitive execution order constraint shown in Figure 3.62 in a tabular way.

Cycle	Slot 1	Slot 2	Slot 3
Cycle1	RE1	RE2	RE3
Cycle2	RE1	RE4	RE5
Cycle3	RE7	RE8	RE9
Cycle4	RE1	RE8	RE6

Table 3.74: Example Repetitive Execution Order Constraint

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

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

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

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

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

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

[constr_4530] An EOCExecutableEntityRefGroup representing a cycle shall reference only EOCExecutableEntityRefs respectively EOCEventRefs ⌈The EOCExecutableEntityRefGroup representing a cycle shall reference only EOCExecutableEntityRefs, respectively EOCEventRefs.⌋*()*

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

[constr_4539] The successor relationships successor and directSuccessor shall not be used ⌈The successor relationships successor and directSuccessor shall not be used in a executionOrderConstraintType==repetitiveEOC.⌋()

[constr_6907]{DRAFT} **Restriction of EOCExecutableEntityRefGroup.triggeringEvent** ⌈The TimingDescriptionEvent referenced in the role EOCExecutableEntityRefGroup.triggeringEvent shall exist only if the EOCExecutableEntityRefGroup is transitively aggregated by an ExecutionOrderConstraint.executionOrderConstraintType==repetitiveEOC in the role orderedElement **at the imposition time associated with the concrete subclass of TimingExtension**.⌋()

3.4.3.2 ExecutionTimeConstraint

An AUTOSAR ResourceConsumption.executionTime can be used to describe the execution time of an ExecutableEntity. The concept is described in [8]. This execution time description represents a timing property of a ExecutableEntity.

Class	**ExecutionTimeConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionTimeConstraint			
Note	Constrains the execution time of the referenced executable in component between a minimum and maximum interval. The time to execute the executable 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*			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
component	SwComponent Prototype	0..1	iref	The component that containts the referenced Executable Entity for the ExecutionTimeConstraint. If the entity is in a basic software module no component shall be provided. **InstanceRef implemented by:** ComponentIn CompositionInstanceRef
executable	ExecutableEntity	0..1	ref	The referenced ExecutableEntity for the ExecutionTime Constraint.
executionTime Type	ExecutionTimeType Enum	0..1	attr	Specifies the type of the execution time constrained by ExecutionTimeConstraint,
maximum	MultidimensionalTime	0..1	aggr	The maximum execution time.
minimum	MultidimensionalTime	0..1	aggr	The minimum execution time.

Table 3.75: ExecutionTimeConstraint

Enumeration	**ExecutionTimeTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionTimeConstraint
Note	Specifies the type of the executionTimeType for a ExecutionTimeConstraint.
Aggregated by	ExecutionTimeConstraint.executionTimeType

▽

△

Enumeration	**ExecutionTimeTypeEnum**
Literal	***Description***
gross	Indicates that the given execution time is the time used to execute the executable WITHOUT any interruption and WITH external calls. **Tags:** atp.EnumerationLiteralIndex=0
net	Indicates that the given execution time is the time used to execute the executable WITHOUT any interruption and WITHOUT any external calls. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.76: ExecutionTimeTypeEnum

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

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).

[constr_6835] Existence of ExecutionTimeConstraint.executionTimeType ⌈For each ExecutionTimeConstraint, the attribute executionTimeType shall exist **at the time when the Swc Timing Description is complete**.⌋*()*

[constr_6836] Existence of ExecutionTimeConstraint.executable ⌈For each ExecutionTimeConstraint, the reference to ExecutableEntity in the role executable shall exist **at the time when the Swc Timing Description is complete**.⌋*()*

3.4.3.3 SynchronizationPointConstraint

The capabilities of an ExecutionOrderConstraint are not sufficient to *explicitly* express that the execution of one or more ExecutableEntitys shall be started if and only if one or more other ExecutableEntitys have finished execution. In other words, one or more ExecutableEntitys shall succeed the execution of one or more other ExecutableEntitys when those ExecutableEntitys have all finished their execution. In order to specify this behavior the SynchronizationPointConstraint is used which specifies synchronization points between ExecutableEntitys, referenced by their corresponding RTEEvents or BswEvents, or EOCExecutableEntityRefGroups.

Note that the element SynchronizationTimingConstraint is also related to synchronization, but this element imposes *temporal* constraints on the occurrence of events, like the occurrence of activation, start and termination of an executable entity. For example, a number of ExecutableEntitys shall terminate and a number

of executable entities shall start within a given time interval. But it is not possible to specify that the ExecutableEntitys shall terminate in this time interval *before* the ExecutableEntitys start in this time interval. Therefore, this capability is not sufficient to express a synchronization point.

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

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

Table 3.77: SynchronizationPointConstraint

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

4 Application

This chapter presents how aspects in 3 are combined to achieve particular higher-level application of timing.

4.1 Logical Execution Time

Logical Execution Time (LET) is a real-time programming *abstraction*. It abstracts from actual (*physical*) execution time of a real-time program. LET determines the time it takes from reading program input to writing program output regardless of the time it takes to execute the program which processes the input and writes output. LET is motivated by the observation that the relevant behavior of real-time programs is determined by when input is read and output is written and not when programs just execute any code.

This *logical* notion is shown in the upper part of Figure 4.1. At the *release* point input is read and at the *terminate* point output is written. Between these two points — the logical execution time — the program is executed, and writing output is delayed until the logical execution time elapsed even if the program completes execution before the logical execution time elapsed. This is sketched in the lower part of Figure 4.1 and shows the actual *physical* execution of a program.

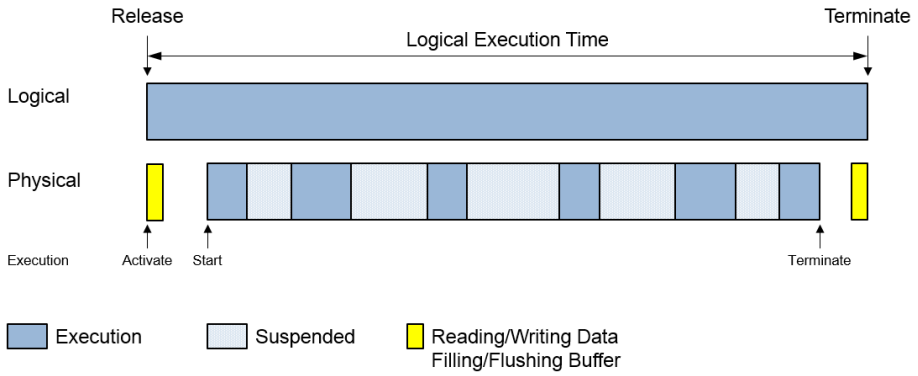


Figure 4.1: LET Interval

4.1.1 LET Interval Definition

Modeling of LET starts with a categorized TimingDescriptionEventChain. The TimingDescriptionEventChain.category==LET_INTERVAL to represent the LET interval span. This TimingDescriptionEventChain shall reference two TDEventComplexs to represent the stimulus (see [TPS_TIMEX_00111]) and the response (see [TPS_TIMEX_00114]) points.

The TDEventComplex is used because both the stimulus and response are not referencing an *observable* location, like the other types of TimingDescriptionEvents. A TDEventOccurrenceExpression on the TDEventComplex is used to specify the particular TimingDescriptionEvent supposed to occur in case of the release and terminate point respectively.

[TPS_TIMEX_00057] Representation of an LET interval in a TimingDescriptionEventChain ⌈An LET Interval shall be described by a TimingDescriptionEventChain.category==LET_INTERVAL⌋*(RS_TIMEX_00022)*

[TPS_TIMEX_00112]{DRAFT} **Representation of an LET interval release** ⌈An LET Interval release shall be described by stimulus referencing a TDEventComplex.category==LET_RELEASE⌋*(RS_TIMEX_00022)*

[TPS_TIMEX_00113]{DRAFT} **Representation of an LET interval terminate** ⌈An LET Interval terminate shall be described by response referencing a TDEventComplex.category==LET_TERMINATE⌋*(RS_TIMEX_00022)*

After defining the stimulus and response events, the timing related parameters of the an LET interval are specified using TimingConstraints. A LatencyTimingConstraint specifies the **duration** and a PeriodicEventTriggering specifies the **recurrence** of an LET interval.

[TPS_TIMEX_00121]{DRAFT} **Representation of the duration of an LET interval** ⌈The **duration** of an LET interval shall be described by a LatencyTimingConstraint where:

- latencyConstraintType==reaction
- maximum shall specify the **duration** semantics.

⌋*()*

[TPS_TIMEX_00117]{DRAFT} **Representation of the recurrence of an LET interval** ⌈The **recurrence** of an LET interval shall be described by a PeriodicEventTriggering referencing a TimingDescriptionEvent in the role event.⌋*()*

In the context of [TPS_TIMEX_00121], the attribute minimum has no meaning and shall be ignored.

[TPS_TIMEX_00102]{DRAFT} **Optionality of LatencyTimingConstraint.minimum used in an LET interval** ⌈For a LatencyTimingConstraint.latencyConstraintType==reaction the attribute minimum shall be ignored if specified.⌋*()*

It is only necessary to have a This PeriodicEventTriggering constraint shall be imposed on the stimulus event (the release point of the LET interval) of the timing description event chain representing a LET interval.

[TPS_TIMEX_00122]{DRAFT} **Application of a PeriodicEventTriggering constraint used to specify the *recurrence* of an LET interval** ⌈For a PeriodicEventTriggering constraint used to configure the *recurrence* semantics of an LET interval,

the reference event shall refer to the TDEventComplex used in the role TimingDescriptionEventChain.stimulus.⌋()

[constr_6912]{DRAFT} **Mandatory specification of LET interval recurrence** ⌈For a TDEventComplex.category==LET_RELEASE, there shall exist a PeriodicEventTriggering referencing that TDEventComplex.category==LET_RELEASE in the role event **at the time when the Swc Timing Description is complete**.⌋()

Figure 4.2 shows an example LET model with ARXML listings C.2, C.3, C.4.

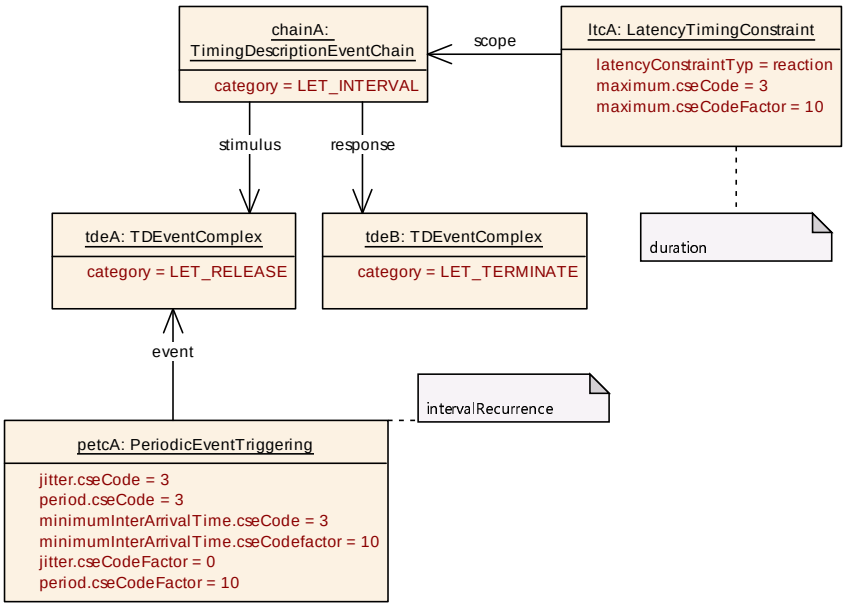


Figure 4.2: LET Interval Model

4.1.2 Relation between LET Intervals

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

In Figure 4.3 a periodic LET interval is specified. The LET interval is periodically repeated and the value of the LET interval is the same as the period.

Figure 4.3: Specifying Periodic LET Interval

The corresponding ARXML fragments that describe this pattern are shown in listing C.2, C.3 and C.4. The first listing presents an ARXML fragment that specifies the LET interval; the second listing presents an ARXML fragment that specifies the duration of the LET interval; and the third listing presents an ARXML fragment that specifies the recurrence of the LET interval.

In Figure 4.4 a periodic LET interval is specified, but the value of the period is larger than the LET interval. This results in a gap between the *terminate* point of the LET interval and the next *release* point of the LET interval. The corresponding ARXML fragment is shown in listing C.5. The first part of the ARXML fragments specifies the duration of the LET interval which is 6ms by using the latency timing constraint `LatencyTimingConstraint`. The second part specifies the recurrence of the LET interval which is 10ms by using the periodic event triggering constraint `PeriodicEventTriggering`.

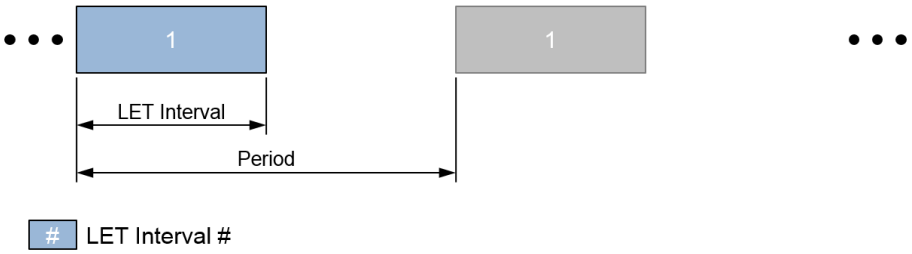


Figure 4.4: Specifying Periodic LET Interval with Gap

In Figure 4.5 three LET intervals are specified. The duration of the first LET interval 1 is 2.5ms, the duration of the second LET interval 2 is 5ms, and the duration of the third LET interval 3 is 10ms. The very first release of each LET interval takes place at the same point in time and then each of the LET intervals is repeated periodically. This case is modeled using the Offset Timing Constraint `OffsetTimingConstraint` and the offset between the *release* points of those LET intervals is 0ms, namely no offset.

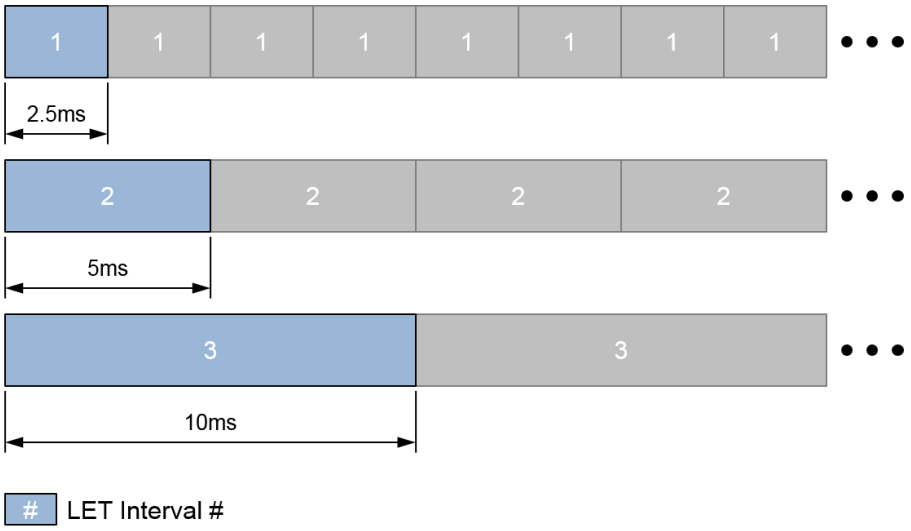


Figure 4.5: Specifying Relationship between Periodic LET Intervals

The listing C.6 shows an ARXML fragment which specifies two offset timing constraints between the *release* point of LET interval 1 and the two *release* points of the LET interval 2 and LET interval 3. In both cases the offset is zero (0ms) specifying that all LET intervals are released at the same point in time.

In Figure 4.6 three LET intervals are specified. The duration of the first LET interval 1 is 2.5ms, the duration of the second LET interval 2 is 5ms, and the duration of the third LET interval 3 is 10ms. The very first release of each LET interval takes place at different points in time and then each LET interval is repeated periodically . This case is modeled using the Offset Timing Constraint `OffsetTimingConstraint`. The offset between the *release* points of the first LET interval 1 and second LET interval 2 is 2.5ms; and the offset between the *release* points of the first LET interval 1 and third LET interval 3 is 7.5ms.

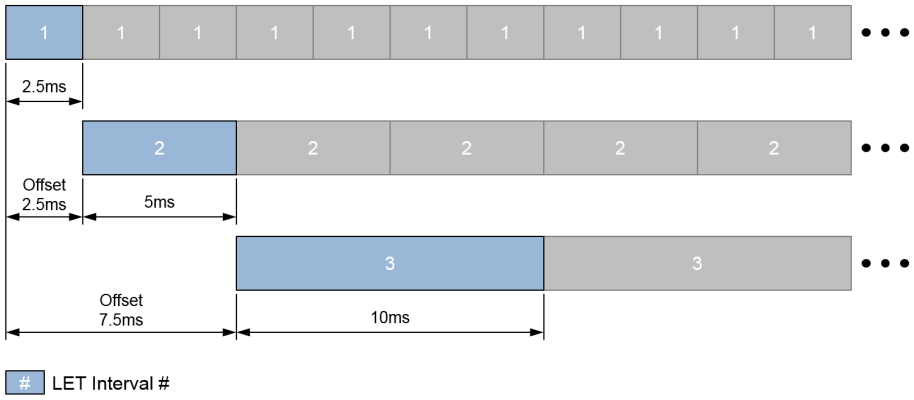


Figure 4.6: Specifying Relationship between Periodic LET Intervals With Offsets

The listing C.7 shows an ARXML fragment which specifies two offset timing constraints between the *release* point of LET interval 1 and the two *release* points of the LET

interval 2 and LET interval 3. In the first case the offset between the *release* points of LET interval 1 and LET interval 2 is 2.5ms; and in the second case the offset between the *release* points of LET interval 1 and LET interval 3 is 7.5ms.

In Figure 4.7 four LET intervals are specified. The first LET interval 1 terminates before the second LET interval 2 is released; the second LET interval 2 terminates before the third LET interval 3 is released; and the third LET interval 3 terminates before the fourth LET interval is released. After the last, the fourth LET interval 4 terminates, the same sequence of LET intervals is repeated. This case is modeled using the Offset Timing Constraint `OffsetTimingConstraint` to specify the offsets between the *release* point of the first LET interval 1 and the *release* points of each LET interval 2 through 4. The duration of each LET interval is listed in Table 4.1 along with the corresponding offsets between the *release* points of the LET intervals.

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

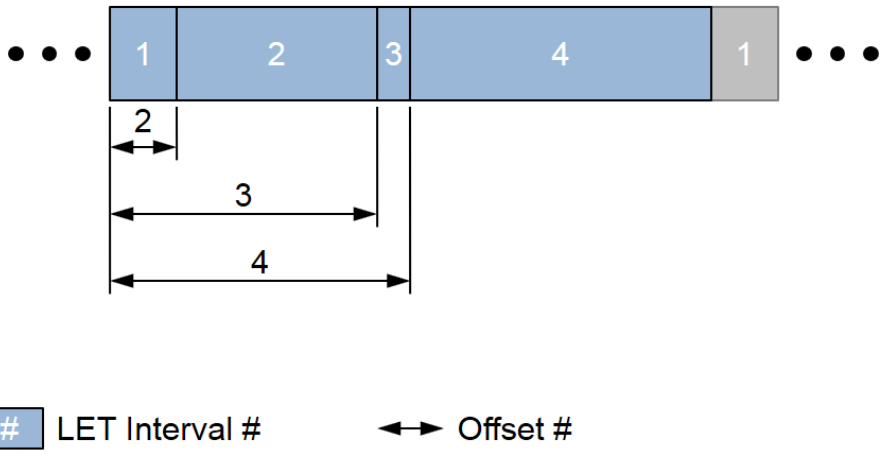


Figure 4.7: Specifying Arbitrary LET Intervals

Table 4.1 below summarizes the duration of the LET intervals and the offsets between the release point of the LET interval 1 and the release points of the LET intervals 2 through 4 as shown in Figure 4.7.

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

Table 4.1: Specifying Arbitrary LET Intervals

The listing C.8 shows an ARXML fragment which specifies the three offsets between the *release* point of the first LET interval 1 and the *release* points of the three LET intervals 2 through 4.

In Figure 4.8 a similar case as described in Figure 4.7 is shown but the offset between some LET intervals and the duration of some LET intervals are different which result in some gaps between LET intervals. The duration of each LET interval and the offsets between the LET intervals of this case are listed in Table 4.2 along with the corresponding offsets between the *release* points of the LET intervals.

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

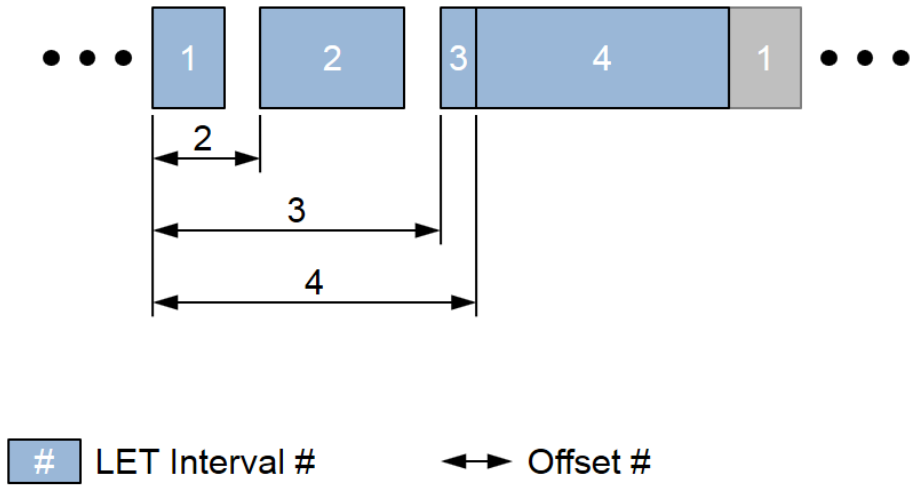


Figure 4.8: Specifying Arbitrary LET Intervals with Gaps

Table 4.2 below summarizes the duration of the LET intervals and the offsets between the release point of the LET interval 1 and the release points of the LET intervals 2 through 4 as shown in Figure 4.8.

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

Table 4.2: Specifying Arbitrary LET Intervals with Gaps

The ARXML fragment of this example is the same as shown in listing C.8 except the values of the offset timing constraints, and the latency timing constraints, which specify the duration of the four LET intervals.

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

In Figure 4.9 four LET intervals are specified which overlap. The LET interval 3 and LET interval 4 are overlapping each other and overlapping LET interval 1 and LET interval 2, too.

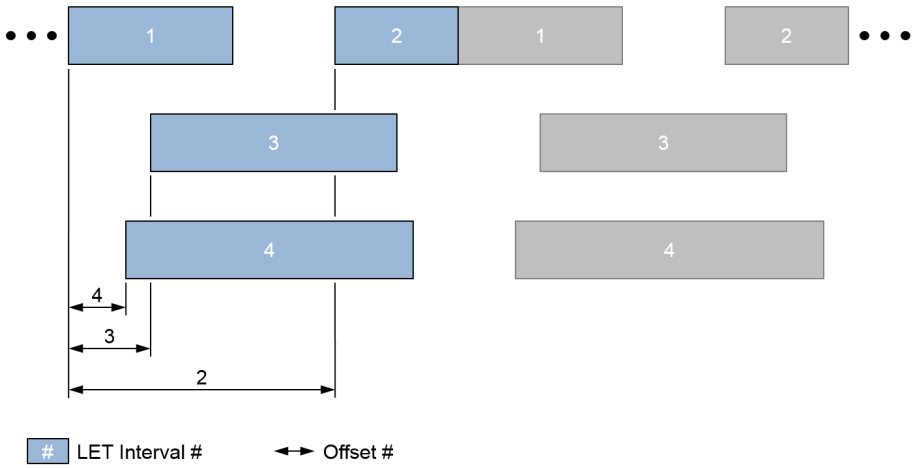


Figure 4.9: Specifying Arbitrary LET Intervals with Overlap

Table 4.3 below summarizes the duration of the LET intervals and the offsets between the release point of the LET interval 1 and the release points of the LET intervals 2 through 4 as shown in Figure 4.9.

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

Table 4.3: Specifying Arbitrary LET Intervals with Overlap

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

The listing C.9 shows an ARXML fragment which specifies three offset timing constraints between the *release* point of LET interval 1 and the three *release* points of the LET interval 2 through LET interval 4. The values of the offsets are selected such that LET interval 3 and LET interval 4 are overlapping the LET interval 1 and LET interval 2. The offset between the *release* points of LET interval 1 and LET interval 2 is 13ms; the offset between the *release* point of LET interval 1 and LET interval 3 is 4ms; and the offset between the *release* point of LET interval 1 and LET interval 4 is 3ms.

4.1.3 Executable Entity Cluster to LET Interval mapping

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

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

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

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

As an example, assume there are 8 x AtomicSwComponentTypes: $Swc01$ .. $Swc08$. Each of these SWCs has an internalBehavior $ib\_Swc0\#$ where the number sign indicates one of the SWCs $Swc01$ .. $Swc08$. Each SWC has 1 x RunnableEntity called $re\_re01\_Swc0\#$ and a Timing Event called $te\_re01\_Swc0\#$ is specified and references the corresponding RunnableEntity in the SWC.

The RunnableEntitys have data read and data write access points which read data from the ports of the particular SWC and write data to the provided ports of the particular SWC. The data dependency graph among the RunnableEntitys is shown in Figure 4.10. For this example three executable entities clusters are defined: The first executable entities cluster consists of the runnable entities re_re01_Swc01, re_re01_Swc02 and re_re01_Swc04; the second one consists of the runnable entities re_re01_Swc03 and re_re01_Swc07; and the third executable entities cluster consists of the runnable entities re_re01_Swc05, re_re01_Swc06 and re_re01_Swc08. Three LET intervals are defined in the example and are called $LETinterval1$, $LETinterval2$ and $LETinterval3$. The first executable entities cluster is mapped to the first LET interval using the association called letInterval between the element EOCExecutableEntityRefGroup and the element TimingDescriptionEventChain.

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

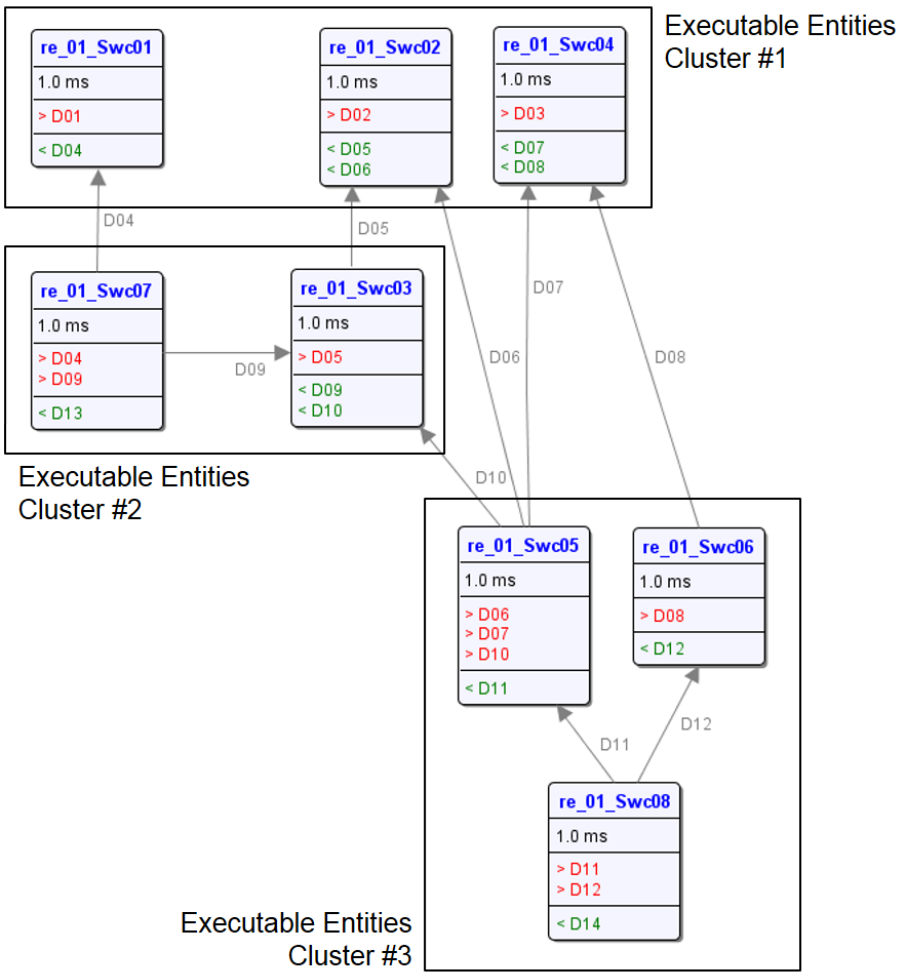


Figure 4.10: Data Dependencies among Executable Entities

The listing C.10 shows an ARXML fragment which specifies that an executable entities cluster and that it is mapped to the $LETinterval1$.

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

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

[constr_4554] Restriction of the referenced `TimingDescriptionEventChain` for a `letInterval` ⌈The element `EOCExecutableEntityRefGroup.letInterval` shall be present only in a `ROOT_GROUP` (according to [constr_6909]) **`at the time when the Swc Timing Description is complete`**.⌋*()*

[constr_6913]{DRAFT} **Restriction on `RTEEvent`s used in an LET interval** ⌈An `EOCExecutableEntityRefGroup` which references a `TimingDescriptionEventChain.category==LET_INTERVAL` in the role `letInterval` **and** transitively

references an EOCEventRef in the role successor **or** references an EOCEventRef in the role nestedElement, those EOCEventRefs shall reference either:

- TimingEvent in the role event or
- BswTimingEvent in the role event

at the time when the Swc Timing Description is complete.⌋()

4.1.4 Data flow within an LET Interval

The release and terminate event of a letInterval define when input data is expected to be read and output data to be written. This provides a unique data flow for data exchange between ExecutableEntitys that are part of EOCExecutableEntityRefGroups which reference different LET intervals.

For all ExecutableEntitys in EOCExecutableEntityRefGroups that relate to a single letInterval, two paradigms are possible of how data is exchanged **within** an letInterval are possible - in Figure 4.11.

Enumeration	**LetDataExchangeParadigmEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::ExecutionOrderConstraint
Note	Specifies the data exchange paradigm between ExecutableEntitys within a LET interval. **Tags:** atp.Status=draft
Aggregated by	EOCExecutableEntityRefGroup.letDataExchangeParadigm
Literal	***Description***
interLetOnly	All ExecutableEntitys mapped to this LET interval exchange data ONLY at the release and terminate event of the LET interval. This allows for a straightforward translation of the required label buffering but results in longer end-to-end latencies (multiple of the period). The execution order of \ARMetaClass{Executable Entity}s within the LET interval does not affect the data flow. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=draft
intraLetEOC	The ExecutableEntitys that belong to the same EOCExecutableEntityRefGroup and are mapped to this LET interval are executed in the order defined by the EOCExecutableEntityRefGroup and exchange data directly within this LET interval according to implicit semantics. Only at the borders of the LET interval or between independent EOCExecutableEntityRefGroups, is data propagated according to the LET paradigm. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=draft

Table 4.4: LetDataExchangeParadigmEnum

[TPS_TIMEX_00128]{DRAFT} **Default letDataExchangeParadigm** ⌈The default letDataExchangeParadigm=intraLetEOC.⌋()

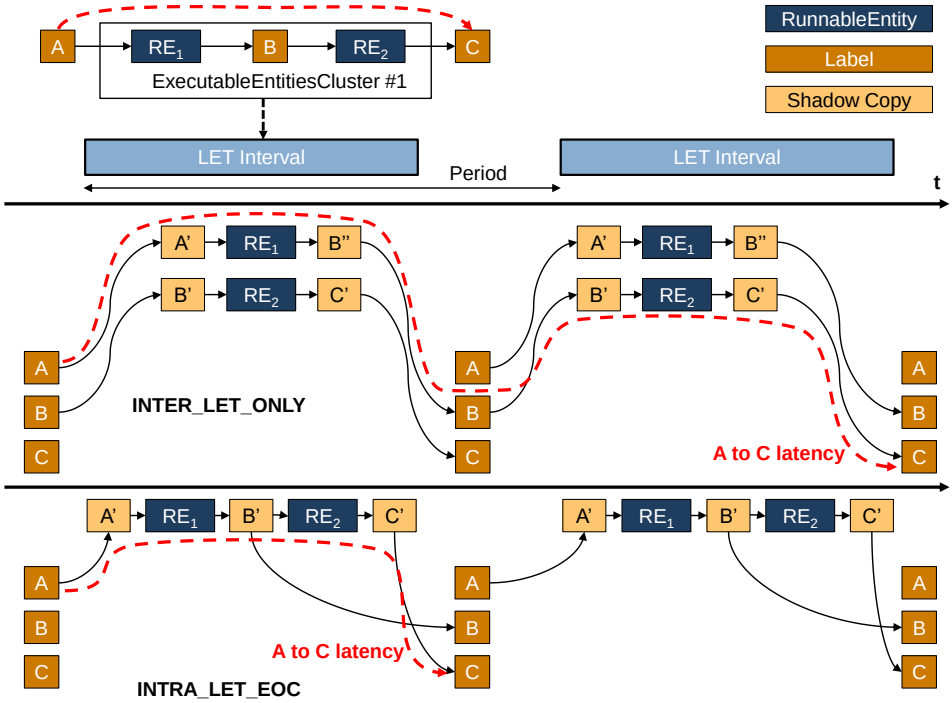


Figure 4.11: Data exchange paradigms within an LET interval

The `letDataExchangeParadigm` shall only apply to `EOCExecutableEntityRefGroup`s used in the context of an LET interval and shall therefore be constrained to reflect that.

[constr_6908]{DRAFT} **Restriction of `EOCExecutableEntityRefGroup.letDataExchangeParadigm`** ⌈The attribute `letDataExchangeParadigm` shall exist only if the `letInterval` in the same same `EOCExecutableEntityRefGroup` references a `TimingDescriptionEventChain.category==LET_INTERVAL` **at the time when the Swc Timing Description is complete**.⌋*()*

4.2 System Level Logical Execution Time

System Level Logical Execution Time (SL-LET) builds upon the concept of Logical Execution Time but targets a system-level view, e.g. between ports on (heterogenous) CP ECUs and/or AP Machines [9].

While the scope of LET in AUTOSAR is to specify a time interval for the execution of runnable entities (and therefore CP only) on a local ECU, SL-LET:

- addresses the data flow on arbitrary ports (e.g., ports of functional blocks, SWCs, services) and allows to specify latencies (resp. SL-LET interval durations) larger than the period,
- allows to incorporate distributed communication by explicitly addressing distributed clocks that have a bounded synchronization accuracy,
- can be applied already on the functional level and decomposed throughout the development process.

The assumption of instantaneous SL-LET events for read and write actions implies that those actions take place in logically zero time. A correct implementation of SL-LET must preserve the data flow semantics of the SL-LET model.

4.2.1 SL-LET Interval Definition

Modeling of SL-LET starts with a categorized TimingDescriptionEventChain.

[TPS_TIMEX_00125]{DRAFT} **Representation of an SL-LET interval in a `TimingDescriptionEventChain`** ⌈An SL-LET Interval shall be described by a TimingDescriptionEventChain.category==SL_LET_INTERVAL⌋*(RS_TIMEX_00026)*

The TimingDescriptionEventChain.category==SL_LET_INTERVAL shall reference two TDEventSLLETs to represent the stimulus (see [TPS_TIMEX_00111]) and the response (see [TPS_TIMEX_00114]).

[TPS_TIMEX_00115]{DRAFT} **Representation of an SL-LET interval `release`** ⌈An SL-LET Interval release shall be described by stimulus referencing a TDEventSLLET.category==SL_LET_RELEASE⌋*(RS_TIMEX_00022)*

[TPS_TIMEX_00116]{DRAFT} **Representation of an SL-LET interval `terminate`** ⌈An SL-LET Interval terminate shall be described by response referencing a TDEventSLLET.category==SL_LET_TERMINATE⌋*(RS_TIMEX_00022)*

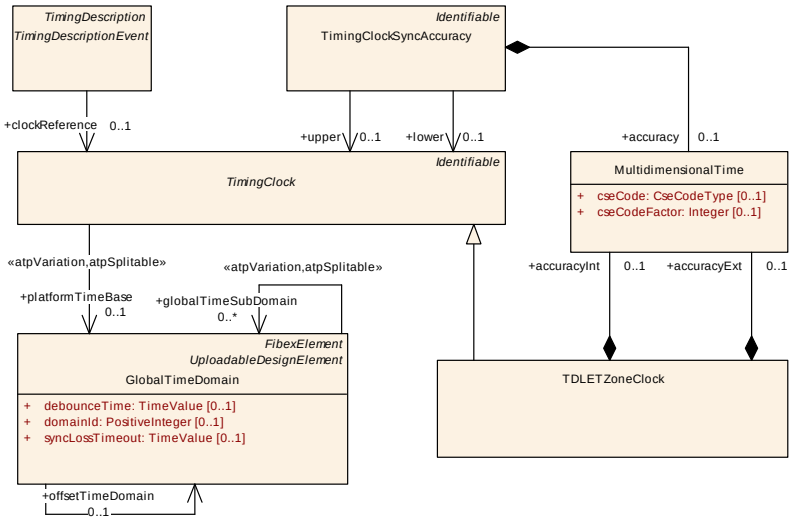


Figure 4.12: TDLETZoneClock and TimingClockSyncAccuracy

The attribute `TimingDescriptionEventChain.isPipeliningPermitted` is provided to specify whether the `TimingDescriptionEventChain.category==SL_LET_INTERVAL` **must** use a pipelined execution or not [9].

[constr_6816]{DRAFT} **Restricted usage of `TimingDescriptionEventChain.isPipeliningPermitted` in `TimingDescriptionEventChain`** ⌈The attribute `isPipeliningPermitted` shall only exist if the `TimingDescriptionEventChain.category==SL_LET_INTERVAL`.⌋*()*

A `TimingDescriptionEvent` used in this context may reference an optional `TimingDescriptionEvent.clockReference` (4.2.2) but shall be locked out for other contexts other than SL-LET.

[constr_6817]{DRAFT} **Restricted usage of `TimingDescriptionEvent.clockReference`** ⌈The reference `TimingDescriptionEvent.clockReference` shall exist if (and only if), the `TimingDescriptionEvent` has a:

- `category==SL_LET_RELEASE` or
- `category==SL_LET_TERMINATE`

⌋*()*

The listing in C.12 shows an ARXML which specifies the timing description event chain and timing description events. Pictorially, figure 4.13 shows a simplified model of an SL-LET interval.

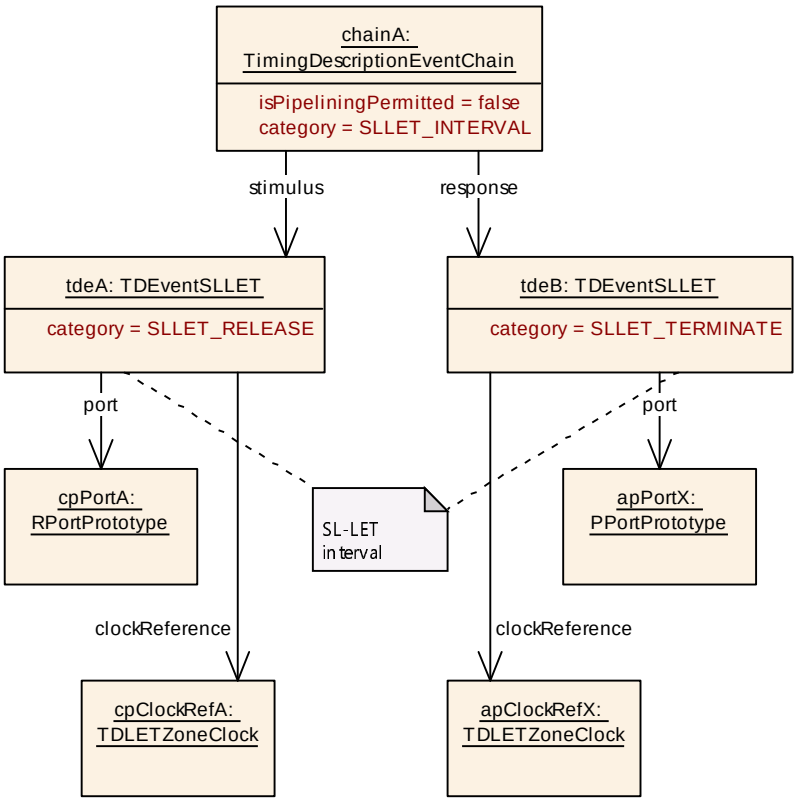


Figure 4.13: SL-LET Interval Model

The timing parameters of an SL-LET interval are specified in the same fashion as an LET interval and shall be specified according to the explanation in 4.1.1. Example ARXMLs are shown in C.12 (duration) and C.12 (recurrence).

4.2.1.1 Execution Order

Figure 4.14 shows an example regarding *execution order*. The release and terminate event of the SL-LET interval $SLLET_1$ is related to the required port of $Swc01$ and the provided port of $Swc02$. This implies the execution order $R_1 \rightarrow R_2 \rightarrow R_3$ for the runnable entities. The example is platform agnostic.

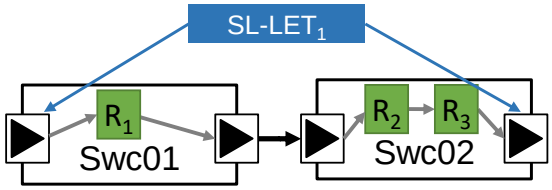


Figure 4.14: Execution order with SL-LET events related to TDEventSLLET

4.2.2 SL-LET Time Zones

(LET Zones) are an extension within the domain of SL-LET. The background is that timing description events in a timing event chain *may* have actual different physical time bases (LET Zone Clocks) and therefore need to reference those LET Zone Clocks discretely to determine synchronization accuracy for a SL-LET interval.

In a deployment view LET Zone Clocks may be structured in a tree-based hierarchy. As the tree is descended from a high-level to a low-level view, thus the time bases (LET Zone Clock)s move from a logical high-level e.g. vehicle-level to a e.g. logical low-level sensor/actuator-level.

Figure 4.15 shows a graphical example of these concepts. In this example 3 LET-zones are used to represent the hierarchy, in principal any number of LET Zones could be possible but the synchronization accuracy is always between the upper and lower LET zones (timing description event end points).

A LET-zone called TZ_A is represented by a TDLETZoneClock. By referencing the corresponding TimingClocks within the TimingDescriptionEvents, a SL-LET interval can either be specified between the LET zones TZ_1 and TZ_2 or on a higher hierarchical level with respect to LET zone TZ_0.

The LET-zone TZ_B is a lower LET-zone of TZ_A, if there exists a TimingClockSyncAccuracy.upper referring to a TDLETZoneClock A and a lower pointing to TDLETZoneClock B. This definition can be applied recursively, so if TZ_C is a lower LET-zone of TZ_B and TZ_B is a lower LET-zone of TZ_A, then TZ_C is also a lower LET-zone of TZ_A.

The internal synchronization accuracy requirement $\epsilon_{0,int}$ (attribute accuracyInt of TDLETZoneClock) is therefore used as a wrapper to abstract lower LET-zones of TZ_0.

Different LET zone clocks (which can be specified by TDLETZoneClock) are synchronized in a hierarchical manner. This relation can be specified by the element TimingClockSyncAccuracy. Figure 4.12 shows the relation between the elements.

The corresponding ARXML specification is shown in listing C.12.

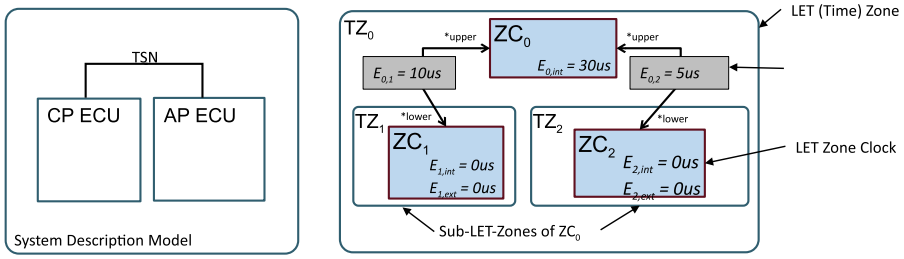


Figure 4.15: Exemplary definition of three LET Zones within a vehicle

In Figure 4.15, the internal synchronization accuracy requirement of $\epsilon_{0,int} = 30\mu s$, which is given as an attribute of TDLETZoneClock, is fulfilled, since the worst-case between

any two lower LET-zones of TZ_0 is $\epsilon_{1,ext} + \epsilon_{0,1} + \epsilon_{0,2} + \epsilon_{2,ext} = 0\mu s + 10\mu s + 5\mu s + 0\mu s = 15\mu s \leq 30\mu s$.

Figure 4.16 shows an example when the LET zone TZ_2 shall include further lower LET zones. At the end, the SL-LET interval SL-LET-1 will reference the zone clock ZC_1 and the - not yet defined - zone clock ZC_x. Such a hierarchical concretization can be constrained in advance, by specifying the attributes `accuracyExt` and `accuracyInt` of `TDLETZoneClock`.

Without knowledge of the specific lower LET zones, a SL-LET interval SL-LET-1 can be specified with respect to ZC_2. If both, the release as well as the terminate event of the SL-LET interval relate to ZC_2, the internal synchronization accuracy `accuracyInt` of ZC_2 has to be considered. This abstracts the synchronization between any two lower LET-zones of TZ_2. If only one of the two events relates to ZC_2 and the other one for example relates to ZC_1, the external synchronization accuracy `accuracyExt` of ZC_2 has to be considered. This abstracts the synchronization accuracy between a lower LET-zone of TZ_2 and any foreign LET-zone.

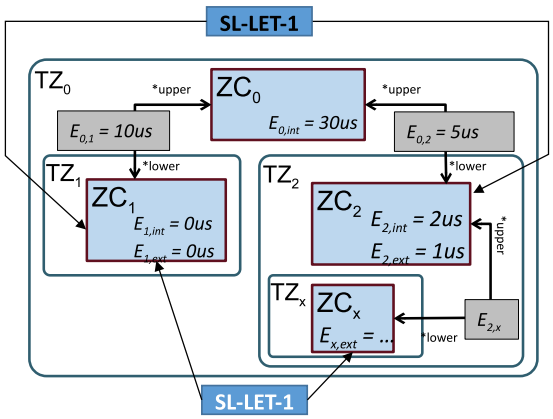


Figure 4.16: Exemplary definition of hierarchical LET Zones

4.2.2.1 SL-LET Zone Clocks

This section details the `TimingClock`.

[TPS_TIMEX_00123]{DRAFT} **Purpose of `TimingClock`** ⌈The meta-class `TimingClock` provides and abstract model of a timing extensions time base.⌋*(RS_TIMEX_00001)*

[TPS_TIMEX_00119]{DRAFT} **Purpose of `TDLETZoneClock`** ⌈The meta-class `TDLETZoneClock` is a specialization of `TimingClock` and represents a `LET Zone Clock`.⌋*(RS_TIMEX_00026)*

[TPS_TIMEX_00118]{DRAFT} **Usage of `TimingClock.platformTimeBase`** ⌈The reference `TimingClock.platformTimeBase` refers to an actual time base `GlobalTimeDomain` on CP/AP platform level.⌋*(RS_TIMEX_00001)*

[TPS_TIMEX_00100]{DRAFT} **Optionality of `accuracyInt`** ⌈If the attribute `accuracyInt` is unspecified, it shall be treated as zero. This is the case, if a LET-zone has no lower LET-zones.⌋*(RS_TIMEX_00026)*

[TPS_TIMEX_00101]{DRAFT} **Semantics of `accuracyInt`** ⌈The attribute `accuracyInt` constrains an upper-bounds for any combination of two `accuracy`s of referenced `TimingClockSyncAccuracy`s plus the `accuracyExt`s of the `TDLETZoneClock`s, that are referenced by those `TimingClockSyncAccuracy`s.⌋*(RS_TIMEX_00026)*

[TPS_TIMEX_00103]{DRAFT} **Optionality of `accuracyExt`** ⌈If the attribute `accuracyExt` is unspecified, it shall be treated as zero. This is the case, if a LET-zone has no lower LET-zones.⌋*(RS_TIMEX_00026)*

[TPS_TIMEX_00104]{DRAFT} **Semantics of `accuracyExt`** ⌈The attribute `accuracyExt` constrains an upper-bounds for any `TimingClockSyncAccuracy.accuracy` plus the `accuracyExt` of the `TDLETZoneClock`, that is referenced by the `TimingClockSyncAccuracy`.⌋*(RS_TIMEX_00026)*

4.2.2.2 SL-LET Zone Clock Accuracy

[TPS_TIMEX_00105]{DRAFT} **Purpose of `TimingClockSyncAccuracy`** ⌈A `TimingClockSyncAccuracy` is used to describe the synchronization accuracy between two `TimingClock`s.⌋*(RS_TIMEX_00026)*

[TPS_TIMEX_00106]{DRAFT} **Purpose of `TimingClockSyncAccuracy.upper`** ⌈The reference `upper` is used to refer to a higher level `TimingClock`, from which a lower level `TimingClock` is derived.⌋*(RS_TIMEX_00026)*

[TPS_TIMEX_00107]{DRAFT} **Purpose of `TimingClockSyncAccuracy.lower`** ⌈The reference `lower` is used to refer a lower level `TimingClock`, which is derived from a higher level `TimingClock`.⌋*(RS_TIMEX_00026)*

[TPS_TIMEX_00108]{DRAFT} **Usage of the attribute `accuracy` of `TimingClockSyncAccuracy`** ⌈The attribute `accuracy` of `TimingClockSyncAccuracy` specifies the synchronization accuracy between the higher level `TDLETZoneClock`, which is referenced by the `upper` attribute and a lower level `TDLETZoneClock`, which is referenced by the `lower` attribute.⌋*(RS_TIMEX_00026)*

[TPS_TIMEX_00109]{DRAFT} **Optionality of `accuracy`** ⌈If the attribute `accuracy` is not specified, it shall be treated as zero.⌋*(RS_TIMEX_00026)*

4.3 Classic Platform Software Clusters

This section describes the support for specifying timing models for *software clusters*. The notion of software cluster and its purpose are described in more detail in [10].

During system design *software clusters* are defined. Software clusters consist of an arbitrary number of software components as sketched in Figure 4.17. The internal structure of those software clusters, like further composition software components and atomic software components, are not known at system design time. Typically, those internals, especially the internal behaviors of the atomic software components along with runnable entities, are the result of further steps taken during the development of the software components. Eventually, the runnable entities need to be assigned to appropriate operating system tasks to ensure the proper execution and therefore a correct provision of the expected functionality realized by the runnable entities. Or, in other words, the operating system being used must provide the needed execution contexts for the runnable entities of a software cluster, if the software cluster is supposed to provide such temporal and dynamic resources.

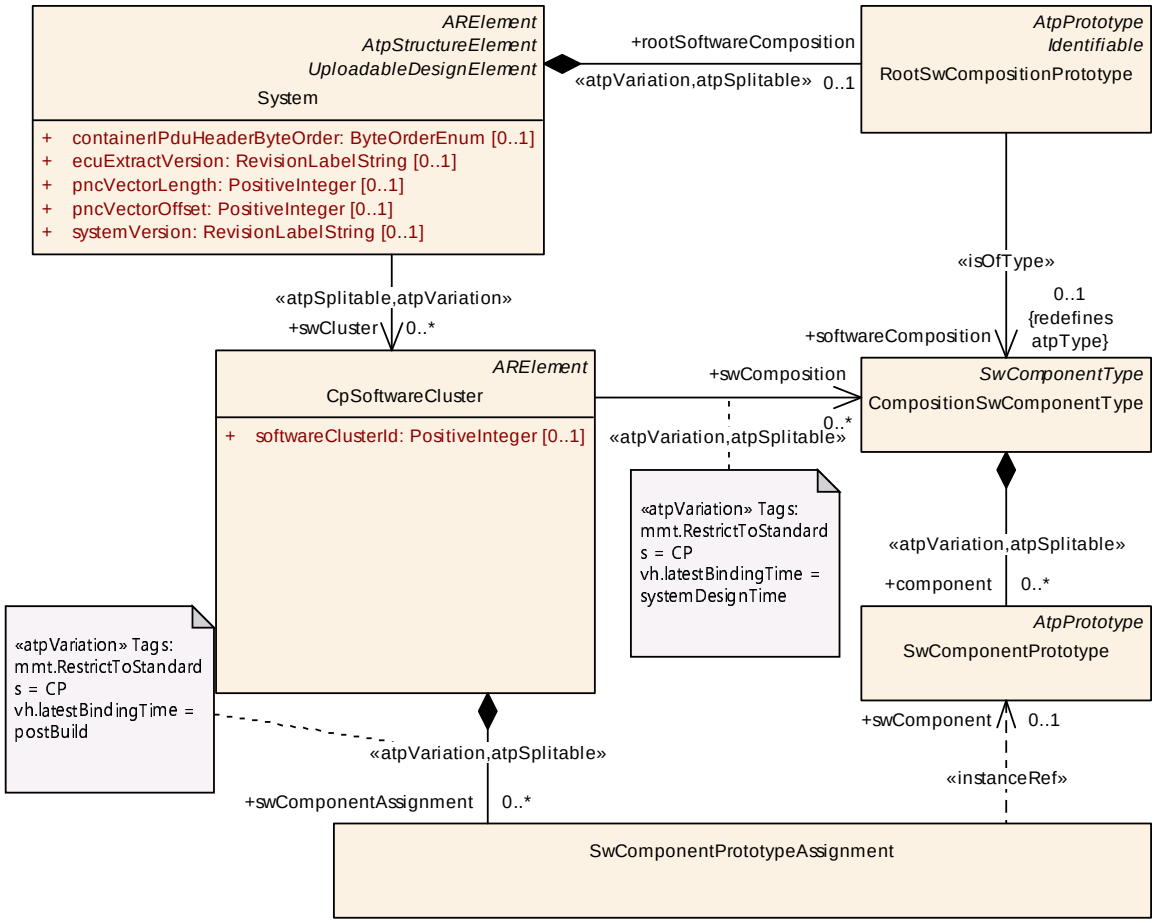


Figure 4.17: Software Cluster Overview

In order to provide a frame at the beginning and during the system design, the elements available in TIMEX are used for describing the temporal and dynamic requirements and characteristics imposed on the execution environment that is needed for proper execution of the software cluster's runnable entities. The notion behind the described approach is that the execution contexts needed by possible runnable entities of a software cluster are abstracted using the elements `TimingDescriptionEvent`,

TimingDescriptionEventChain, and TimingConstraint to specify the temporal parameters of such execution contexts. The execution context is called *dispatcher* as an abstraction for a system event that occurs to activate the execution of a set of runnable entities in the target system.

The way the AUTOSAR elements are utilized is described in the remaining part of this section.

Since software clusters are defined during system design respectively on the AUTOSAR software system level, the system timing view (SystemTiming) is used for creating the corresponding timing models. This is shown in the Figure 4.18.

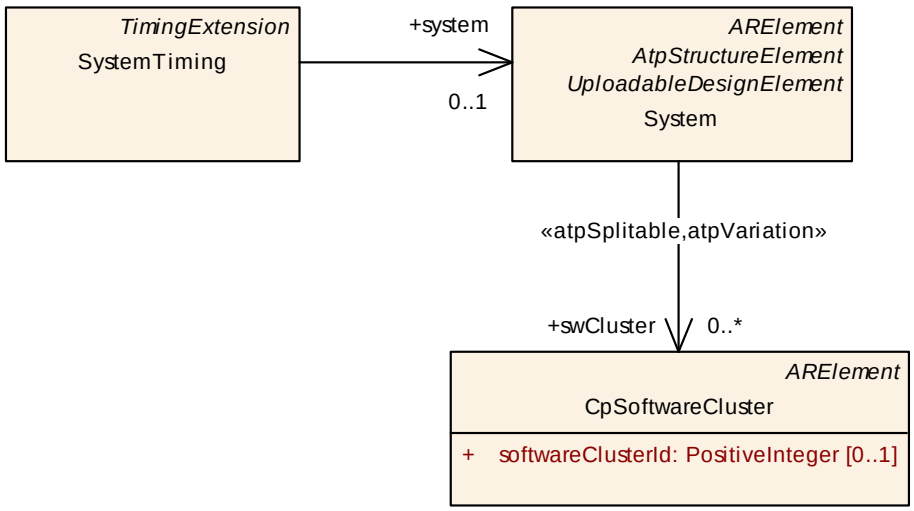


Figure 4.18: System timing used to specify the timing for one or more software clusters

A system timing describing the timing model for a software cluster or a group of software clusters references a system that in turn references one or more software clusters. In the latter case, the scope of the timing model is the set of referenced software clusters. Primarily, the timing model of a system timing for a software cluster contains the definition of the execution context – dispatchers – required for the proper execution of the runnable entities that are later defined in the development.

[TPS_TIMEX_00127]{DRAFT} **SystemTiming of classic platform software clusters** ⌈A SystemTiming which references a System.category== SW_CLUSTER_SYSTEM_DESCRIPTION ([TPS_SYST_01003]) is intended to observe the timing of Classic Platform software clusters.⌋*()*

4.3.1 Specifying a Dispatcher

The value of the attribute category of a TimingDescriptionEvent indicates if the timing description represents the entry point or the exit point point of a dispatcher. The value DISPATCH_ENTRY_POINT indicates the point in time the dispatcher is entered. And the value DISPATCH_EXIT_POINT indicates the point in time the dispatcher is exited.

[constr_4561] Usage of the category value DISPATCH_ENTRY_POINT in TimingDescriptionEvent ⌈The value DISPATCH_ENTRY_POINT of the attribute category of a TimingDescriptionEvent shall be set if and only if the timing description event plays the role of a stimulus event and the corresponding timing description event chain, referencing this timing description event, represents a dispatcher in the context of describing timing of a software cluster.⌋*()*

[constr_4562] Usage of the category value DISPATCH_EXIT_POINT in TimingDescriptionEvent ⌈The value DISPATCH_EXIT_POINT of the attribute category of a TimingDescriptionEvent shall be set if and only if the timing description event plays the role of a response event and the corresponding timing description event chain, referencing this timing description event, represents a dispatcher in the context of describing timing of a software cluster.⌋*()*

As sketched in Figure 4.19 a dispatcher for a software cluster is specified by a timing description event chain TimingDescriptionEventChain. The timing description event chain references two timing description events TimingDescriptionEvents representing the dispatcher's entry point (stimulus event of the timing description event chain) and the dispatcher's exit point (response event of the timing description event chain). The value of the first element's attribute category is DISPATCH_ENTRY_POINT; and the value of the latter element's attribute category is DISPATCH_EXIT_POINT. The complex timing event TDEventComplex is used for the purpose to indicate the event that occurs when the dispatcher is entered and exited. The complex timing event is used because both timing description events are not referencing an *observable* location, like the other types of timing description events. The event occurrence expression of the complex timing event may be used to specify the particular event supposed to occur in case of entering and exiting the dispatcher.

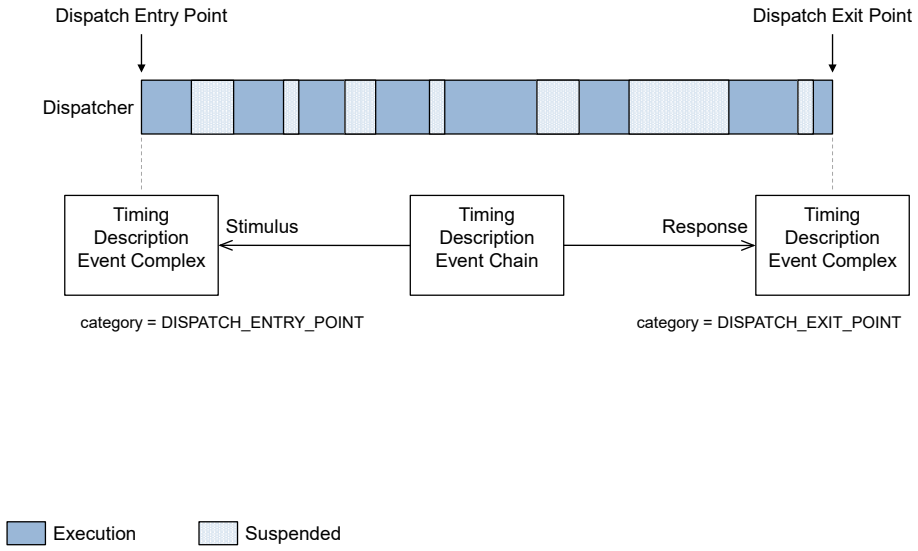


Figure 4.19: Specifying a Dispatcher for a Software Cluster

The listing C.11 shows an ARXML fragment that matches what is shown in Figure 4.19. It specifies a timing description event chain, and two timing description events representing the entry and exit point of this dispatcher. The former timing description event

plays the role *stimulus* and the latter timing description event plays the role *response* of the timing description event chain representing the dispatcher.

The attribute `category` of a `TimingDescriptionEvent` indicates the use of this element in the context of timing models for software clusters respectively dispatchers.

The value of the attribute `category` of a `TimingDescriptionEvent` indicates if the timing description event represents the dispatch entry point of a dispatcher belonging to a software cluster or the dispatch exit point of a dispatcher belonging to a software cluster. The value `DISPATCH_ENTRY_POINT` indicates the point in time the dispatcher is entered respectively activated. And the value `DISPATCH_EXIT_POINT` indicates the point in time the dispatcher is exited respectively terminated. For more details refer to section 3.3.2 and in particular constraints [constr_4559], [constr_4561] and [constr_4562].

4.3.2 Specifying Timing Parameters of a Dispatcher

The timing parameters of a dispatcher are specified in the same fashion as an LET interval and shall be specified according to the explanation in 4.1 particularly the *maximum execution time* (*duration*) of a dispatcher.

A latency timing constraint is imposed on a timing description event chain representing a dispatcher and the attribute `maximum` specifies the maximum execution time of the dispatcher, as shown in Figure 4.20. The value of the attribute `minimum` shall be set to zero (0) because it is not expected that a dispatcher *must* execute for a minimum period of time.

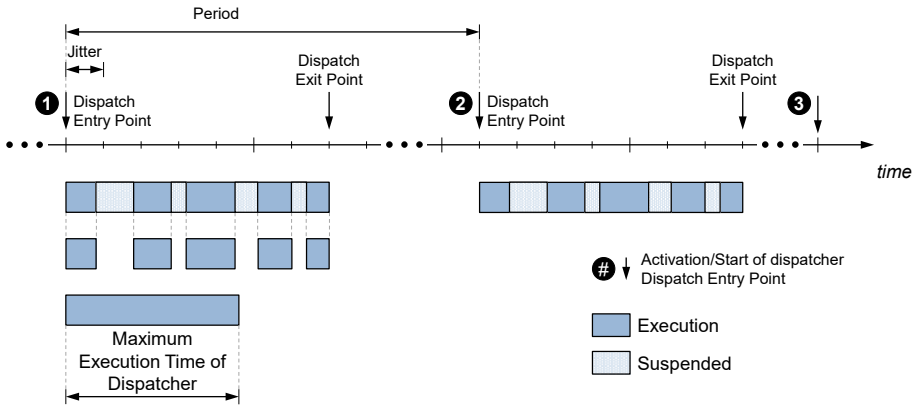


Figure 4.20: Maximum execution time of a dispatcher

As depicted in Figure 4.20 the timing parameters of a dispatcher are specified by a latency timing constraint and periodic event triggering constraint. The latency timing constraint specifies the maximum execution time of a dispatcher within the given period specified by the periodic event triggering constraint.

The listing C.11 shows an ARXML fragment which defines a latency timing constraint of 500μs that is imposed on the timing description event chain — dispatcher — defined in

listing C.11. In essence, this latency timing constraint specifies the maximum execution time of the dispatcher to be 500μs.

The recurrence of a dispatcher is specified using the periodic event triggering constraint `PeriodicEventTriggering`. This timing constraint is imposed on the stimulus event — the entry point of the dispatcher — of the timing description event chain representing a dispatcher. The listing C.11 shows an ARXML fragment which defines a periodic event triggering constraint that specifies a recurrence of 10ms; in other words the dispatcher is entered every 10ms. Note that the execution of a dispatcher is supposed to be completed before the dispatcher is started — entered — again.

4.3.3 Mapping Executable Entities to a Dispatcher

Eventually, the executable entities belonging to the internal behavior of atomic software components within the software clusters shall be assigned to temporal and dynamic resources that are provided by the available *host* software cluster. Since the temporal resource is either a dispatcher or a LET interval, the executable entities need to be assigned to either of them.

In both cases the capabilities of the element `ExecutionOrderConstraint` are used for this purpose. To assign executable entities to LET intervals is described in section 4.1.3 in detail.

An `ExecutionOrderConstraint` is specified consisting of `EOCEventRef`s which reference the RTE events of the executable entities that shall be assigned to the dispatcher. These `EOCEventRef`s are specified as nested elements in an `EOCExecutableEntityRefGroup`. The `triggeringEvent` of the `EOCExecutableEntityRefGroup` references the `TDEventComplex` that plays the role of a dispatch entry point respectively dispatcher.

The listing C.11 shows an ARXML fragment which specifies that the runnable entity one (1) of application software component five (5), runnable entity one (1) of application software component two (2), runnable entity two (2) of application software component 15, runnable entity two (2) of application software component eight (8), runnable entity one (1) of application software component 13, and runnable entity three (3) of application software component nine (9) are executed in the context of the dispatcher called ”Dispatcher10ms”. Those runnable entities are all part of application software components that are part of the application software cluster called ”SoftwareClusterA”.

4.3.4 Mappings between Software Clusters

An important step during the development of AUTOSAR software systems is to specify contracts between various parties that are involved in the development. To specify provided and needed temporal and dynamic resources the *software cluster mappings* are used. The purpose of these mappings are firstly to unequivocally identify a temporal and dynamic resource, and secondly to specify the software cluster providing such

resources and specify the software clusters that request such resources. The Figure 4.21 shows these mapping elements and their relationships.

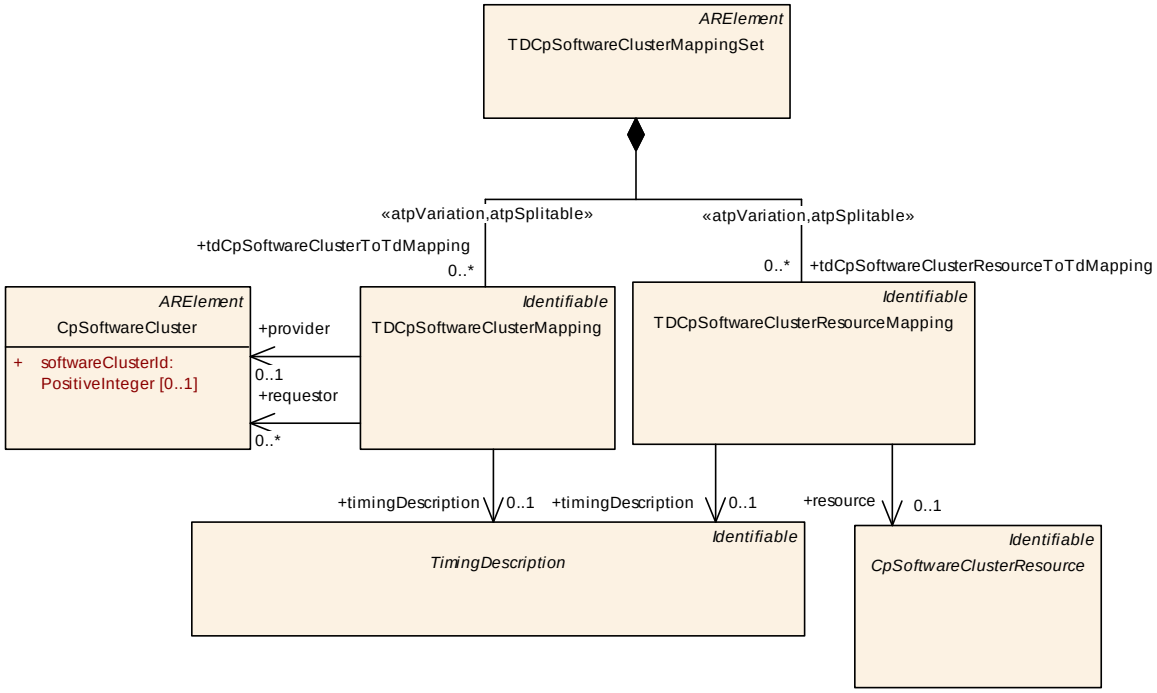


Figure 4.21: TimingCpSoftwareClusterMapping

[TPS_TIMEX_00066] Purpose of `TDCpSoftwareClusterMappingSet` ⌈The element `TDCpSoftwareClusterMappingSet` is used to gather a number of software cluster mappings.⌋ *(RS_TIMEX_00025)*

Class	**TDCpSoftwareClusterMappingSet**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingCpSoftwareCluster			
Note	This is used to gather of classic platform software cluster mappings. **Tags:** atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tdCpSoftware Cluster ResourceToTd Mapping	TDCpSoftwareCluster ResourceMapping	*	aggr	Maps a CP software cluster resource to a temporal resource. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=tdCpSoftwareClusterResourceToTd Mapping.shortName, tdCpSoftwareClusterResourceToTd Mapping.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	TDCpSoftwareClusterMappingSet			
tdCpSoftware ClusterToTd Mapping	TDCpSoftwareCluster Mapping	*	aggr	Maps a temporal resource to a mapping between a providing CP software cluster and requesting CP software clusters. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=tdCpSoftwareClusterToTdMapping.short Name, tdCpSoftwareClusterToTdMapping.variation Point.shortLabel vh.latestBindingTime=postBuild

Table 4.5: TDCpSoftwareClusterMappingSet

[TPS_TIMEX_00067] Purpose of `TDCpSoftwareClusterMapping` ⌈The element `TDCpSoftwareClusterMapping` is used to specify a mapping between a software cluster that provides a temporal and dynamic resources and the software clusters that need this resource.⌋*(RS_TIMEX_00025)*

Class	TDCpSoftwareClusterMapping			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingCpSoftwareCluster			
Note	This is used to specify a mapping between a software cluster that provides temporal and dynamic resources and the software clusters that need these resources.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	TDCpSoftwareClusterMappingSet.tdCpSoftwareClusterToTdMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
provider	CpSoftwareCluster	0..1	ref	This is the software cluster that provides the temporal and dynamic resource.
requestor	CpSoftwareCluster	*	ref	This is the software cluster that requests the temporal and dynamic resource.
timing Description	TimingDescription	0..1	ref	The timing description representing the temporal and dynamic resource.

Table 4.6: TDCpSoftwareClusterMapping

[constr_4563] `TDCpSoftwareClusterMapping` shall reference only dispatchers or LET intervals ⌈The element `TDCpSoftwareClusterMapping` shall reference as timing description either a:

- `TDEventComplex.category==DISPATCH_ENTRY_POINT`, or
- `TimingDescriptionEventChain.category==LET_INTERVAL`.

⌋*()*

[constr_4567] Reference `provider` of `TDCpSoftwareClusterMapping` ⌈The reference `provider` of `TDCpSoftwareClusterMapping` shall refer to a `CpSoftwareCluster` which represents a host software cluster.⌋*()*

[constr_4568] Reference `requestor` of `TDCpSoftwareClusterMapping` ⌈The reference `requestor` of `TDCpSoftwareClusterMapping` shall refer to a `CpSoftwareCluster` which represents an application software cluster.⌋*()*

[TPS_TIMEX_00068] Purpose of `TDCpSoftwareClusterResourceMapping` ⌈The element `TDCpSoftwareClusterResourceMapping` is used to assign an unequivocal global resource identification to a temporal and dynamic resource.⌋*(RS_TIMEX_00025)*

Class	**TDCpSoftwareClusterResourceMapping**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingCpSoftwareCluster			
Note	This is used to assign an unequivocal global resource identification to a temporal and dynamic resource.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	TDCpSoftwareClusterMappingSet.tdCpSoftwareClusterResourceToTdMapping			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
resource	CpSoftwareCluster Resource	0..1	ref	The specific resource identification assigned to the temporal and dynamic resource.
timing Description	TimingDescription	0..1	ref	The timing description representing the temporal and dynamic resource.

Table 4.7: TDCpSoftwareClusterResourceMapping

[constr_4564] `TDCpSoftwareClusterResourceMapping` shall reference only dispatchers or LET intervals ⌈The element `TDCpSoftwareClusterResourceMapping` shall reference as timing description either:

- `TDEventComplex.category==DISPATCH_ENTRY_POINT`, or
- `TimingDescriptionEventChain.category==LET_INTERVAL`.

⌋*()*

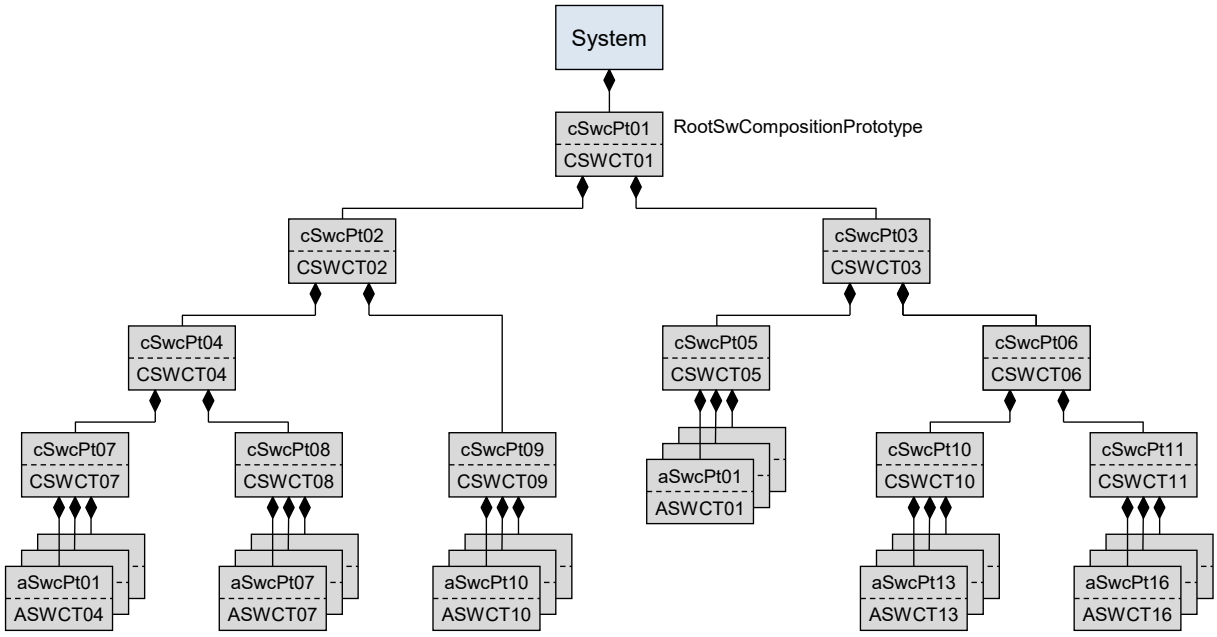
[constr_4565] Consistency of `TDCpSoftwareClusterMapping` and `TDCpSoftwareClusterResourceMapping` ⌈The timing descriptions referenced by the element `TDCpSoftwareClusterMapping` and the element `TDCpSoftwareClusterResourceMapping` shall be consistent.⌋*()*

The listing C.11 shows an ARXML fragment which describes that the *application* software cluster called ”SoftwareClusterA” requires two temporal resources provided by the *host* software cluster called ”SoftwareClusterHost”. The two required temporal resources are the dispatcher called ”Dispatcher10ms” and the logical execution time interval called ”Let5msInterval”.

4.3.5 Example

In the given example, an AUTOSAR software system consists of eleven (11) composition software components. The composition software component *type* called ”CSWCT01” plays the role of the root software composition software prototype of this system. The composition software component type contains two composition software component *prototypes* called ”cSwcPt02” and ”cSwcPt03”. The composition software component type ”CSWCPT02” contains two composition software component

prototypes called ”cSwcPt04” and ”cSwcPt09”; and the composition software component type called ”CSWCT03” contains the composition software component prototypes called ”cSwcPt05 and cSwcPt06”. The composition software component type called ”CSWCT04” contains two composition software component prototypes called ”cSwcPt07” and ”cSwcPt08”. And last but not least, the composition software component type called ”CSWCT06” contains two composition software component prototypes called ”cSwcPt10” and ”cSwcPt11”. The composition hierarchy of the software components is shown in Figure 4.22.



Legend:

cSwcPt# Composition SW Component Prototype
CSWCT# Composition SW Component Type
aSwcPt# Atomic SW Component Prototype
ASWCT# Atomic SW Component Type

Figure 4.22: Example Software Cluster

Each of the composition software component types called ”CSWCT05”, ”CSWCT07”, ”CSWCT08”, ”CSWCT09”, ”CSWCT10” and ”CSWCT11” contain three atomic software component prototypes. The names of these atomic software component prototypes are following the rule ”aSwcPt##”. The placeholder ”##”, representing a two digit number with leading zero, ranges from 01 to 18. For example, the composition software component type ”CSWCT05” contains the three atomic software component prototypes called ”aSwcPt01”, ”aSwcPt02” and ”aSwcPt03”. Any of the atomic software component types have an internal behavior and each of those internal behaviors consists of three runnable entities named ”re#ASWCT##”; for example, the name ”re1ASWCT05” denotes the first runnable entity of the atomic software component type five (5). An RTE Event — Timing Event — is specified for each of the runnable entities, and is named as follows: ”teRe#ASWCT##”.

The described software system consists of two software clusters called ”Software Cluster A” and ”Software Cluster B”. This is shown in the Figure 4.23. The software cluster A consists of the composition software components ”cSwcPt04”, ”cSwcPt05”, ”cSwcPt07”, ”cSwcPt08” and ”cSwcPt10”. And the software cluster B consists of the composition software components ”cSwcPt09” and ”cSwcPt11”.

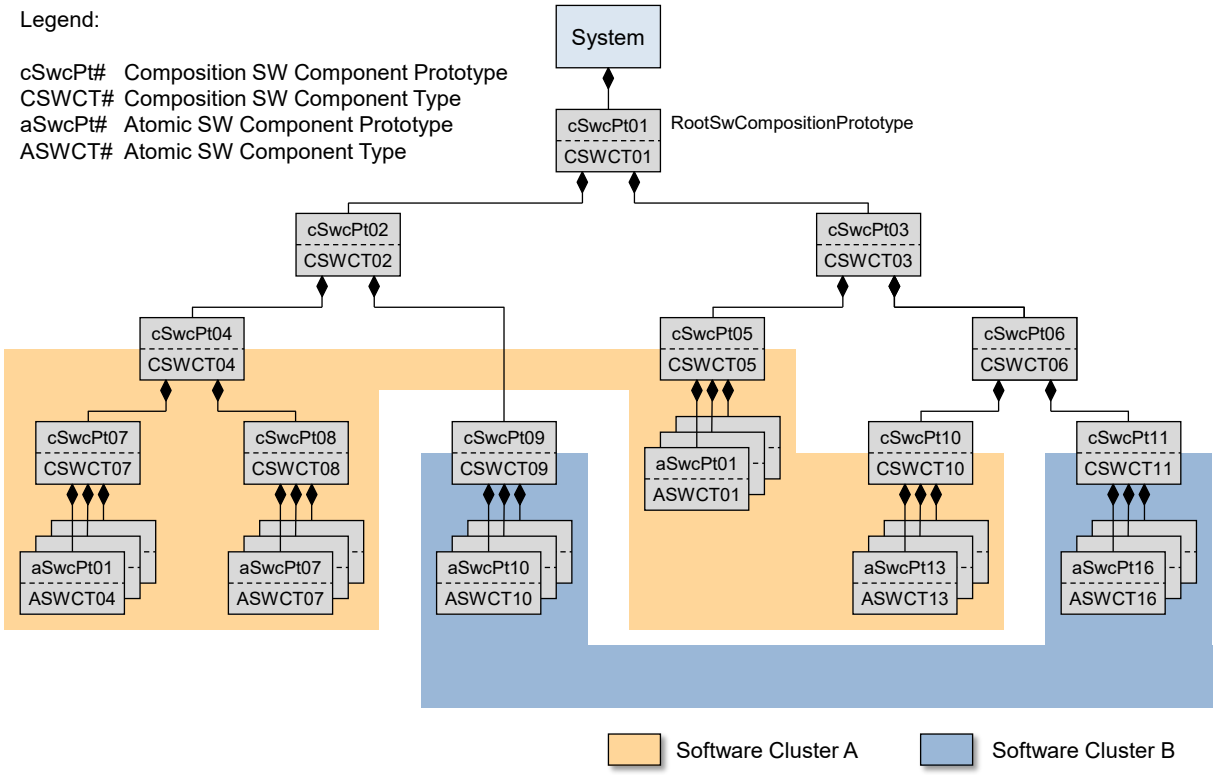


Figure 4.23: System consisting of two Software Clusters

The assumption is that the temporal resources, needed by the software cluster A and software cluster B, are provided by a third software cluster representing the *host* software cluster.

A Reference Material

A.1 Terms and Abbreviations

The main list of terms and abbreviations are defined in [5]. The following table contains the list of terms and abbreviations used in the scope of this document which are not already defined in [5] along with the spelled-out meaning of each of the abbreviations.

Abbreviation	Meaning
EOC	Execution Order Constraint
TD	Timing Description

Table A.1: Abbreviations used in the scope of this Document

Term	Meaning
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 shall occur simultaneously within a certain tolerance interval.
Stimulus	Start point of an event chain.
Timing analysis	Timing analysis is a method of determining the timing behavior of the system. This includes consideration of timing relevant system behavior like task preemptions, interrupt handling, resource blocking, etc.
Timing constraint	A timing constraint may have two different interpretation alternatives. On the one hand, it may define a restriction for the timing behavior of the system (e.g. minimum (maximum) latency bound for a certain event sequence). In this case, a timing constraint is a requirement which the system shall 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 or software component consists of events and event chains. The former one describes events that can be observed and the latter one describe their causal relationship.

△

Term	Meaning
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.
Timing guarantee	See `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. 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 shall 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 See `Timing analysis` with the expected behavior defined by See `Timing constraint`s.

Table A.2: Terms used in the scope of this Document

A.2 Imposition Times of Constraints

The constraints formulated in this document have different *actual* imposition times which denote the steps in the workflow when the respective constraint has to be imposed.

Some imposition times "include" other imposition times, an example for this relation is discussed in the table A.3.

The imposition times that are considered applicable in the scope of this document[1] are listed in Table A.3.

Please note that the imposition times are intentionally rendered as technical terms such that it is possible to link back from each constraint to the definition of the affected imposition time in Table A.3.

[1] Different imposition times may be defined in the context of other AUTOSAR standard documents

Some constraints, however, *may* also be meaningful in the context of other imposition times, applicable for other *AUTOSAR platforms*.

Imposition Time	Description
at the time when the VFB Timing Description is complete	This imposition time is aimed at the time when a VFB Timing is complete.
at the time when the Swc Timing Description is complete	This imposition time is aimed at the time when a Swc Timing is complete.
at the time when the System Timing Description is complete	This imposition time is aimed at the time when a System Timing is complete.
at the time when the Bsw Timing Description is complete	This imposition time is aimed at the time when a Bsw Timing is complete. This applies for both the Bsw Module Timing and the Bsw Composition Timing.
at the time when the Ecu Timing Description is complete	This imposition time is aimed at the time when a Ecu-wide Timing is complete.
at the imposition time associated with the concrete subclass of TimingDescriptionEvent	This means that the imposition time of the constraint cannot be unambiguously defined on the level of the abstract meta-class `TimingDescriptionEvent`. Sub-classes of `TimingDescriptionEvent` have imposition times associated with them (by means of constraints that refer to the subclasses) and the constraints that apply in the context of the definition of `TimingDescriptionEvent` shall therefore not contain a concrete imposition time but take over the imposition time from the applicable subclass. Example: subclass `TDEventVfb` is associated with the imposition time **at the time when the VFB Timing Description is complete**.

▽

△

at the imposition time associated with the concrete subclass of TimingExtension	This means that the imposition time is relative to the concrete subclass of TimingExtension (Timing View) in use, namely: • at the time when the VFB Timing Description is complete • at the time when the Swc Timing Description is complete • at the time when the System Timing Description is complete • at the time when the Bsw Timing Description is complete • at the time when the Ecu Timing Description is complete

Table A.3: Imposition Times considered in the scope of this document

A.3 Requirements Traceability

The following table references the requirements specified in AUTOSAR RS Timing Extensions [11] and denotes how each of them are satisfied by the meta-model.

Requirement	Description	Satisfied by
[RS_TIMEX_00001]	Timing properties	[TPS_TIMEX_00001] [TPS_TIMEX_00002] [TPS_TIMEX_00003] [TPS_TIMEX_00004] [TPS_TIMEX_00005] [TPS_TIMEX_00006] [TPS_TIMEX_00007] [TPS_TIMEX_00008] [TPS_TIMEX_00010] [TPS_TIMEX_00011] [TPS_TIMEX_00012] [TPS_TIMEX_00013] [TPS_TIMEX_00014] [TPS_TIMEX_00015] [TPS_TIMEX_00016] [TPS_TIMEX_00017] [TPS_TIMEX_00018] [TPS_TIMEX_00019] [TPS_TIMEX_00020] [TPS_TIMEX_00021] [TPS_TIMEX_00022] [TPS_TIMEX_00023] [TPS_TIMEX_00024] [TPS_TIMEX_00025] [TPS_TIMEX_00026] [TPS_TIMEX_00027] [TPS_TIMEX_00028] [TPS_TIMEX_00029] [TPS_TIMEX_00030] [TPS_TIMEX_00031] [TPS_TIMEX_00032] [TPS_TIMEX_00033] [TPS_TIMEX_00034] [TPS_TIMEX_00035] [TPS_TIMEX_00036] [TPS_TIMEX_00038] [TPS_TIMEX_00039] [TPS_TIMEX_00041] [TPS_TIMEX_00042] [TPS_TIMEX_00043] [TPS_TIMEX_00044] [TPS_TIMEX_00045] [TPS_TIMEX_00046] [TPS_TIMEX_00047] [TPS_TIMEX_00048] [TPS_TIMEX_00052] [TPS_TIMEX_00053] [TPS_TIMEX_00118] [TPS_TIMEX_00123]
[RS_TIMEX_00002]	Timing constraints	[TPS_TIMEX_00003] [TPS_TIMEX_00004] [TPS_TIMEX_00006] [TPS_TIMEX_00007] [TPS_TIMEX_00010] [TPS_TIMEX_00011] [TPS_TIMEX_00012] [TPS_TIMEX_00013] [TPS_TIMEX_00014] [TPS_TIMEX_00015] [TPS_TIMEX_00038] [TPS_TIMEX_00041] [TPS_TIMEX_00046] [TPS_TIMEX_00047] [TPS_TIMEX_00048]
[RS_TIMEX_00003]	Optionality of timing constraints	[TPS_TIMEX_00009]
[RS_TIMEX_00004]	Event chains	[TPS_TIMEX_00002]

▽

△

Requirement	Description	Satisfied by
[RS_TIMEX_00005]	Structure of event chains	[TPS_TIMEX_00002]
[RS_TIMEX_00006]	Triggering behavior of event chains	[TPS_TIMEX_00003] [TPS_TIMEX_00010] [TPS_TIMEX_00011] [TPS_TIMEX_00012] [TPS_TIMEX_00013] [TPS_TIMEX_00014]
[RS_TIMEX_00007]	Synchronization of event chains	[TPS_TIMEX_00006]
[RS_TIMEX_00008]	Multiple asynchronous time bases	[TPS_TIMEX_00003] [TPS_TIMEX_00006] [TPS_TIMEX_00010] [TPS_TIMEX_00011] [TPS_TIMEX_00012] [TPS_TIMEX_00013] [TPS_TIMEX_00014] [TPS_TIMEX_00015]
[RS_TIMEX_00009]	Loop-back signal flow in sender-receiver communication	[TPS_TIMEX_00002] [TPS_TIMEX_00005]
[RS_TIMEX_00011]	Mode dependency	[TPS_TIMEX_00049] [TPS_TIMEX_00050] [TPS_TIMEX_00051]
[RS_TIMEX_00012]	Sensor/actuator delay	[TPS_TIMEX_00004]
[RS_TIMEX_00013]	Specification of timing resources for software-component description	[TPS_TIMEX_00008]
[RS_TIMEX_00014]	Sequence of execution of runnable entities	[TPS_TIMEX_00007] [TPS_TIMEX_00038] [TPS_TIMEX_00041] [TPS_TIMEX_00046] [TPS_TIMEX_00047] [TPS_TIMEX_00048]
[RS_TIMEX_00015]	Timing-requirements of SW-Components	[TPS_TIMEX_00004] [TPS_TIMEX_00010]
[RS_TIMEX_00017]	Synchronization constraint on events	[TPS_TIMEX_00006]
[RS_TIMEX_00018]	Predefined events for port interfaces at VFB level	[TPS_TIMEX_00039]
[RS_TIMEX_00019]	AUTOSAR Methodology support	[TPS_TIMEX_00020] [TPS_TIMEX_00042] [TPS_TIMEX_00043] [TPS_TIMEX_00044] [TPS_TIMEX_00045]
[RS_TIMEX_00020]	Support for events indicating variable accesses	[TPS_TIMEX_00020] [TPS_TIMEX_00044]
[RS_TIMEX_00022]	Support for Logical Execution Time	[TPS_TIMEX_00055] [TPS_TIMEX_00057] [TPS_TIMEX_00112] [TPS_TIMEX_00113] [TPS_TIMEX_00115] [TPS_TIMEX_00116]
[RS_TIMEX_00023]	Support for Specifying Synchronization	[TPS_TIMEX_00054]
[RS_TIMEX_00025]	Support for Software Cluster	[TPS_TIMEX_00066] [TPS_TIMEX_00067] [TPS_TIMEX_00068]
[RS_TIMEX_00026]	Support for System Level Logical Execution Time	[TPS_TIMEX_00100] [TPS_TIMEX_00101] [TPS_TIMEX_00103] [TPS_TIMEX_00104] [TPS_TIMEX_00105] [TPS_TIMEX_00106] [TPS_TIMEX_00107] [TPS_TIMEX_00108] [TPS_TIMEX_00109] [TPS_TIMEX_00119] [TPS_TIMEX_00125]

Table A.4: RequirementsTracing

B Test Cases

This chapter outlines two test case examples describing a potential approach to use TIMEX in a practical way. Furthermore, chapter B.3 describes the use of external VFB events in more detail.

B.1 Component integration

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

Integrating a software component instance delivered by an external party requires the provision of timing information related to this component. As this information can be attached to specific `SwComponentType`, with regards to its communication partners, the according view `VfbTiming` (see 3.1.1) is used. Additionally, specific timing constraints for implementing this software component are given, too.

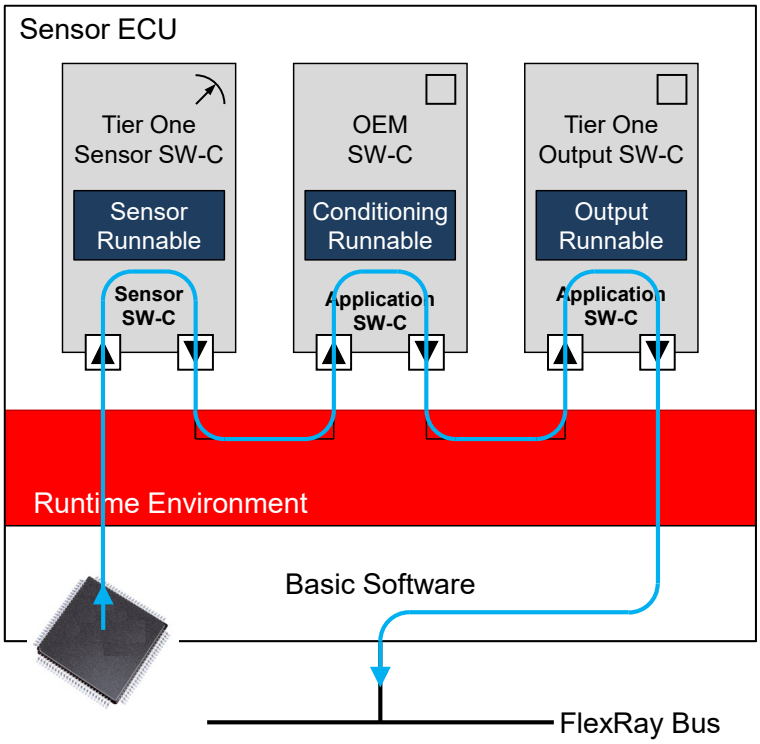


Figure B.1: A Sensor ECU connected with a FlexRay Network and three software components

Figure B.1 outlines the scenario in a demonstrative way. The shown ECU holds three software components: the first one, called ”Sensor SW-C”, reads data from a hardware sensor; the second one, called ”Conditioning SW-C”, performs signal data conditioning like filtering and averaging. And last but not least, the third one, called ‘”Output SW-C”, converts internal data representations (like 32bit Float) to ready-to-send data representations (like UInt16). As certain requirements for sensor data conditioning as an input for several other functions within the vehicle exist, the software component ”Conditioning SW-C” may be delivered by the OEM, directly. A partial description of these components, ports, and interfaces is shown in listing C.16.

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

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

B.1.1 VFB view

At first, timing descriptions and constraints on VFB level are defined. The component ”Conditioning SW-C” receives data via its required port ”UnprocessedSensorData” at a specific point in time. This point is denoted by the event ”ConditioningReceived”, whereas the event ”ConditioningSent” denotes the point in time data is sent via the provided port ”ProcessedSensorData”. To prescribe a ”maximum age” for the reading input the `LatencyTimingConstraint` is used. For this, the *external* event ”SensorDataProduced” is defined. Based on this, an event chain between this external event ”SensorDataProduced” and the ”ConditioningReceived” event is specified — the event ”SensorDataProduced” plays the role of the stimulus event and the event ”ConditioningReceived” is playing the role of the response event with regard to the specified event chain. The latency timing constraint is pointing to this event chain. The representation of the events, event chain and the corresponding timing constraint is shown in listing C.17.

B.1.2 ECU view

After generating the ECU extract, implementation related details of the ECU are available and execution order constraints for the mapped software components — more precise, their runnable entities — exist. For the sake of easiness, each software component implements one `RunnableEntity`. Constraining their execution order using the `ExecutionOrderConstraint` is shown in listing C.18.

Another typical constraint describes the maximum time to be elapsed for sending data on the bus. Therefore, an event ”DataTransmitted” representing the point in time the data is sent on the bus is specified using the event type `TDEventFrame` (listing C.19). Additionally a `TimingDescriptionEventChain` is specified having ”ConditioningSent” as stimulus event and ”DataTransmitted” as response event (see listing C.20).

The constraint prescribing the maximum latency between the point in time the stimulus event occurs and the point in time the response event occurs is shown in listing C.21.

B.2 Engine control

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

B.2.1 Overview

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

Sensors Three `SensorActuatorSwComponentType`s called ”MassAirFlowSensor”, ”AcceleratorPedalSensor”, and ”ThrottleSensor” 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 ”AcceleratorPedalVoter”, ”ThrottleController”, ”ThrottleChange”, ”BaseFuelMass”, ”TransientFuelMass”, ”Ignition”, and ”TotalFuelMass”.

Actuators The control of the actuators is encapsulated by the `SensorActuatorSwComponentType`s ”ThrottleActuator”, ”InjectionActuator”, and ”IgnitionActuator”.

Engine Mode and Control The engine can be operated in different operation modes. The `AtomicSwComponentType` ”OperatingMode” includes a state machine

which decides what setting for the application based calculation is used depending on the current mode, for example normal drive, idle speed, etc. Similar values are delivered by the ”IdleSpeedControl” which determines important inputs for application calculation during idle speed.

Miscellaneous The `AtomicSwComponentType` ”InjBatVoltCorrectionSensor” provides the input from the battery voltage sensor. The `AtomicSwComponentType`”CylNumObserver” is checking whether a change in the cylinder number is sensed and afterwards schedules the application based calculation. In this example application it is assumed that the cylinder number is provided externally within a rate of 2.5ms.

Since giving a complete overview of the system would result in an highly connected graph, Figure B.2 shows a simplified sketch of the `System` because a detailed presentation of such a system would go far beyond the scope of this section. The blue colored lines show important signal paths that are considered to be important for timing analysis and are typically subject to be constrained by timing requirements.

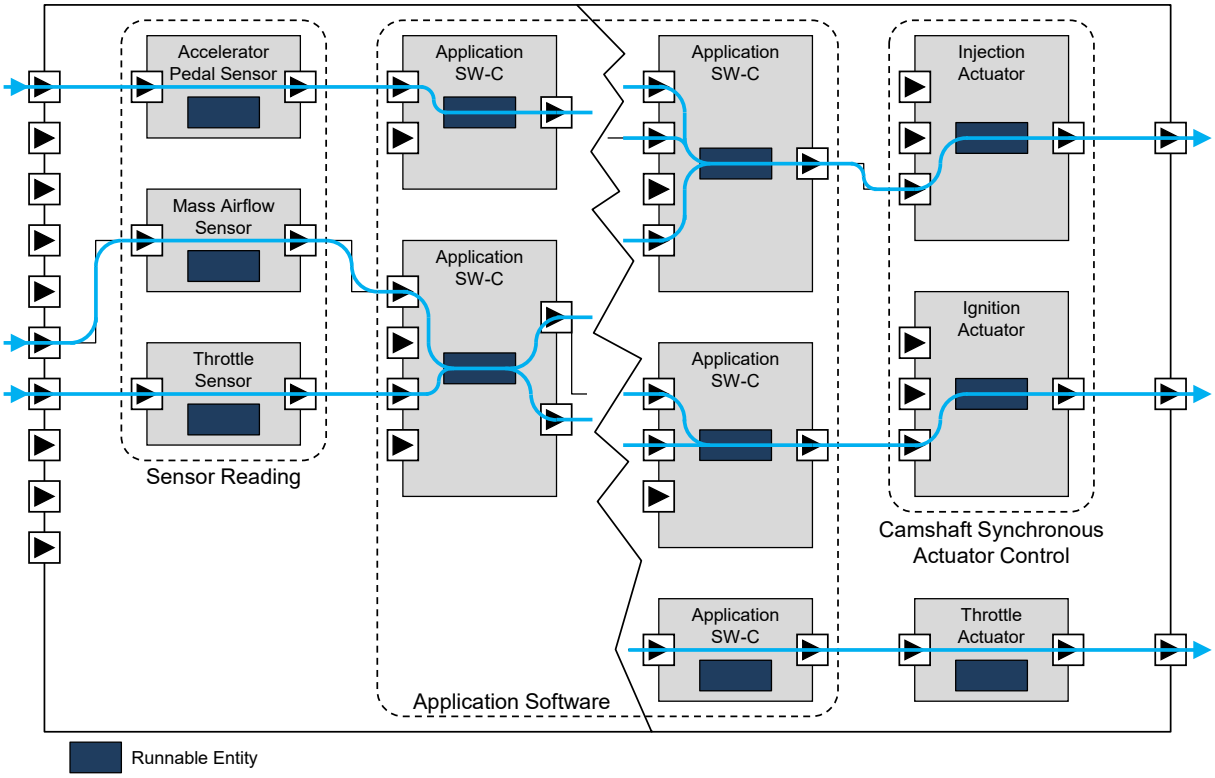


Figure B.2: Rough sketch of an internal combustion engine control system including important signal flow paths.

B.2.2 Timing Requirements

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

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

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

B.2.3 Formal description of timing constraints in VFB View

It should be understood that the requirements from section B.2.2 can be mapped to timing constraints that reference different parts of the system. Since a comprehensive and detailed overview of the whole system would go beyond the scope of this section only the important parts of the system and its timing are given to convey the idea behind using the AUTOSAR Specification of Timing Extensions for each presented timing requirement.

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

B.2.3.1 Requirement 1

Figure B.3 shows the simplified signal flow and involved components. It has been identified that the critical path of execution has an effect on four software components. The sensor software component ”AcceleratorPedal” is responsible for reading in the signal and passes it to the application software component ”AcceleratorPedalVoter”. Afterwards the processed signal is further processed in the application software component ”ThrottleController” until it is finally sent to the actuator via the actuator software component ”ThrottleActuator”. For specification of the timing constraint a timing description event chain shall be defined along with the appropriate timing description events. These timing descriptions and timing constraints are presented in listing C.22 and cover the whole path from the sensor to the actuator.

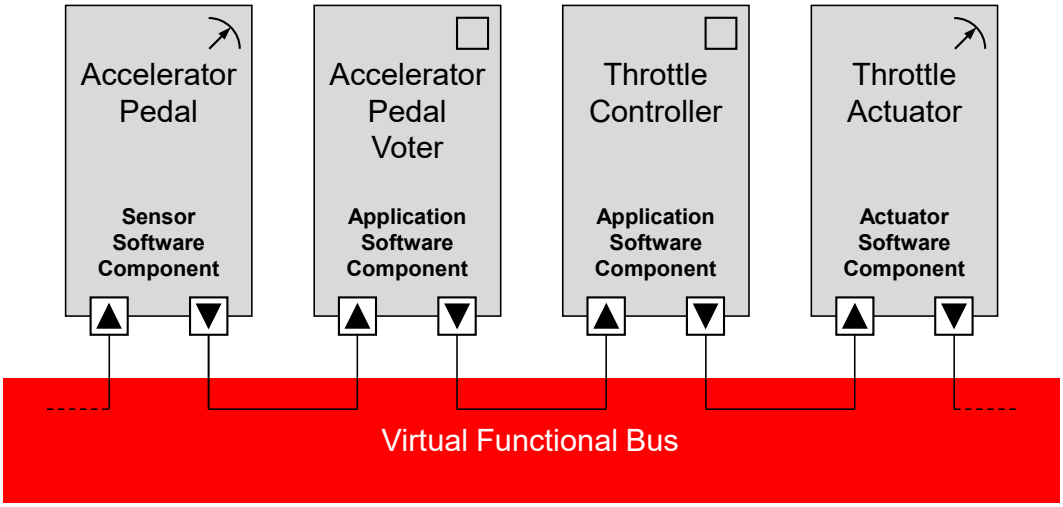


Figure B.3: Involved components for signal flow from ”AcceleratorPedal” to ”Throttle-Actuator” for timing requirement 1.

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

B.2.3.2 Requirement 2

Requirement 2 specifies a typical timing constraint concerning the age of data provided by a sensor. For calculation in the `AtomicSwComponentType` called ”BaseFuelMass” — which is here chosen as an example of the application software — a maximum age of input data concerning the throttle angle shall be guaranteed. The sensor value is determined in the `SensorActuatorSwComponentType` called ”ThrottleSensor” and is passed to the application software. Figure B.4 shows all involved software components.

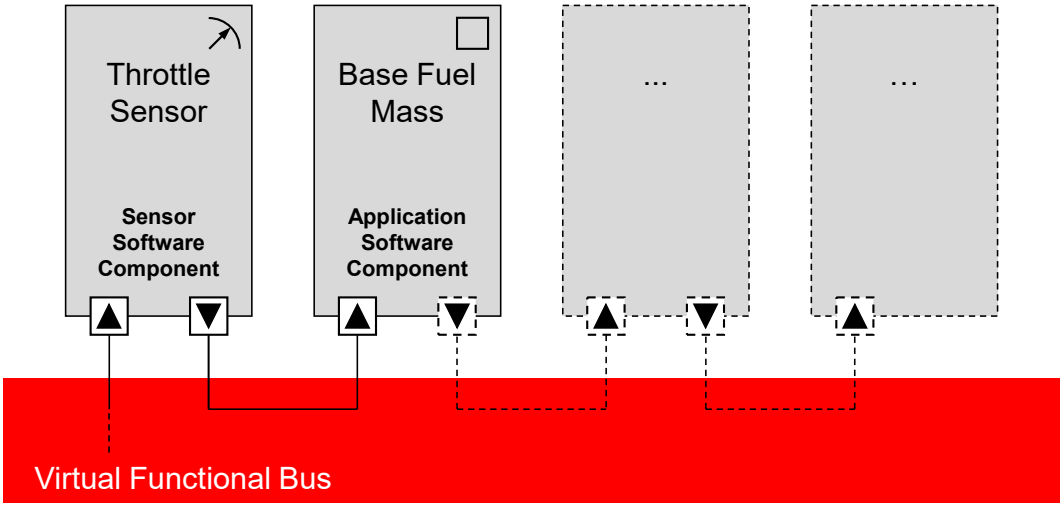


Figure B.4: Involved components for signal flow from ”ThrottleSensor” to application component ”BaseFuelMass” for timing requirement 2.

Please note that even if the signal flow continuous to other parts of the system, it is possible to specify only this aspect of the desired timing behavior as shown in listing C.23.

B.2.3.3 Requirement 3

In requirement 3 a more complex timing description event chain is constrained. The first part of the event chain is already defined in the context of requirement 1. Thus, one can reference the set of defined events as well as the already specified timing description event chains. The second part of the event chain captures the feedback in the system that observes the sensor, in particular ”ThrottleSensor”. Please note that all events shall have a functional dependency, so it is important to understand that the `SensorActuatorSwComponentType` ”ThrottleSensor” shall utilize up-to-date information of the `SensorActuatorSwComponentType` ”ThrottleActuator”. Figure B.5 shows the entire signal path and listing C.24 presents the entire timing information consisting of timing descriptions and timing constraints.

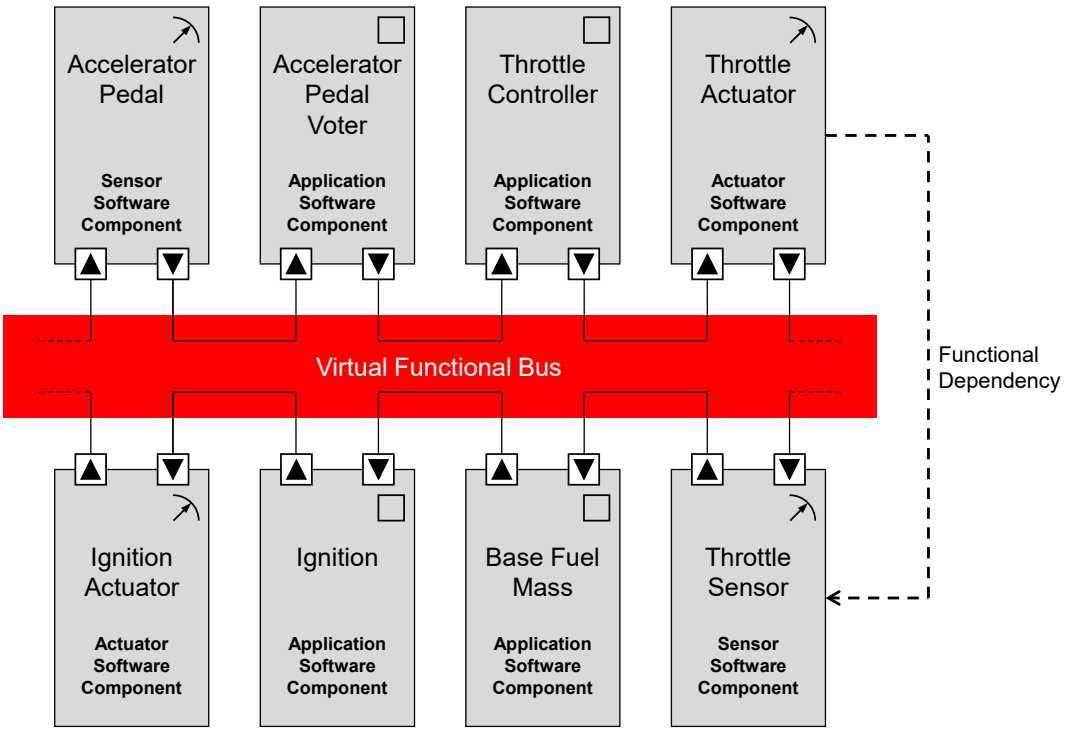


Figure B.5: Involved components for signal flow from sensor software component ”AcceleratorPedal” to actuator software component ”IgnitionActuator” for timing requirement 3.

B.2.4 Formal description of timing constraints in ECU View

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

B.2.4.1 Requirement 4

The stimulus event of the timing description event chain for this requirements is the start of the `BswInterruptEntity` of the basic software module called ”Camshaft”. And as a result the runnable entity of the software component ”IgnitionActuator” is activated. The response event of the timing description event chain is the termination of the `RunnableEntity` belonging to the software component ”IgnitionActuator” as shown in Figure B.6.

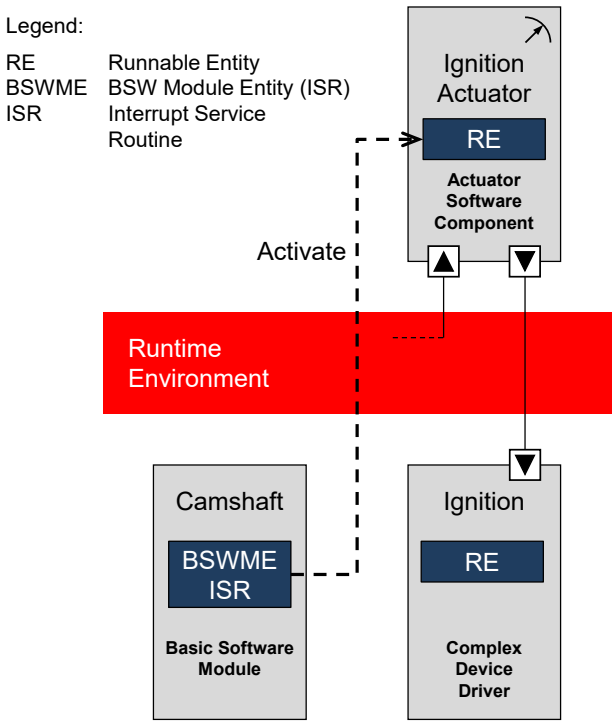


Figure B.6: Involved components and control flow for timing requirement 4.

The timing description events and event chains for this case are presented in listing C.25.

B.2.5 Formal description of timing constraints in SW-C View

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

B.2.5.1 Requirement 5

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

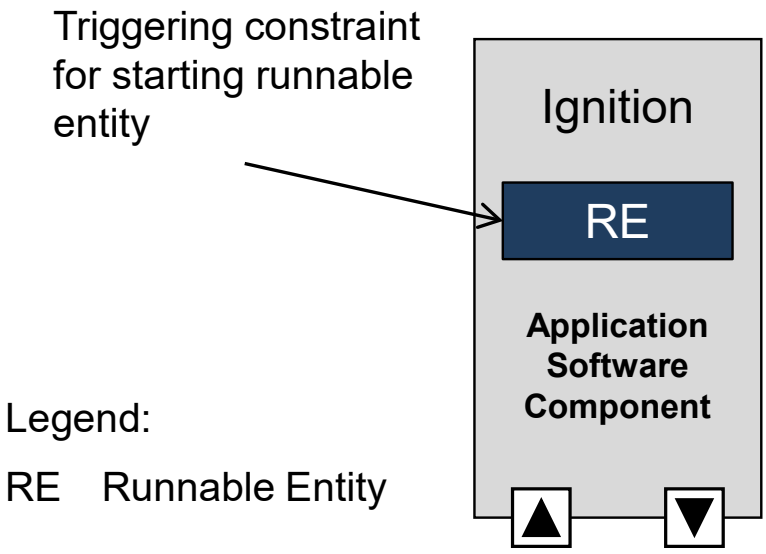


Figure B.7: Involved component and control flow for timing requirement 5.

B.3 Describing and Constraining Sensor and Actuator Timing

Chapter 3.3.2.1 describes the specification of VFB timing description events and introduces the attribute `isExternal` of such events. If the attribute is set to TRUE, then the event is considered to be *external*, which means that the event is supposed to occur on the physical sensor and/or actuator a `SensorActuatorSwComponentType`, a `ComplexDeviceDriverSwComponentType` and `EcuAbstractionSwComponentType` is dealing with. This chapter describes how this attribute is used to describe events for sensor and actuator timing, how the different events of such kind relate to each other in event chains, and how the timing can be constrained using `TimingConstraint`s.

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

B.3.1 External Event of a Sensor accessed via S/R

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

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

B.3.2 External Event of an Actuator accessed via S/R

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

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

B.3.3 External Event of a Sensor accessed via C/S

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

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

B.3.4 External Event of an Actuator accessed via C/S

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

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

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

B.3.5 Considering hardware I/O latency of EventChains at VFB-level

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

B.3.5.1 Input latency

The input latency is defined as the time latency between the point in time where the data is generated by a hardware I/O (e.g. a physical sensor) and the point in time where it is available for the application component, e.g. a SensorActuatorSwComponentType. The input latency is the time between the two events described in B.3.1 and B.3.3, respectively, depending on the communication type.

B.3.5.2 Output latency

The output latency is defined as the time latency between the point in time where the data is sent by the application component, e.g. a SensorActuatorSwComponentType, and the point in time where it is consumed by a hardware I/O (e.g. a physical actuator). The output latency is the time between the two events described in B.3.2 and B.3.4, respectively, depending on the communication type.

B.3.6 Constraining Input or Output Latency

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

C Examples

C.1 Variant Handling

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

For the purpose of specifying time sets the timing extensions utilize the `Variant Handling` capabilities specified and described in [6] chapter "Variant Handling".

As shown in Figure C.1 a SWC called `PlainVanillaSwc` consists of one runnable entity named `RunnableEntityOne`. This runnable entity calculates the value of the variable `DataElementTwo` based on the value of the variable `DataElementOne`.

The latter variable is received via the required port called `rPortOne` and the former one is written to the provided port named `pPortOne`. In the example three alternatives for activating the runnable entity are specified and shown in the table on the right hand side in Figure C.1.

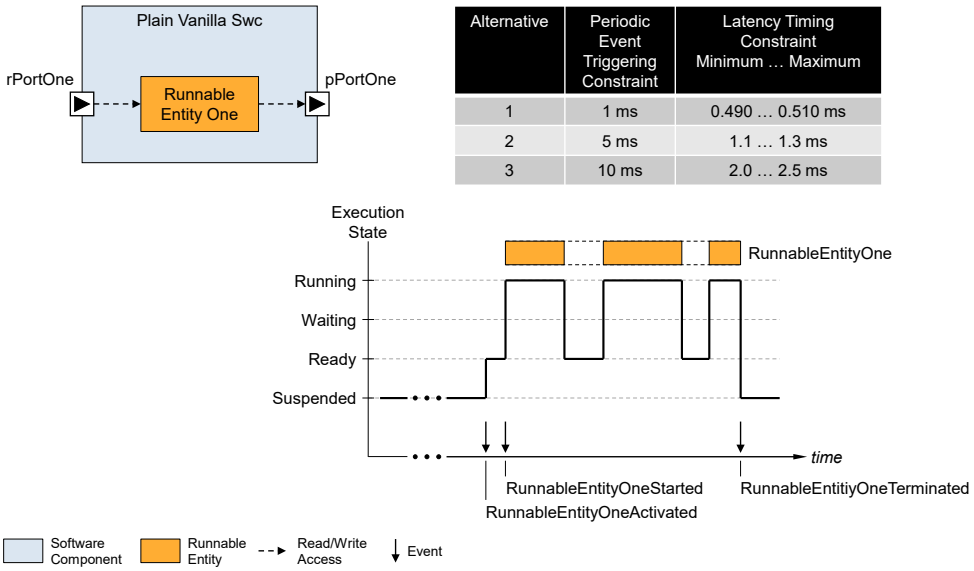


Figure C.1: Example of a set of timings specified for a Runnable Entity.

In listing C.1, the `SwcTiming` called `swctPlainVanillaSwc` specifies the timing descriptions and timing requirements. The timing descriptions describe the events that are related to the runnable entity of a SWC `PlainVanillaSwc`, in particular the activation event `RunnableEntityOneActivated`, start event `RunnableEntityOneStarted` and termination event `RunnableEntityOneTerminated` of the runnable

entity RunnableEntityOne. In addition, a timing description event chain CalculateValueOfDataElementTwo is specified that describes the causal relationship between the events RunnableEntityOneActivated and RunnableEntityOneTerminated.

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

The system constant called ScTimingRunnableEntityOne that eventually is used to select the specific variant of the required timing - the activation frequency and response time of the runnable entity. The selected variant is specified by the system constant's value in the SwSystemconstantValueSet. In the given example the first variant Timing1 is selected.

```
<AR-PACKAGE>
  <SHORT-NAME>TimingExtensions</SHORT-NAME>
  <ELEMENTS>
    <SWC-TIMING>
      <SHORT-NAME>swctPlainVanillaSwc</SHORT-NAME>
      <TIMING-DESCRIPTIONS>
        <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
          <SHORT-NAME>RunnableEntityOneActivated</SHORT-NAME>
          <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/TimeSetExample/PlainVanillaSwc/InternalBehaviorPlainVanillaSwc/
              RunnableEntityOne</RUNNABLE-REF>
          <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-ACTIVATED</TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
        </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
        <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
          <SHORT-NAME>RunnableEntityOneTerminated</SHORT-NAME>
          <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/TimeSetExample/PlainVanillaSwc/InternalBehaviorPlainVanillaSwc/
              RunnableEntityOne</RUNNABLE-REF>
          <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-TERMINATED</TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
        </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
        <TIMING-DESCRIPTION-EVENT-CHAIN>
          <SHORT-NAME>CalculateValueOfDataElementTwo</SHORT-NAME>
          <STIMULUS-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/TimingExtensions/swctPlainVanillaSwc/
              RunnableEntityOneActivated</STIMULUS-REF>
          <RESPONSE-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/TimingExtensions/swctPlainVanillaSwc/
              RunnableEntityOneTerminated</RESPONSE-REF>
          <SEGMENT-REFS>
            <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingExtensions/swctPlainVanillaSwc/
                CalculateValueOfDataElementTwo</SEGMENT-REF>
          </SEGMENT-REFS>
        </TIMING-DESCRIPTION-EVENT-CHAIN>
      </TIMING-DESCRIPTIONS>
      <TIMING-REQUIREMENTS>
        <PERIODIC-EVENT-TRIGGERING>
          <SHORT-NAME>PeriodicActivationRunnableEntityOne</SHORT-NAME>
          <VARIATION-POINT>
            <SHORT-LABEL>VpPetTiming1</SHORT-LABEL>
            <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF DEST="SW-SYSTEMCONST">/SystemConstants/
                ScTimingRunnableEntityOne</SYSC-REF>==1)</SW-SYSCOND>
          </VARIATION-POINT>
          <EVENT-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/TimingExtensions/swctPlainVanillaSwc/
              RunnableEntityOneActivated</EVENT-REF>
          <PERIOD>
            <CSE-CODE>3</CSE-CODE>
            <CSE-CODE-FACTOR>1</CSE-CODE-FACTOR>
          </PERIOD>
        </PERIODIC-EVENT-TRIGGERING>
        <PERIODIC-EVENT-TRIGGERING>
          <SHORT-NAME>PeriodicActivationRunnableEntityOne</SHORT-NAME>
          <VARIATION-POINT>
            <SHORT-LABEL>VpPetTiming2</SHORT-LABEL>
            <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF DEST="SW-SYSTEMCONST">/SystemConstants/
                ScTimingRunnableEntityOne</SYSC-REF>==2)</SW-SYSCOND>
          </VARIATION-POINT>
          <EVENT-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/TimingExtensions/swctPlainVanillaSwc/
              RunnableEntityOneActivated</EVENT-REF>
          <PERIOD>
            <CSE-CODE>3</CSE-CODE>
            <CSE-CODE-FACTOR>5</CSE-CODE-FACTOR>
          </PERIOD>
```

```
        </PERIODIC-EVENT-TRIGGERING>
        <PERIODIC-EVENT-TRIGGERING>
          <SHORT-NAME>PeriodicActivationRunnableEntityOne</SHORT-NAME>
          <VARIATION-POINT>
            <SHORT-LABEL>VpPetTiming3</SHORT-LABEL>
            <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF DEST="SW-SYSTEMCONST">/SystemConstants/
               ScTimingRunnableEntityOne</SYSC-REF>==3)</SW-SYSCOND>
          </VARIATION-POINT>
          <EVENT-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/TimingExtensions/swctPlainVanillaSwc/
             RunnableEntityOneActivated</EVENT-REF>
          <PERIOD>
            <CSE-CODE>3</CSE-CODE>
            <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
          </PERIOD>
        </PERIODIC-EVENT-TRIGGERING>
        <LATENCY-TIMING-CONSTRAINT>
          <SHORT-NAME>ResponseTimeForCalculatingDataElementTwo</SHORT-NAME>
          <VARIATION-POINT>
            <SHORT-LABEL>VpLtcTiming1</SHORT-LABEL>
            <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF DEST="SW-SYSTEMCONST">/SystemConstants/
               ScTimingRunnableEntityOne</SYSC-REF>==1)</SW-SYSCOND>
          </VARIATION-POINT>
          <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
          <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingExtensions/swctPlainVanillaSwc/
             CalculateValueOfDataElementTwo</SCOPE-REF>
          <MINIMUM>
            <CSE-CODE>2</CSE-CODE>
            <CSE-CODE-FACTOR>490</CSE-CODE-FACTOR>
          </MINIMUM>
          <MAXIMUM>
            <CSE-CODE>2</CSE-CODE>
            <CSE-CODE-FACTOR>510</CSE-CODE-FACTOR>
          </MAXIMUM>
        </LATENCY-TIMING-CONSTRAINT>
        <LATENCY-TIMING-CONSTRAINT>
          <SHORT-NAME>ResponseTimeForCalculatingDataElementTwo</SHORT-NAME>
          <VARIATION-POINT>
            <SHORT-LABEL>VpLtcTiming2</SHORT-LABEL>
            <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF DEST="SW-SYSTEMCONST">/SystemConstants/
               ScTimingRunnableEntityOne</SYSC-REF>==2)</SW-SYSCOND>
          </VARIATION-POINT>
          <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
          <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingExtensions/swctPlainVanillaSwc/
             CalculateValueOfDataElementTwo</SCOPE-REF>
          <MINIMUM>
            <CSE-CODE>2</CSE-CODE>
            <CSE-CODE-FACTOR>1100</CSE-CODE-FACTOR>
          </MINIMUM>
          <MAXIMUM>
            <CSE-CODE>2</CSE-CODE>
            <CSE-CODE-FACTOR>1300</CSE-CODE-FACTOR>
          </MAXIMUM>
        </LATENCY-TIMING-CONSTRAINT>
        <LATENCY-TIMING-CONSTRAINT>
          <SHORT-NAME>ResponseTimeForCalculatingDataElementTwo</SHORT-NAME>
          <VARIATION-POINT>
            <SHORT-LABEL>VpLtcTiming3</SHORT-LABEL>
            <SW-SYSCOND BINDING-TIME="SYSTEM-DESIGN-TIME">(<SYSC-REF DEST="SW-SYSTEMCONST">/SystemConstants/
               ScTimingRunnableEntityOne</SYSC-REF>==3)</SW-SYSCOND>
          </VARIATION-POINT>
          <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
          <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingExtensions/swctPlainVanillaSwc/
             CalculateValueOfDataElementTwo</SCOPE-REF>
          <MINIMUM>
            <CSE-CODE>2</CSE-CODE>
            <CSE-CODE-FACTOR>2000</CSE-CODE-FACTOR>
          </MINIMUM>
          <MAXIMUM>
            <CSE-CODE>2</CSE-CODE>
            <CSE-CODE-FACTOR>2500</CSE-CODE-FACTOR>
          </MAXIMUM>
        </LATENCY-TIMING-CONSTRAINT>
      </TIMING-REQUIREMENTS>
      <BEHAVIOR-REF DEST="SWC-INTERNAL-BEHAVIOR">/TimeSetExample/PlainVanillaSwc/InternalBehaviorPlainVanillaSwc</BEHAVIOR
         -REF>
    </SWC-TIMING>
  </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
  <SHORT-NAME>SystemConstants</SHORT-NAME>
  <ELEMENTS>
    <SW-SYSTEMCONST>
      <SHORT-NAME>ScTimingRunnableEntityOne</SHORT-NAME>
    </SW-SYSTEMCONST>
  </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
  <SHORT-NAME>Variants</SHORT-NAME>
  <ELEMENTS>
    <SW-SYSTEMCONSTANT-VALUE-SET>
      <SHORT-NAME>TIMEX_EXP_Time_Set_VariantValues</SHORT-NAME>
      <SW-SYSTEMCONSTANT-VALUES>
        <SW-SYSTEMCONST-VALUE>
  <!-- The values of the system constant ScTimingRunnableEntityOne may range from 1 to 3. -->
          <SW-SYSTEMCONST-REF DEST="SW-SYSTEMCONST">/SystemConstants/ScTimingRunnableEntityOne</SW-SYSTEMCONST-REF>
```

```
          <VALUE>1</VALUE>
        </SW-SYSTEMCONST-VALUE>
      </SW-SYSTEMCONSTANT-VALUES>
    </SW-SYSTEMCONSTANT-VALUE-SET>
    <POST-BUILD-VARIANT-CRITERION-VALUE-SET>
      <SHORT-NAME>TIMEX_EXP_Time_Set_VariantPostBuildValues</SHORT-NAME>
    </POST-BUILD-VARIANT-CRITERION-VALUE-SET>
    <PREDEFINED-VARIANT>
      <SHORT-NAME>TIMEX_EXP_Time_Set_Variant</SHORT-NAME>
      <POST-BUILD-VARIANT-CRITERION-VALUE-SET-REFS>
        <POST-BUILD-VARIANT-CRITERION-VALUE-SET-REF DEST="POST-BUILD-VARIANT-CRITERION-VALUE-SET">/Variants/
          TIMEX_EXP_Time_Set_VariantPostBuildValues</POST-BUILD-VARIANT-CRITERION-VALUE-SET-REF>
      </POST-BUILD-VARIANT-CRITERION-VALUE-SET-REFS>
      <SW-SYSTEMCONSTANT-VALUE-SET-REFS>
        <SW-SYSTEMCONSTANT-VALUE-SET-REF DEST="SW-SYSTEMCONSTANT-VALUE-SET">/Variants/TIMEX_EXP_Time_Set_VariantValues</SW
          -SYSTEMCONSTANT-VALUE-SET-REF>
      </SW-SYSTEMCONSTANT-VALUE-SET-REFS>
    </PREDEFINED-VARIANT>
  </ELEMENTS>
</AR-PACKAGE>
```

Listing C.1: ARXML for Figure C.1

C.2 Logical Execution Time

C.2.1 Logical Execution Time - Timing descriptions

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

Listing C.2: Specifying LET Interval

C.2.2 Logical Execution Time - Timing constraints - Latency Timing

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

Listing C.3: Specifying LET Interval Timing

C.2.3 Logical Execution Time - Timing constraints - Event Triggering

```
<PERIODIC-EVENT-TRIGGERING>
  <SHORT-NAME>petcLetIntervalRecurrence</SHORT-NAME>
  <EVENT-REF DEST="TD-EVENT-COMPLEX">/LogicalExecutionTime/TimingExtensions/LogicalExecutionTime02/
    LetIntervalRelease</EVENT-REF>
  <MINIMUM-INTER-ARRIVAL-TIME>
```

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

Listing C.4: Specifying Recurrence of a LET Interval

C.2.4 Logical Execution Time - Timing constraints - Event Triggering - Gap

```
<LATENCY-TIMING-CONSTRAINT>
  <SHORT-NAME>ltcLetInterval_1</SHORT-NAME>
  <CATEGORY>LET_INTERVAL</CATEGORY>
  <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
  <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/SpecifyingPeriodicLetIntervalWithGap/
    sysTiming_SpecifyingPeriodicLetIntervalWithGap/tdecLetInterval_1</SCOPE-REF>
  <MINIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>6</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>6</CSE-CODE-FACTOR>
  </MAXIMUM>
</LATENCY-TIMING-CONSTRAINT>
<PERIODIC-EVENT-TRIGGERING>
  <SHORT-NAME>petcLetInterval_1</SHORT-NAME>
  <MINIMUM-INTER-ARRIVAL-TIME>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
  </MINIMUM-INTER-ARRIVAL-TIME>
  <JITTER>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
  </JITTER>
  <PERIOD>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
  </PERIOD>
</PERIODIC-EVENT-TRIGGERING>
```

Listing C.5: Periodic LET Interval with Gap

C.2.5 Logical Execution Time - Timing constraints - Offset Timing - Without

```
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_2</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/SpecifyingPeriodicLetIntervalsWithNoOffset/
    sysTiming_SpecifyingPeriodicLetIntervalsWithNoOffset/LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/SpecifyingPeriodicLetIntervalsWithNoOffset/
    sysTiming_SpecifyingPeriodicLetIntervalsWithNoOffset/LetIntervalRelease_2</TARGET-REF>
  <MINIMUM>
    <CSE-CODE>2</CSE-CODE>
    <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>2</CSE-CODE>
    <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
  </MAXIMUM>
</OFFSET-TIMING-CONSTRAINT>
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_3</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/SpecifyingPeriodicLetIntervalsWithNoOffset/
    sysTiming_SpecifyingPeriodicLetIntervalsWithNoOffset/LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/SpecifyingPeriodicLetIntervalsWithNoOffset/
    sysTiming_SpecifyingPeriodicLetIntervalsWithNoOffset/LetIntervalRelease_3</TARGET-REF>
  <MINIMUM>
    <CSE-CODE>2</CSE-CODE>
    <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>2</CSE-CODE>
    <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
  </MAXIMUM>
</OFFSET-TIMING-CONSTRAINT>
```

Listing C.6: Periodic LET Intervals with no Offset

C.2.6 Logical Execution Time - Timing constraints - Offset Timing - With

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

Listing C.7: Periodic LET Intervals with Offset

C.2.7 Logical Execution Time - Timing constraints - Offset Timing - Arbitrary

```
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_2</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervals/sysTiming_SpecifyingArbitraryLetIntervals/
     LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervals/sysTiming_SpecifyingArbitraryLetIntervals/
     LetIntervalRelease_2</TARGET-REF>
  <MINIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>2</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>2</CSE-CODE-FACTOR>
  </MAXIMUM>
</OFFSET-TIMING-CONSTRAINT>
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_3</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervals/sysTiming_SpecifyingArbitraryLetIntervals/
     LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervals/sysTiming_SpecifyingArbitraryLetIntervals/
     LetIntervalRelease_3</TARGET-REF>
  <MINIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>8</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>8</CSE-CODE-FACTOR>
  </MAXIMUM>
</OFFSET-TIMING-CONSTRAINT>
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_4</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervals/sysTiming_SpecifyingArbitraryLetIntervals/
     LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervals/sysTiming_SpecifyingArbitraryLetIntervals/
     LetIntervalRelease_4</TARGET-REF>
  <MINIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>9</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>9</CSE-CODE-FACTOR>
  </MAXIMUM>
</OFFSET-TIMING-CONSTRAINT>
```

Listing C.8: Arbitrary LET Intervals

C.2.8 Logical Execution Time - Timing constraints - Offset Timing - Arbitrary Overlap

```
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_2</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervalsWithOverlap/
     sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervalsWithOverlap/
     sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/LetIntervalRelease_2</TARGET-REF>
  <MINIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>13</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>13</CSE-CODE-FACTOR>
  </MAXIMUM>
</OFFSET-TIMING-CONSTRAINT>
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_3</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervalsWithOverlap/
     sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervalsWithOverlap/
     sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/LetIntervalRelease_3</TARGET-REF>
  <MINIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>4</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>4</CSE-CODE-FACTOR>
  </MAXIMUM>
</OFFSET-TIMING-CONSTRAINT>
<OFFSET-TIMING-CONSTRAINT>
  <SHORT-NAME>otc_LetInterval_4</SHORT-NAME>
  <SOURCE-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervalsWithOverlap/
     sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/LetIntervalRelease_1</SOURCE-REF>
  <TARGET-REF DEST="TD-EVENT-COMPLEX">/SpecifyingArbitraryLetIntervalsWithOverlap/
     sysTiming_SpecifyingArbitraryLetIntervalsWithOverlap/LetIntervalRelease_4</TARGET-REF>
  <MINIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>3</CSE-CODE-FACTOR>
  </MINIMUM>
  <MAXIMUM>
    <CSE-CODE>3</CSE-CODE>
    <CSE-CODE-FACTOR>3</CSE-CODE-FACTOR>
  </MAXIMUM>
</OFFSET-TIMING-CONSTRAINT>
```

Listing C.9: Arbitrary LET Intervals with Overlap

C.2.9 Logical Execution Time - Timing constraints - Execution Order

```
<EOC-EXECUTABLE-ENTITY-REF-GROUP>
  <SHORT-NAME>eocEerg_ExecutableEntitiesCluster_01</SHORT-NAME>
   <LET-INTERVAL-REFS>
    <LET-INTERVAL-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/MappingEecToLetInterval/
       sysTiming_MappingEecToLetInterval/tdecLetInterval_1</LET-INTERVAL-REF>
  </LET-INTERVAL-REFS>
  <NESTED-ELEMENT-REFS>
    <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/MappingEecToLetInterval/sysTiming_MappingEecToLetInterval/
       eoc_DefiningAndMappingEecs/eocERef_te_re01_Swc01_1ms</NESTED-ELEMENT-REF>
    <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/MappingEecToLetInterval/sysTiming_MappingEecToLetInterval/
       eoc_DefiningAndMappingEecs/eocERef_te_re01_Swc02_1ms</NESTED-ELEMENT-REF>
    <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/MappingEecToLetInterval/sysTiming_MappingEecToLetInterval/
       eoc_DefiningAndMappingEecs/eocERef_te_re01_Swc04_1ms</NESTED-ELEMENT-REF>
  </NESTED-ELEMENT-REFS>
  <SUCCESSOR-REFS>
    <SUCCESSOR-REF DEST="EOC-EXECUTABLE-ENTITY-REF-GROUP">/MappingEecToLetInterval/
       sysTiming_MappingEecToLetInterval/eoc_DefiningAndMappingEecs/eocEerg_ExecutableEntitiesCluster_02</
       SUCCESSOR-REF>
    <SUCCESSOR-REF DEST="EOC-EXECUTABLE-ENTITY-REF-GROUP">/MappingEecToLetInterval/
       sysTiming_MappingEecToLetInterval/eoc_DefiningAndMappingEecs/eocEerg_ExecutableEntitiesCluster_03</
       SUCCESSOR-REF>
  </SUCCESSOR-REFS>
```

Listing C.10: Mapping Executable Entities Cluster to LET Intervals

C.3 Software Cluster

C.3.1 Software Cluster - Timing descriptions

```
<?xml version="1.0" encoding="UTF-8"?>
<AUTOSAR xmlns="http://autosar.org/schema/r4.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
   http://autosar.org/schema/r4.0 AUTOSAR_00052.xsd">
  <AR-PACKAGES>
    <AR-PACKAGE UUID="bdbade53-5890-4e1c-aac0-91d3e71b2e19">
      <SHORT-NAME>System</SHORT-NAME>
      <ELEMENTS>
        <SYSTEM S="">
          <SHORT-NAME>SoftwareClusterA</SHORT-NAME>
          <CATEGORY>SW_CLUSTER_SYSTEM_DESCRIPTION</CATEGORY>
          <SW-CLUSTERS>
            <CP-SOFTWARE-CLUSTER-REF-CONDITIONAL>
              <CP-SOFTWARE-CLUSTER-REF DEST="CP-SOFTWARE-CLUSTER">/SoftwareClusters/SoftwareClusterA</CP-SOFTWARE-CLUSTER-REF>
            </CP-SOFTWARE-CLUSTER-REF-CONDITIONAL>
          </SW-CLUSTERS>
        </SYSTEM>
        <SYSTEM S="">
          <SHORT-NAME>SoftwareClusterB</SHORT-NAME>
          <CATEGORY>SW_CLUSTER_SYSTEM_DESCRIPTION</CATEGORY>
          <SW-CLUSTERS>
            <CP-SOFTWARE-CLUSTER-REF-CONDITIONAL>
              <CP-SOFTWARE-CLUSTER-REF DEST="CP-SOFTWARE-CLUSTER">/SoftwareClusters/SoftwareClusterB</CP-SOFTWARE-CLUSTER-REF>
            </CP-SOFTWARE-CLUSTER-REF-CONDITIONAL>
          </SW-CLUSTERS>
        </SYSTEM>
        <SYSTEM S="">
          <SHORT-NAME>SoftwareClusterHost</SHORT-NAME>
          <CATEGORY>SW_CLUSTER_SYSTEM_DESCRIPTION</CATEGORY>
          <SW-CLUSTERS>
            <CP-SOFTWARE-CLUSTER-REF-CONDITIONAL>
              <CP-SOFTWARE-CLUSTER-REF DEST="CP-SOFTWARE-CLUSTER">/SoftwareClusters/SoftwareClusterHost</CP-SOFTWARE-CLUSTER-
                  REF>
            </CP-SOFTWARE-CLUSTER-REF-CONDITIONAL>
          </SW-CLUSTERS>
        </SYSTEM>
        <SYSTEM>
          <SHORT-NAME>System</SHORT-NAME>
          <CATEGORY>SYSTEM_DESCRIPTION</CATEGORY>
          <ROOT-SOFTWARE-COMPOSITIONS>
            <ROOT-SW-COMPOSITION-PROTOTYPE>
              <SHORT-NAME>rscp_System</SHORT-NAME>
              <SOFTWARE-COMPOSITION-TREF DEST="COMPOSITION-SW-COMPONENT-TYPE">/SwComponents/CSWCT01</SOFTWARE-COMPOSITION-TREF
                  >
            </ROOT-SW-COMPOSITION-PROTOTYPE>
          </ROOT-SOFTWARE-COMPOSITIONS>
        </SYSTEM>
      </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
      <SHORT-NAME>SoftwareClusters</SHORT-NAME>
      <ELEMENTS>
        <CP-SOFTWARE-CLUSTER>
          <SHORT-NAME>SoftwareClusterA</SHORT-NAME>
          <CATEGORY>APPLICATION_SOFTWARE_CLUSTER</CATEGORY>
          <SW-COMPOSITIONS>
            <COMPOSITION-SW-COMPONENT-TYPE-REF-CONDITIONAL>
              <COMPOSITION-SW-COMPONENT-TYPE-REF DEST="COMPOSITION-SW-COMPONENT-TYPE">/SwComponents/CSWCT04</COMPOSITION-SW-
                  COMPONENT-TYPE-REF>
            </COMPOSITION-SW-COMPONENT-TYPE-REF-CONDITIONAL>
            <COMPOSITION-SW-COMPONENT-TYPE-REF-CONDITIONAL>
              <COMPOSITION-SW-COMPONENT-TYPE-REF DEST="COMPOSITION-SW-COMPONENT-TYPE">/SwComponents/CSWCT05</COMPOSITION-SW-
                  COMPONENT-TYPE-REF>
            </COMPOSITION-SW-COMPONENT-TYPE-REF-CONDITIONAL>
            <COMPOSITION-SW-COMPONENT-TYPE-REF-CONDITIONAL>
              <COMPOSITION-SW-COMPONENT-TYPE-REF DEST="COMPOSITION-SW-COMPONENT-TYPE">/SwComponents/CSWCT10</COMPOSITION-SW-
                  COMPONENT-TYPE-REF>
            </COMPOSITION-SW-COMPONENT-TYPE-REF-CONDITIONAL>
          </SW-COMPOSITIONS>
        </CP-SOFTWARE-CLUSTER>
        <CP-SOFTWARE-CLUSTER>
          <SHORT-NAME>SoftwareClusterB</SHORT-NAME>
          <CATEGORY>APPLICATION_SOFTWARE_CLUSTER</CATEGORY>
          <SW-COMPOSITIONS>
            <COMPOSITION-SW-COMPONENT-TYPE-REF-CONDITIONAL>
              <COMPOSITION-SW-COMPONENT-TYPE-REF DEST="COMPOSITION-SW-COMPONENT-TYPE">/SwComponents/CSWCT09</COMPOSITION-SW-
                  COMPONENT-TYPE-REF>
            </COMPOSITION-SW-COMPONENT-TYPE-REF-CONDITIONAL>
            <COMPOSITION-SW-COMPONENT-TYPE-REF-CONDITIONAL>
              <COMPOSITION-SW-COMPONENT-TYPE-REF DEST="COMPOSITION-SW-COMPONENT-TYPE">/SwComponents/CSWCT11</COMPOSITION-SW-
                  COMPONENT-TYPE-REF>
            </COMPOSITION-SW-COMPONENT-TYPE-REF-CONDITIONAL>
          </SW-COMPOSITIONS>
        </CP-SOFTWARE-CLUSTER>
        <CP-SOFTWARE-CLUSTER>
          <SHORT-NAME>SoftwareClusterHost</SHORT-NAME>
          <CATEGORY>HOST_SOFTWARE_CLUSTER</CATEGORY>
          <SW-COMPOSITIONS/>
        </CP-SOFTWARE-CLUSTER>
      </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
      <SHORT-NAME>SoftwareClusterMappings</SHORT-NAME>
      <AR-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>AtoHost</SHORT-NAME>
```

```
        <ELEMENTS>
          <TD-CP-SOFTWARE-CLUSTER-MAPPING-SET>
            <SHORT-NAME>AtoHostMappingSet</SHORT-NAME>
            <TD-CP-SOFTWARE-CLUSTER-TO-TD-MAPPINGS>
              <TD-CP-SOFTWARE-CLUSTER-MAPPING>
                <SHORT-NAME>AToDispatcher10msMapping</SHORT-NAME>
                <PROVIDER-REF DEST="CP-SOFTWARE-CLUSTER">/SoftwareClusters/SoftwareClusterHost</PROVIDER-REF>
                <REQUESTOR-REFS>
                  <REQUESTOR-REF DEST="CP-SOFTWARE-CLUSTER">/SoftwareClusters/SoftwareClusterA</REQUESTOR-REF>
                </REQUESTOR-REFS>
                <TIMING-DESCRIPTION-REF DEST="TD-EVENT-COMPLEX">/TimingExtensions/SoftwareClusterHostTiming/
                    Dispatcher10msEntry</TIMING-DESCRIPTION-REF>
              </TD-CP-SOFTWARE-CLUSTER-MAPPING>
              <TD-CP-SOFTWARE-CLUSTER-MAPPING>
                <SHORT-NAME>AToLet5msMapping</SHORT-NAME>
                <PROVIDER-REF DEST="CP-SOFTWARE-CLUSTER">/SoftwareClusters/SoftwareClusterHost</PROVIDER-REF>
                <REQUESTOR-REFS>
                  <REQUESTOR-REF DEST="CP-SOFTWARE-CLUSTER">/SoftwareClusters/SoftwareClusterA</REQUESTOR-REF>
                </REQUESTOR-REFS>
                <TIMING-DESCRIPTION-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingExtensions/SoftwareClusterHostTiming/
                    Let5msInterval</TIMING-DESCRIPTION-REF>
              </TD-CP-SOFTWARE-CLUSTER-MAPPING>
            </TD-CP-SOFTWARE-CLUSTER-TO-TD-MAPPINGS>
          </TD-CP-SOFTWARE-CLUSTER-MAPPING-SET>
        </ELEMENTS>
      </AR-PACKAGE>
      <AR-PACKAGE>
        <SHORT-NAME>BtoHost</SHORT-NAME>
      </AR-PACKAGE>
    </AR-PACKAGES>
  </AR-PACKAGE>
  <AR-PACKAGE UUID="0fddb17c-de73-447d-a0a6-85b85f5d2449">
    <SHORT-NAME>TimingExtensions</SHORT-NAME>
    <ELEMENTS>
      <SYSTEM-TIMING>
        <SHORT-NAME>SoftwareClusterATiming</SHORT-NAME>
        <TIMING-REQUIREMENTS>
          <EXECUTION-ORDER-CONSTRAINT>
            <SHORT-NAME>eocDispatcher10ms</SHORT-NAME>
            <IGNORE-ORDER-ALLOWED>false</IGNORE-ORDER-ALLOWED>
            <IS-EVENT>true</IS-EVENT>
            <ORDERED-ELEMENTS>
              <EOC-EXECUTABLE-ENTITY-REF-GROUP>
                <SHORT-NAME>eoceergDispatcher10ms</SHORT-NAME>
                <NESTED-ELEMENT-REFS>
                  <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/TimingExtensions/SoftwareClusterATiming/eocDispatcher10ms/
                      eocerRe1ASWCT05</NESTED-ELEMENT-REF>
                  <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/TimingExtensions/SoftwareClusterATiming/eocDispatcher10ms/
                      eocerRe1ASWCT02</NESTED-ELEMENT-REF>
                  <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/TimingExtensions/SoftwareClusterATiming/eocDispatcher10ms/
                      eocerRe2ASWCT15</NESTED-ELEMENT-REF>
                  <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/TimingExtensions/SoftwareClusterATiming/eocDispatcher10ms/
                      eocerRe2ASWCT08</NESTED-ELEMENT-REF>
                  <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/TimingExtensions/SoftwareClusterATiming/eocDispatcher10ms/
                      eocerRe1ASWCT13</NESTED-ELEMENT-REF>
                  <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/TimingExtensions/SoftwareClusterATiming/eocDispatcher10ms/
                      eocerRe3ASWCT09</NESTED-ELEMENT-REF>
                </NESTED-ELEMENT-REFS>
                <TRIGGERING-EVENT-REF DEST="TD-EVENT-COMPLEX">/TimingExtensions/SoftwareClusterHostTiming/
                    Dispatcher10msEntry</TRIGGERING-EVENT-REF>
              </EOC-EXECUTABLE-ENTITY-REF-GROUP>
              <EOC-EVENT-REF>
                <SHORT-NAME>eocerRe1ASWCT13</SHORT-NAME>
                <EVENT-REF DEST="TIMING-EVENT">/SwComponents/ASWCT13/ibASWCT13/teRe1ASWCT13</EVENT-REF>
              </EOC-EVENT-REF>
              <EOC-EVENT-REF>
                <SHORT-NAME>eocerRe1ASWCT02</SHORT-NAME>
                <EVENT-REF DEST="TIMING-EVENT">/SwComponents/ASWCT02/ibASWCT02/teRe1ASWCT02</EVENT-REF>
              </EOC-EVENT-REF>
              <EOC-EVENT-REF>
                <SHORT-NAME>eocerRe1ASWCT05</SHORT-NAME>
                <EVENT-REF DEST="TIMING-EVENT">/SwComponents/ASWCT05/ibASWCT05/teRe1ASWCT05</EVENT-REF>
              </EOC-EVENT-REF>
              <EOC-EVENT-REF>
                <SHORT-NAME>eocerRe2ASWCT08</SHORT-NAME>
                <EVENT-REF DEST="TIMING-EVENT">/SwComponents/ASWCT08/ibASWCT08/teRe2ASWCT08</EVENT-REF>
              </EOC-EVENT-REF>
              <EOC-EVENT-REF>
                <SHORT-NAME>eocerRe2ASWCT15</SHORT-NAME>
                <EVENT-REF DEST="TIMING-EVENT">/SwComponents/ASWCT15/ibASWCT15/teRe2ASWCT15</EVENT-REF>
              </EOC-EVENT-REF>
              <EOC-EVENT-REF>
                <SHORT-NAME>eocerRe3ASWCT09</SHORT-NAME>
                <EVENT-REF DEST="TIMING-EVENT">/SwComponents/ASWCT09/ibASWCT09/teRe3ASWCT09</EVENT-REF>
              </EOC-EVENT-REF>
            </ORDERED-ELEMENTS>
          </EXECUTION-ORDER-CONSTRAINT>
          <EXECUTION-ORDER-CONSTRAINT>
            <SHORT-NAME>eocLet5ms</SHORT-NAME>
            <IGNORE-ORDER-ALLOWED>false</IGNORE-ORDER-ALLOWED>
            <IS-EVENT>true</IS-EVENT>
            <ORDERED-ELEMENTS>
              <EOC-EXECUTABLE-ENTITY-REF-GROUP>
                <SHORT-NAME>eoceergLet5ms</SHORT-NAME>
                <LET-INTERVAL-REFS>
```

```
            <LET-INTERVAL-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingExtensions/SoftwareClusterHostTiming/
                Let5msInterval</LET-INTERVAL-REF>
            </LET-INTERVAL-REFS>
            <NESTED-ELEMENT-REFS>
              <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/TimingExtensions/SoftwareClusterATiming/eocLet5ms/
                  eocerRe1ASWCT14</NESTED-ELEMENT-REF>
              <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/TimingExtensions/SoftwareClusterATiming/eocLet5ms/
                  eocerRe2ASWCT05</NESTED-ELEMENT-REF>
              <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/TimingExtensions/SoftwareClusterATiming/eocLet5ms/
                  eocerRe2ASWCT03</NESTED-ELEMENT-REF>
              <NESTED-ELEMENT-REF DEST="EOC-EVENT-REF">/TimingExtensions/SoftwareClusterATiming/eocLet5ms/
                  eocerRe3ASWCT06</NESTED-ELEMENT-REF>
            </NESTED-ELEMENT-REFS>
          </EOC-EXECUTABLE-ENTITY-REF-GROUP>
          <EOC-EVENT-REF>
            <SHORT-NAME>eocerRe1ASWCT14</SHORT-NAME>
            <EVENT-REF DEST="TIMING-EVENT">/SwComponents/ASWCT14/ibASWCT14/teRe1ASWCT14</EVENT-REF>
          </EOC-EVENT-REF>
          <EOC-EVENT-REF>
            <SHORT-NAME>eocerRe2ASWCT03</SHORT-NAME>
            <EVENT-REF DEST="TIMING-EVENT">/SwComponents/ASWCT03/ibASWCT03/teRe2ASWCT03</EVENT-REF>
          </EOC-EVENT-REF>
          <EOC-EVENT-REF>
            <SHORT-NAME>eocerRe2ASWCT05</SHORT-NAME>
            <EVENT-REF DEST="TIMING-EVENT">/SwComponents/ASWCT05/ibASWCT05/teRe2ASWCT05</EVENT-REF>
          </EOC-EVENT-REF>
          <EOC-EVENT-REF>
            <SHORT-NAME>eocerRe3ASWCT06</SHORT-NAME>
            <EVENT-REF DEST="TIMING-EVENT">/SwComponents/ASWCT06/ibASWCT06/teRe3ASWCT06</EVENT-REF>
          </EOC-EVENT-REF>
        </ORDERED-ELEMENTS>
        <PERMIT-MULTIPLE-REFERENCES-TO-EE>false</PERMIT-MULTIPLE-REFERENCES-TO-EE>
      </EXECUTION-ORDER-CONSTRAINT>
    </TIMING-REQUIREMENTS>
    <SYSTEM-REF DEST="SYSTEM">/System/SoftwareClusterA</SYSTEM-REF>
  </SYSTEM-TIMING>
  <SYSTEM-TIMING S="">
    <SHORT-NAME>SoftwareClusterBTiming</SHORT-NAME>
    <SYSTEM-REF DEST="SYSTEM">/System/SoftwareClusterB</SYSTEM-REF>
  </SYSTEM-TIMING>
  <SYSTEM-TIMING UUID="">
    <SHORT-NAME>SoftwareClusterHostTiming</SHORT-NAME>
    <TIMING-DESCRIPTIONS>
      <TIMING-DESCRIPTION-EVENT-CHAIN>
        <SHORT-NAME>Dispatcher10ms</SHORT-NAME>
        <STIMULUS-REF DEST="TD-EVENT-COMPLEX">/TimingExtensions/SoftwareClusterHostTiming/Dispatcher10msEntry</STIMULUS-
            REF>
        <RESPONSE-REF DEST="TD-EVENT-COMPLEX">/TimingExtensions/SoftwareClusterHostTiming/Dispatcher10msExit</RESPONSE-
            REF>
        <SEGMENT-REFS>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingExtensions/SoftwareClusterHostTiming/Dispatcher10ms<
              /SEGMENT-REF>
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
      <TD-EVENT-COMPLEX>
        <SHORT-NAME>Dispatcher10msEntry</SHORT-NAME>
        <CATEGORY>DISPATCH_ENTRY_POINT</CATEGORY>
      </TD-EVENT-COMPLEX>
      <TD-EVENT-COMPLEX>
        <SHORT-NAME>Dispatcher10msExit</SHORT-NAME>
        <CATEGORY>DISPATCH_EXIT_POINT</CATEGORY>
      </TD-EVENT-COMPLEX>
      <TIMING-DESCRIPTION-EVENT-CHAIN>
        <SHORT-NAME>Let5msInterval</SHORT-NAME>
        <CATEGORY>LET_INTERVAL</CATEGORY>
        <STIMULUS-REF DEST="TD-EVENT-COMPLEX">/TimingExtensions/SoftwareClusterHostTiming/Let5msRelease</STIMULUS-REF>
        <RESPONSE-REF DEST="TD-EVENT-COMPLEX">/TimingExtensions/SoftwareClusterHostTiming/Let5msTerminate</RESPONSE-REF>
        <SEGMENT-REFS>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingExtensions/SoftwareClusterHostTiming/Let5msInterval<
              /SEGMENT-REF>
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
      <TD-EVENT-COMPLEX>
        <SHORT-NAME>Let5msRelease</SHORT-NAME>
        <CATEGORY>LET_RELEASE</CATEGORY>
      </TD-EVENT-COMPLEX>
      <TD-EVENT-COMPLEX>
        <SHORT-NAME>Let5msTerminate</SHORT-NAME>
        <CATEGORY>LET_TERMINATE</CATEGORY>
      </TD-EVENT-COMPLEX>
    </TIMING-DESCRIPTIONS>
    <TIMING-REQUIREMENTS>
      <LATENCY-TIMING-CONSTRAINT>
        <SHORT-NAME>Dispatcher10msMaxExecutionTime</SHORT-NAME>
        <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
        <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingExtensions/SoftwareClusterHostTiming/Dispatcher10ms</
            SCOPE-REF>
        <MINIMUM>
          <CSE-CODE>2</CSE-CODE>
          <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
        </MINIMUM>
        <MAXIMUM>
          <CSE-CODE>2</CSE-CODE>
          <CSE-CODE-FACTOR>5</CSE-CODE-FACTOR>
        </MAXIMUM>
```

```
            </LATENCY-TIMING-CONSTRAINT>
            <PERIODIC-EVENT-TRIGGERING>
              <SHORT-NAME>Dispatcher10msPetc</SHORT-NAME>
              <EVENT-REF DEST="TD-EVENT-COMPLEX">/TimingExtensions/SoftwareClusterHostTiming/Dispatcher10msEntry</EVENT-REF>
              <MINIMUM-INTER-ARRIVAL-TIME>
                <CSE-CODE>3</CSE-CODE>
                <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
              </MINIMUM-INTER-ARRIVAL-TIME>
              <JITTER>
                <CSE-CODE>3</CSE-CODE>
                <CSE-CODE-FACTOR>1</CSE-CODE-FACTOR>
              </JITTER>
              <PERIOD>
                <CSE-CODE>3</CSE-CODE>
                <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
              </PERIOD>
            </PERIODIC-EVENT-TRIGGERING>
            <LATENCY-TIMING-CONSTRAINT>
              <SHORT-NAME>Let5msLatency</SHORT-NAME>
              <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
              <MINIMUM>
                <CSE-CODE>3</CSE-CODE>
                <CSE-CODE-FACTOR>5</CSE-CODE-FACTOR>
              </MINIMUM>
              <MAXIMUM>
                <CSE-CODE>3</CSE-CODE>
                <CSE-CODE-FACTOR>5</CSE-CODE-FACTOR>
              </MAXIMUM>
            </LATENCY-TIMING-CONSTRAINT>
            <PERIODIC-EVENT-TRIGGERING>
              <SHORT-NAME>Let5msPetc</SHORT-NAME>
              <EVENT-REF DEST="TD-EVENT-COMPLEX">/TimingExtensions/SoftwareClusterHostTiming/Let5msRelease</EVENT-REF>
              <MINIMUM-INTER-ARRIVAL-TIME>
                <CSE-CODE>3</CSE-CODE>
                <CSE-CODE-FACTOR>5</CSE-CODE-FACTOR>
              </MINIMUM-INTER-ARRIVAL-TIME>
              <JITTER>
                <CSE-CODE>3</CSE-CODE>
                <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
              </JITTER>
              <PERIOD>
                <CSE-CODE>3</CSE-CODE>
                <CSE-CODE-FACTOR>5</CSE-CODE-FACTOR>
              </PERIOD>
            </PERIODIC-EVENT-TRIGGERING>
          </TIMING-REQUIREMENTS>
          <SYSTEM-REF DEST="SYSTEM">/System/SoftwareClusterHost</SYSTEM-REF>
        </SYSTEM-TIMING>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AUTOSAR>
```

Listing C.11: Timing Extensions - Software Cluster Dispatcher Logical Execution Time

C.4 System Level Logical Execution Time

C.4.1 System Level Logical Execution Time

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Purpose:    Example ARXML listings for use in TPS_AbstractPlatformDescription
    Date:       First version 2020-August
    Author:     B.R.
-->
<AUTOSAR xmlns="http://autosar.org/schema/r4.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
   http://autosar.org/schema/r4.0␣AUTOSAR_00052.xsd">
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>SystemLevelLogicalExecutionTime</SHORT-NAME>
      <AR-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>Interfaces</SHORT-NAME>
          <ELEMENTS>
            <SENDER-RECEIVER-INTERFACE>
              <SHORT-NAME>Interface1</SHORT-NAME>
              <DATA-ELEMENTS>
                <VARIABLE-DATA-PROTOTYPE>
                  <SHORT-NAME>DataElement1</SHORT-NAME>
                </VARIABLE-DATA-PROTOTYPE>
              </DATA-ELEMENTS>
            </SENDER-RECEIVER-INTERFACE>
          </ELEMENTS>
        </AR-PACKAGE>
        <AR-PACKAGE>
          <SHORT-NAME>Components</SHORT-NAME>
          <ELEMENTS>
            <APPLICATION-SW-COMPONENT-TYPE>
              <SHORT-NAME>Component1</SHORT-NAME>
```

```
        <PORTS>
          <P-PORT-PROTOTYPE>
            <SHORT-NAME>PPort1</SHORT-NAME>
            <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/SystemLevelLogicalExecutionTime/Interfaces/
              Interface1</PROVIDED-INTERFACE-TREF>
          </P-PORT-PROTOTYPE>
          <R-PORT-PROTOTYPE>
            <SHORT-NAME>RPort1</SHORT-NAME>
            <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/SystemLevelLogicalExecutionTime/Interfaces/
              Interface1</REQUIRED-INTERFACE-TREF>
          </R-PORT-PROTOTYPE>
        </PORTS>
      </APPLICATION-SW-COMPONENT-TYPE>
    </ELEMENTS>
  </AR-PACKAGE>
  <AR-PACKAGE>
    <SHORT-NAME>TimingExtensions</SHORT-NAME>
    <ELEMENTS>
      <SYSTEM-TIMING>
        <SHORT-NAME>SystemTiming</SHORT-NAME>
        <TIMING-CLOCKS>
          <TDLET-ZONE-CLOCK>
            <SHORT-NAME>ZoneClock0</SHORT-NAME>
            <ACCURACY-INT>
              <CSE-CODE>0</CSE-CODE>
              <CSE-CODE-FACTOR>30</CSE-CODE-FACTOR>
            </ACCURACY-INT>
          </TDLET-ZONE-CLOCK>
          <TDLET-ZONE-CLOCK>
            <SHORT-NAME>ZoneClock1</SHORT-NAME>
          </TDLET-ZONE-CLOCK>
          <TDLET-ZONE-CLOCK>
            <SHORT-NAME>ZoneClock2</SHORT-NAME>
          </TDLET-ZONE-CLOCK>
        </TIMING-CLOCKS>
        <TIMING-CLOCK-SYNC-ACCURACYS>
          <TIMING-CLOCK-SYNC-ACCURACY>
            <SHORT-NAME>SyncAccuracy1</SHORT-NAME>
            <ACCURACY>
              <CSE-CODE>0</CSE-CODE>
              <CSE-CODE-FACTOR>10</CSE-CODE-FACTOR>
            </ACCURACY>
            <LOWER-REF DEST="TDLET-ZONE-CLOCK">/SystemLevelLogicalExecutionTime/TimingExtensions/SystemTiming/ZoneClock1
              </LOWER-REF>
            <UPPER-REF DEST="TDLET-ZONE-CLOCK">/SystemLevelLogicalExecutionTime/TimingExtensions/SystemTiming/ZoneClock0
              </UPPER-REF>
          </TIMING-CLOCK-SYNC-ACCURACY>
          <TIMING-CLOCK-SYNC-ACCURACY>
            <SHORT-NAME>SyncAccuracy2</SHORT-NAME>
            <ACCURACY>
              <CSE-CODE>0</CSE-CODE>
              <CSE-CODE-FACTOR>5</CSE-CODE-FACTOR>
            </ACCURACY>
            <LOWER-REF DEST="TDLET-ZONE-CLOCK">/SystemLevelLogicalExecutionTime/TimingExtensions/SystemTiming/ZoneClock2
              </LOWER-REF>
            <UPPER-REF DEST="TDLET-ZONE-CLOCK">/SystemLevelLogicalExecutionTime/TimingExtensions/SystemTiming/ZoneClock0
              </UPPER-REF>
          </TIMING-CLOCK-SYNC-ACCURACY>
        </TIMING-CLOCK-SYNC-ACCURACYS>
        <!-- start example sllet tds - line 48 -->
        <TIMING-DESCRIPTIONS>
          <TIMING-DESCRIPTION-EVENT-CHAIN>
            <SHORT-NAME>TimingDescriptionChainSlLet1</SHORT-NAME>
            <CATEGORY>SL_LET_INTERVAL</CATEGORY>
            <STIMULUS-REF DEST="TD-EVENT-SLLET">/SystemLevelLogicalExecutionTime/TimingExtensions/SystemTiming/
              RPort1_Release</STIMULUS-REF>
            <RESPONSE-REF DEST="TD-EVENT-SLLET">/SystemLevelLogicalExecutionTime/TimingExtensions/SystemTiming/
              PPort1_Terminate</RESPONSE-REF>
            <SEGMENT-REFS>
              <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/SystemLevelLogicalExecutionTime/TimingExtensions/
                SystemTiming/TimingDescriptionChainSlLet1</SEGMENT-REF>
            </SEGMENT-REFS>
          </TIMING-DESCRIPTION-EVENT-CHAIN>
          <TD-EVENT-SLLET-PORT>
            <SHORT-NAME>RPort1_Release</SHORT-NAME>
            <CATEGORY>SL_LET_RELEASE</CATEGORY>
            <CLOCK-REFERENCE-REF DEST="TDLET-ZONE-CLOCK">/SystemLevelLogicalExecutionTime/TimingExtensions/SystemTiming/
              ZoneClock0</CLOCK-REFERENCE-REF>
            <PORT-REF DEST="R-PORT-PROTOTYPE">/SystemLevelLogicalExecutionTime/Components/Component1/RPort1</PORT-REF>
          </TD-EVENT-SLLET-PORT>
          <TD-EVENT-SLLET-PORT>
            <SHORT-NAME>PPort1_Terminate</SHORT-NAME>
            <CATEGORY>SL_LET_TERMINATE</CATEGORY>
            <CLOCK-REFERENCE-REF DEST="TDLET-ZONE-CLOCK">/SystemLevelLogicalExecutionTime/TimingExtensions/SystemTiming/
              ZoneClock1</CLOCK-REFERENCE-REF>
            <PORT-REF DEST="P-PORT-PROTOTYPE">/SystemLevelLogicalExecutionTime/Components/Component1/PPort1</PORT-REF>
          </TD-EVENT-SLLET-PORT>
        </TIMING-DESCRIPTIONS>
        <!-- end example sllet tds - line 72 -->
        <!-- start example sllet duration/recurrence - line 73 -->
        <TIMING-REQUIREMENTS>
          <PERIODIC-EVENT-TRIGGERING>
            <SHORT-NAME>PeriodicEventTriggeringConstraintSlLetRecurrence1</SHORT-NAME>
            <EVENT-REF DEST="TD-EVENT-SLLET">/SystemLevelLogicalExecutionTime/TimingExtensions/SystemTiming/
              RPort1_Release</EVENT-REF>
```

```
            <MINIMUM-INTER-ARRIVAL-TIME>
              <CSE-CODE>3</CSE-CODE>
              <CSE-CODE-FACTOR>20</CSE-CODE-FACTOR>
            </MINIMUM-INTER-ARRIVAL-TIME>
            <JITTER>
              <CSE-CODE>3</CSE-CODE>
              <CSE-CODE-FACTOR>0</CSE-CODE-FACTOR>
            </JITTER>
            <PERIOD>
              <CSE-CODE>3</CSE-CODE>
              <CSE-CODE-FACTOR>20</CSE-CODE-FACTOR>
            </PERIOD>
          </PERIODIC-EVENT-TRIGGERING>
          <LATENCY-TIMING-CONSTRAINT>
            <SHORT-NAME>LatencyTimingConstraintSlLetDuration1</SHORT-NAME>
            <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
            <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/SystemLevelLogicalExecutionTime/TimingExtensions/
               SystemTiming/TimingDescriptionChainSlLet1</SCOPE-REF>
            <MAXIMUM>
              <CSE-CODE>3</CSE-CODE>
              <CSE-CODE-FACTOR>400</CSE-CODE-FACTOR>
            </MAXIMUM>
          </LATENCY-TIMING-CONSTRAINT>
        </TIMING-REQUIREMENTS>
        <!-- end example sllet duration/recurrence - line 101 -->
      </SYSTEM-TIMING>
    </ELEMENTS>
  </AR-PACKAGE>
 </AR-PACKAGES>
</AR-PACKAGE>
</AR-PACKAGES>
</AUTOSAR>
```

Listing C.12: Timing Extensions - System Level Logical Execution Time

C.5 Timing Description Event Occurrence

C.5.1 Timing Description Event Occurrence - Filter Expression

```
1  // Condition 1
2  TIMEX_occurs( reference( "TD-EVENT-VARIABLE-DATA-PROTOTYPE:/Example/Expression/E1" ) )
3    || TIMEX_occurs( reference( "TD-EVENT-VARIABLE-DATA-PROTOTYPE:/Example/Expression/E2" ) )
4  // Condition 2
5  && TIMEX_value( reference( "AUTOSAR-VARIABLE-INSTANCE:/Example/Expression/ECX/avi_DE3" ) ) > 3
6  // Condition 3
7  && abs( TIMEX_timeSinceLastOccurrence( reference( "TD-EVENT-VARIABLE-DATA-PROTOTYPE:/Example/Expression/E1" ) ) -
8     TIMEX_timeSinceLastOccurrence( reference( "TD-EVENT-VARIABLE-DATA-PROTOTYPE:/Example/Expression/E2" ) ) ) <= 0.0005
```

Listing C.13: Event Occurrence Filter

C.5.2 Timing Description Event Occurrence - Filter Expression in ARXML

```
<?xml version="1.0" encoding="UTF-8"?>
<AUTOSAR xmlns="http://autosar.org/schema/r4.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
   http://autosar.org/schema/r4.0_AUTOSAR_00052.xsd">
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>Example</SHORT-NAME>
      <ELEMENTS>
        <VFB-TIMING>
          <SHORT-NAME>Expression</SHORT-NAME>
          <SHORT-NAME-PATTERN />
          <TIMING-DESCRIPTIONS>
            <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME>E1</SHORT-NAME>
              <PORT-REF DEST="R-PORT-PROTOTYPE">/Example/Swc1/RequiredPort</PORT-REF>
              <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Example/SenderReceiverInterface1/DE1</DATA-ELEMENT-REF>
              <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
            </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
            <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME>E2</SHORT-NAME>
              <PORT-REF DEST="R-PORT-PROTOTYPE">/Example/Swc1/RequiredPort</PORT-REF>
              <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Example/SenderReceiverInterface1/DE2</DATA-ELEMENT-REF>
              <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
            </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
            <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
              <SHORT-NAME>E3</SHORT-NAME>
              <PORT-REF DEST="R-PORT-PROTOTYPE">/Example/Swc1/RequiredPort</PORT-REF>
              <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Example/SenderReceiverInterface1/DE3</DATA-ELEMENT-REF>
              <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
            </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
            <TD-EVENT-COMPLEX>
              <SHORT-NAME>ECX</SHORT-NAME>
              <OCCURRENCE-EXPRESSION>
```

```
            <FORMULA>
          ((TIMEX_occurs(&lt;EVENT-REF DEST=&quot;TD-EVENT-VARIABLE-DATA-PROTOTYPE&quot;&gt;/Example/Expression/E1&lt;/EVENT-
             REF&gt;)
          || TIMEX_occurs(&lt;EVENT-REF DEST=&quot;TD-EVENT-VARIABLE-DATA-PROTOTYPE&quot;&gt;/Example/Expression/E2&lt;/EVENT-
             REF&gt;))
          &amp;&amp;
          TIMEX_value(&lt;VARIABLE-REF DEST=&quot;AUTOSAR-VARIABLE-INSTANCE&quot;&gt;/Example/Expression/ECX/avi_DE3 &lt;/
             VARIABLE-REF&gt;) &gt; 3
          &amp;&amp;
          abs(TIMEX_timeSinceLastOccurrence(&lt;EVENT-REF DEST=&quot;TD-EVENT-VARIABLE-DATA-PROTOTYPE&quot;&gt;
            /Example/Expression/E1&lt;/EVENT-REF&gt;)
          - TIMEX_timeSinceLastOccurrence(&lt;EVENT-REF DEST=&quot;TD-EVENT-VARIABLE-DATA-PROTOTYPE&quot;&gt;
              /Example/Expression/E2&lt;EVENT-REF&gt;))
          &lt;= 0.0005)
            </FORMULA>
          <VARIABLES>
          <AUTOSAR-VARIABLE-INSTANCE>
              <SHORT-NAME>avi_DE3</SHORT-NAME>
            <VARIABLE-INSTANCE-IREF>
                    <CONTEXT-PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">/Example/Swc1/ProvidedPort</CONTEXT-PORT-PROTOTYPE
                       -REF>
                    <TARGET-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">/Example/SenderReceiverInterface1/DE3</TARGET
                       -DATA-PROTOTYPE-REF>
          </VARIABLE-INSTANCE-IREF>
            </AUTOSAR-VARIABLE-INSTANCE>
            </VARIABLES>
            </OCCURRENCE-EXPRESSION>
          </TD-EVENT-COMPLEX>
        </TIMING-DESCRIPTIONS>
        <COMPONENT-REF DEST="APPLICATION-SW-COMPONENT-TYPE">/Example/Swc1</COMPONENT-REF>
      </VFB-TIMING>
      <SENDER-RECEIVER-INTERFACE>
        <SHORT-NAME>SenderReceiverInterface1</SHORT-NAME>
        <DATA-ELEMENTS>
          <VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>DE1</SHORT-NAME>
          </VARIABLE-DATA-PROTOTYPE>
          <VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>DE2</SHORT-NAME>
          </VARIABLE-DATA-PROTOTYPE>
          <VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>DE3</SHORT-NAME>
          </VARIABLE-DATA-PROTOTYPE>
        </DATA-ELEMENTS>
      </SENDER-RECEIVER-INTERFACE>
      <APPLICATION-SW-COMPONENT-TYPE>
        <SHORT-NAME>Swc1</SHORT-NAME>
        <PORTS>
          <P-PORT-PROTOTYPE>
            <SHORT-NAME>ProvidedPort</SHORT-NAME>
          </P-PORT-PROTOTYPE>
          <R-PORT-PROTOTYPE>
            <SHORT-NAME>RequiredPort</SHORT-NAME>
            <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/Example/SenderReceiverInterface1</REQUIRED-INTERFACE-
               TREF>
          </R-PORT-PROTOTYPE>
        </PORTS>
      </APPLICATION-SW-COMPONENT-TYPE>
    </ELEMENTS>
  </AR-PACKAGE>
</AR-PACKAGES>
</AUTOSAR>
```

Listing C.14: AUTOSAR ARXML representation of the occurrence expression for the complex event ”ECX”

C.5.3 Timing Description Event Occurrence - Grammar

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

Listing C.15: AUTOSAR Timing Extensions Occurrence Expression Language

C.6 Application Notes

C.6.1 Application Notes - VFB Elements

```
  <AR-PACKAGE>
    <SHORT-NAME>Interfaces</SHORT-NAME>
```

```
    <ELEMENTS>
      <SENDER-RECEIVER-INTERFACE>
        <SHORT-NAME>InternalSensorData</SHORT-NAME>
        <DATA-ELEMENTS>
          <VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>internValueX</SHORT-NAME>
            <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/Float</TYPE-TREF>
          </VARIABLE-DATA-PROTOTYPE>
          <VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>internValueY</SHORT-NAME>
            <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/Float</TYPE-TREF>
          </VARIABLE-DATA-PROTOTYPE>
          <VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>internValueZ</SHORT-NAME>
            <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/Float</TYPE-TREF>
          </VARIABLE-DATA-PROTOTYPE>
        </DATA-ELEMENTS>
      </SENDER-RECEIVER-INTERFACE>
      <SENDER-RECEIVER-INTERFACE>
        <SHORT-NAME>OutputSensorData</SHORT-NAME>
        <DATA-ELEMENTS>
          <VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>outValueX</SHORT-NAME>
            <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/UInt16</TYPE-TREF>
          </VARIABLE-DATA-PROTOTYPE>
          <VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>outValueY</SHORT-NAME>
            <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/UInt16</TYPE-TREF>
          </VARIABLE-DATA-PROTOTYPE>
          <VARIABLE-DATA-PROTOTYPE>
            <SHORT-NAME>outValueZ</SHORT-NAME>
            <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/Datatypes/UInt16</TYPE-TREF>
          </VARIABLE-DATA-PROTOTYPE>
        </DATA-ELEMENTS>
      </SENDER-RECEIVER-INTERFACE>
    </ELEMENTS>
  </AR-PACKAGE>
  <AR-PACKAGE>
    <SHORT-NAME>SensorPackage</SHORT-NAME>
    <ELEMENTS>
      <SENSOR-ACTUATOR-SW-COMPONENT-TYPE>
        <SHORT-NAME>SensorComponent</SHORT-NAME>
        <PORTS>
          <P-PORT-PROTOTYPE>
            <SHORT-NAME>InternalSensorData</SHORT-NAME>
            <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/Interfaces/InternalSensorData</PROVIDED-INTERFACE-
                TREF>
          </P-PORT-PROTOTYPE>
        </PORTS>
        <INTERNAL-BEHAVIORS>
          <SWC-INTERNAL-BEHAVIOR>
            <SHORT-NAME>SensorBehavior</SHORT-NAME>
            <RUNNABLES>
              <RUNNABLE-ENTITY>
                <SHORT-NAME>SensorRunnable</SHORT-NAME>
              </RUNNABLE-ENTITY>
            </RUNNABLES>
          </SWC-INTERNAL-BEHAVIOR>
        </INTERNAL-BEHAVIORS>
      </SENSOR-ACTUATOR-SW-COMPONENT-TYPE>
      <APPLICATION-SW-COMPONENT-TYPE>
        <SHORT-NAME>ConditioningComponent</SHORT-NAME>
        <PORTS>
          <R-PORT-PROTOTYPE>
            <SHORT-NAME>UnprocessedSensorData</SHORT-NAME>
            <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/Interfaces/InternalSensorData</REQUIRED-INTERFACE-
                TREF>
          </R-PORT-PROTOTYPE>
          <P-PORT-PROTOTYPE>
            <SHORT-NAME>ProcessedSensorData</SHORT-NAME>
            <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/Interfaces/InternalSensorData</PROVIDED-INTERFACE-
                TREF>
          </P-PORT-PROTOTYPE>
        </PORTS>
        <INTERNAL-BEHAVIORS>
          <SWC-INTERNAL-BEHAVIOR>
            <SHORT-NAME>ConditioningBehavior</SHORT-NAME>
            <RUNNABLES>
              <RUNNABLE-ENTITY>
                <SHORT-NAME>ConditioningRunnable</SHORT-NAME>
              </RUNNABLE-ENTITY>
            </RUNNABLES>
          </SWC-INTERNAL-BEHAVIOR>
        </INTERNAL-BEHAVIORS>
      </APPLICATION-SW-COMPONENT-TYPE>
      <APPLICATION-SW-COMPONENT-TYPE>
        <SHORT-NAME>OutputComponent</SHORT-NAME>
        <PORTS>
          <R-PORT-PROTOTYPE>
            <SHORT-NAME>UnprocessedSensorData</SHORT-NAME>
            <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">/Interfaces/InternalSensorData</REQUIRED-INTERFACE-
                TREF>
          </R-PORT-PROTOTYPE>
          <P-PORT-PROTOTYPE>
            <SHORT-NAME>ProcessedSensorData</SHORT-NAME>
```

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

Listing C.16: SWC Descriptions and Interface Definitions

C.6.2 Application Notes - VFB Timing - Latency Constraints

```
    <SHORT-NAME>SensorVfbTiming</SHORT-NAME>
    <TIMING-DESCRIPTIONS>
      <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
        <SHORT-NAME>ConditioningReceived</SHORT-NAME>
        <IS-EXTERNAL>false</IS-EXTERNAL>
        <PORT-REF DEST="R-PORT-PROTOTYPE">/SensorPackage/ConditioningComponent/UnprocessedSensorData</PORT-REF>
        <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/InternalSensorData/internValueX</DATA-ELEMENT-REF>
        <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
      </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
        <SHORT-NAME>ConditioningSent</SHORT-NAME>
        <IS-EXTERNAL>false</IS-EXTERNAL>
        <PORT-REF DEST="P-PORT-PROTOTYPE">/SensorPackage/ConditioningComponent/ProcessedSensorData</PORT-REF>
        <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/InternalSensorData/internValueX</DATA-ELEMENT-REF>
        <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
      </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
        <SHORT-NAME>SensorDataProduced</SHORT-NAME>
        <IS-EXTERNAL>true</IS-EXTERNAL>
        <PORT-REF DEST="R-PORT-PROTOTYPE">/SensorPackage/ConditioningComponent/UnprocessedSensorData</PORT-REF>
        <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/InternalSensorData/internValueX</DATA-ELEMENT-REF>
        <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
      </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
      <TIMING-DESCRIPTION-EVENT-CHAIN>
        <SHORT-NAME>InputVfbChain</SHORT-NAME>
        <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/TimingPackage/SensorVfbTiming/SensorDataProduced</
            STIMULUS-REF>
        <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/TimingPackage/SensorVfbTiming/ConditioningReceived</
            RESPONSE-REF>
        <SEGMENT-REFS>
          <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingPackage/SensorVfbTiming/InputVfbChain</SEGMENT-REF>
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
    </TIMING-DESCRIPTIONS>
    <TIMING-REQUIREMENTS>
      <LATENCY-TIMING-CONSTRAINT>
        <SHORT-NAME>InputVfbLatency</SHORT-NAME>
        <LATENCY-CONSTRAINT-TYPE>AGE</LATENCY-CONSTRAINT-TYPE>
        <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/TimingPackage/SensorVfbTiming/InputVfbChain</SCOPE-REF>
        <MINIMUM>
          <CSE-CODE>1</CSE-CODE>
          <CSE-CODE-FACTOR>40</CSE-CODE-FACTOR>
        </MINIMUM>
        <MAXIMUM>
          <CSE-CODE>1</CSE-CODE>
          <CSE-CODE-FACTOR>50</CSE-CODE-FACTOR>
        </MAXIMUM>
      </LATENCY-TIMING-CONSTRAINT>
      <EXECUTION-ORDER-CONSTRAINT>
        <SHORT-NAME>EOC1</SHORT-NAME>
        <ORDERED-ELEMENTS>
          <EOC-EXECUTABLE-ENTITY-REF>
```

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

Listing C.17: LatencyConstraint and related events on VfbTiming view

C.6.3 Application Notes - VFB Timing - Execution Order Constraints

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

Listing C.18: Execution Order Constraint for Three Runnable Entities

C.6.4 Application Notes - COM Frame Timing

```
      <FRAME-REF DEST="FLEXRAY-FRAME">/SystemDescriptionPackage/SensorFrame</FRAME-REF>
      <PHYSICAL-CHANNEL-REF DEST="FLEXRAY-PHYSICAL-CHANNEL">/SystemDescriptionPackage/SampleFrCluster/FrChannel110MBit
          </PHYSICAL-CHANNEL-REF>
      <TD-EVENT-TYPE>FRAME-TRANSMITTED-ON-BUS</TD-EVENT-TYPE>
    </TD-EVENT-FRAME>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>SensorEcuChain</SHORT-NAME>
```

Listing C.19: Event describing the point in time where data is sent on the bus

C.6.5 Application Notes - COM Frame Timing - Timing Description Event Chain

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

Listing C.20: Event chain describing the sending path of data

C.6.6 Application Notes - COM Frame Timing - Latency Constraint

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

Listing C.21: Latency constraint prescribing the maximum latency of sending path within the ECU

C.7 Application Notes - Engine Control

C.7.1 Application Notes - Engine Control - Requirement 1

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

```
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/IVotedPedalPosition/VotedPedalPosition</DATA-
        ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>RVotedPedalPositionReceived</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="R-PORT-PROTOTYPE">/Components/ThrottleController/RVotedPedalPosition</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/IVotedPedalPosition/VotedPedalPosition</DATA-
        ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>PUnlimitedThrottlePositionSent</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="P-PORT-PROTOTYPE">/Components/ThrottleController/PUnlimitedThrottlePosition</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/IUnlimitedThrottlePosition/
        UnlimitedThrottlePosition</DATA-ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>RUnlimitedThrottlePositionReceived</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="R-PORT-PROTOTYPE">/Components/ThrottleActuator/RUnlimitedThrottlePosition</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/IUnlimitedThrottlePosition/
        UnlimitedThrottlePosition</DATA-ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-RECEIVED</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TD-EVENT-VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>PUnlimitedThrottlePositionActuatorSent</SHORT-NAME>
    <IS-EXTERNAL>false</IS-EXTERNAL>
    <PORT-REF DEST="P-PORT-PROTOTYPE">/Components/ThrottleActuator/PUnlimitedThrottlePositionActuator</PORT-REF>
    <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/Interfaces/IUnlimitedThrottlePosition/
        UnlimitedThrottlePosition</DATA-ELEMENT-REF>
    <TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>VARIABLE-DATA-PROTOTYPE-SENT</TD-EVENT-VARIABLE-DATA-PROTOTYPE-TYPE>
  </TD-EVENT-VARIABLE-DATA-PROTOTYPE>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg0</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
        RAcceleratorPedalPositionSensorReceived</STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
        PAcceleratorPedalPositionSent</RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg0</SEGMENT-
          REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg1</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
        PAcceleratorPedalPositionSent</STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
        RAcceleratorPedalPositionReceived</RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg1</SEGMENT-
          REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg1_1</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
        RAcceleratorPedalPositionReceived</STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/PVotedPedalPositionSent</
        RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg1_1</SEGMENT
          -REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg2</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/PVotedPedalPositionSent</
        STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
        RVotedPedalPositionReceived</RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg2</SEGMENT-
          REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg2_1</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
        RVotedPedalPositionReceived</STIMULUS-REF>
    <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
        PUnlimitedThrottlePositionSent</RESPONSE-REF>
    <SEGMENT-REFS>
      <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg2_1</SEGMENT
          -REF>
    </SEGMENT-REFS>
  </TIMING-DESCRIPTION-EVENT-CHAIN>
  <TIMING-DESCRIPTION-EVENT-CHAIN>
    <SHORT-NAME>TimingChain1Seg3</SHORT-NAME>
    <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
        PUnlimitedThrottlePositionSent</STIMULUS-REF>
```

```
      <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
          RUnlimitedThrottlePositionReceived</RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg3</SEGMENT-
            REF>
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>TimingChain1Seg4</SHORT-NAME>
      <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
          RUnlimitedThrottlePositionReceived</STIMULUS-REF>
      <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
          PUnlimitedThrottlePositionActuatorSent</RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg4</SEGMENT-
            REF>
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>TimingChain1AllSeg</SHORT-NAME>
      <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
          RAcceleratorPedalPositionSensorReceived</STIMULUS-REF>
      <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
          PUnlimitedThrottlePositionActuatorSent</RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg0</SEGMENT-
            REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg1</SEGMENT-
            REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg1_1</SEGMENT
            -REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg2</SEGMENT-
            REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg2_1</SEGMENT
            -REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg3</SEGMENT-
            REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1Seg4</SEGMENT-
            REF>
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
  </TIMING-DESCRIPTIONS>
  <TIMING-REQUIREMENTS>
    <LATENCY-TIMING-CONSTRAINT>
      <SHORT-NAME>TimingChain1AllSegLatency</SHORT-NAME>
      <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
      <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1AllSeg</SCOPE-REF>
      <MINIMUM>
        <CSE-CODE>1</CSE-CODE>
        <CSE-CODE-FACTOR>28</CSE-CODE-FACTOR>
      </MINIMUM>
      <MAXIMUM>
        <CSE-CODE>1</CSE-CODE>
        <CSE-CODE-FACTOR>30</CSE-CODE-FACTOR>
      </MAXIMUM>
    </LATENCY-TIMING-CONSTRAINT>
  </TIMING-REQUIREMENTS>
  <COMPONENT-REF DEST="COMPOSITION-SW-COMPONENT-TYPE">/Components/EngineControl</COMPONENT-REF>
</VFB-TIMING>
```

Listing C.22: Event Definitions and Constraints for Requirement 1

C.7.2 Application Notes - Engine Control - Requirement 2

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

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

Listing C.23: Event Definitions and Constraints for Requirement 2

C.7.3 Application Notes - Engine Control - Requirement 3

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

```
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming3/TimingChain3Seg0</SEGMENT-
            REF>
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>TimingChain3Seg1</SHORT-NAME>
      <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming3/PMafRateOutSent</STIMULUS-
          REF>
      <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming3/RMafRateOutReceived</
          RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming3/TimingChain3Seg1_1</SEGMENT
            -REF>
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>TimingChain3Seg1_1</SHORT-NAME>
      <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming3/RMafRateOutReceived</
          STIMULUS-REF>
      <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming3/PIgnitionTimingSent</
          RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming3/TimingChain3Seg1_1</SEGMENT
            -REF>
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>TimingChain3Seg2</SHORT-NAME>
      <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming3/PIgnitionTimingSent</
          STIMULUS-REF>
      <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming3/RIgnitionTimingReceived</
          RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming3/TimingChain3Seg2</SEGMENT-
            REF>
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>TimingChain3AllSeg</SHORT-NAME>
      <STIMULUS-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
          RAcceleratorPedalPositionSensorReceived</STIMULUS-REF>
      <RESPONSE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming3/RIgnitionTimingReceived</
          RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming1/TimingChain1AllSeg</SEGMENT
            -REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming2/TimingChain2Seg0</SEGMENT-
            REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming2/TimingChain2Seg1</SEGMENT-
            REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming3/TimingChain3Seg0</SEGMENT-
            REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming3/TimingChain3Seg1</SEGMENT-
            REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming3/TimingChain3Seg1_1</SEGMENT
            -REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming3/TimingChain3Seg2</SEGMENT-
            REF>
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
  </TIMING-DESCRIPTIONS>
  <TIMING-GUARANTEES>
    <LATENCY-TIMING-CONSTRAINT>
      <SHORT-NAME>TimingChain3AllSegLatency1</SHORT-NAME>
      <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
      <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming3/TimingChain3AllSeg</SCOPE-REF>
      <MAXIMUM>
        <CSE-CODE>3</CSE-CODE>
        <CSE-CODE-FACTOR>50</CSE-CODE-FACTOR>
      </MAXIMUM>
    </LATENCY-TIMING-CONSTRAINT>
  </TIMING-GUARANTEES>
  <TIMING-REQUIREMENTS>
    <OFFSET-TIMING-CONSTRAINT>
      <SHORT-NAME>Otc</SHORT-NAME>
      <SOURCE-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming1/
          PUnlimitedThrottlePositionActuatorSent</SOURCE-REF>
      <TARGET-REF DEST="TD-EVENT-VARIABLE-DATA-PROTOTYPE">/Timing/EngineControlVfbTiming2/RThrottlePositionReceived</
          TARGET-REF>
      <MAXIMUM>
        <CSE-CODE>3</CSE-CODE>
        <CSE-CODE-FACTOR>5</CSE-CODE-FACTOR>
      </MAXIMUM>
    </OFFSET-TIMING-CONSTRAINT>
    <LATENCY-TIMING-CONSTRAINT>
      <SHORT-NAME>TimingChain3AllSegLatency2</SHORT-NAME>
      <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
      <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlVfbTiming3/TimingChain3AllSeg</SCOPE-REF>
      <MAXIMUM>
        <CSE-CODE>3</CSE-CODE>
        <CSE-CODE-FACTOR>50</CSE-CODE-FACTOR>
      </MAXIMUM>
    </LATENCY-TIMING-CONSTRAINT>
  </TIMING-REQUIREMENTS>
```

```
</VFB-TIMING>
```

Listing C.24: Event Definitions and Constraints for Requirement 3

C.7.4 Application Notes - Engine Control - Requirement 4

```
<ECU-TIMING>
  <SHORT-NAME>EngineControlEcuTiming</SHORT-NAME>
  <TIMING-DESCRIPTIONS>
    <TD-EVENT-BSW-INTERNAL-BEHAVIOR>
      <SHORT-NAME>CamShaftISRActivated</SHORT-NAME>
      <BSW-MODULE-ENTITY-REF DEST="BSW-INTERRUPT-ENTITY">/Modules/CamShaft/CamShaftBehavior/CamShaftISR</BSW-MODULE-
          ENTITY-REF>
      <TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>BSW-MODULE-ENTITY-ACTIVATED</TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>
    </TD-EVENT-BSW-INTERNAL-BEHAVIOR>
    <TD-EVENT-BSW-INTERNAL-BEHAVIOR>
      <SHORT-NAME>CamShaftISRStarted</SHORT-NAME>
      <BSW-MODULE-ENTITY-REF DEST="BSW-INTERRUPT-ENTITY">/Modules/CamShaft/CamShaftBehavior/CamShaftISR</BSW-MODULE-
          ENTITY-REF>
      <TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>BSW-MODULE-ENTITY-STARTED</TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>
    </TD-EVENT-BSW-INTERNAL-BEHAVIOR>
    <TD-EVENT-BSW-INTERNAL-BEHAVIOR>
      <SHORT-NAME>CamShaftIsrTerminated</SHORT-NAME>
      <BSW-MODULE-ENTITY-REF DEST="BSW-INTERRUPT-ENTITY">/Modules/CamShaft/CamShaftBehavior/CamShaftISR</BSW-MODULE-
          ENTITY-REF>
      <TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>BSW-MODULE-ENTITY-TERMINATED</TD-EVENT-BSW-INTERNAL-BEHAVIOR-TYPE>
    </TD-EVENT-BSW-INTERNAL-BEHAVIOR>
    <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
      <SHORT-NAME>IgnitionActuatorCalculationActivated</SHORT-NAME>
      <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/Components/IngnitionActuator/IgnitionActuatorBehavior/
          IngnitionActuatorCalculation</RUNNABLE-REF>
      <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-ACTIVATED</TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
    </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
    <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
      <SHORT-NAME>IgnitionActuatorCalculationStarted</SHORT-NAME>
      <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/Components/IngnitionActuator/IgnitionActuatorBehavior/
          IngnitionActuatorCalculation</RUNNABLE-REF>
      <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-STARTED</TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
    </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
    <TD-EVENT-SWC-INTERNAL-BEHAVIOR>
      <SHORT-NAME>IgnitionActuatorCalculationTerminated</SHORT-NAME>
      <RUNNABLE-REF DEST="RUNNABLE-ENTITY">/Components/IngnitionActuator/IgnitionActuatorBehavior/
          IngnitionActuatorCalculation</RUNNABLE-REF>
      <TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>RUNNABLE-ENTITY-TERMINATED</TD-EVENT-SWC-INTERNAL-BEHAVIOR-TYPE>
    </TD-EVENT-SWC-INTERNAL-BEHAVIOR>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>TimingChain4Seg1</SHORT-NAME>
      <STIMULUS-REF DEST="TD-EVENT-BSW-INTERNAL-BEHAVIOR">/Timing/EngineControlEcuTiming/CamShaftISRStarted</STIMULUS-
          REF>
      <RESPONSE-REF DEST="TD-EVENT-BSW-INTERNAL-BEHAVIOR">/Timing/EngineControlEcuTiming/CamShaftIsrTerminated</
          RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlEcuTiming/TimingChain4Seg1</SEGMENT-
            REF>
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>TimingChain4Seg2</SHORT-NAME>
      <STIMULUS-REF DEST="TD-EVENT-BSW-INTERNAL-BEHAVIOR">/Timing/EngineControlEcuTiming/CamShaftIsrTerminated</
          STIMULUS-REF>
      <RESPONSE-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/Timing/EngineControlEcuTiming/
          IgnitionActuatorCalculationStarted</RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlEcuTiming/TimingChain4Seg2</SEGMENT-
            REF>
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>TimingChain4Seg3</SHORT-NAME>
      <STIMULUS-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/Timing/EngineControlEcuTiming/
          IgnitionActuatorCalculationStarted</STIMULUS-REF>
      <RESPONSE-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/Timing/EngineControlEcuTiming/
          IgnitionActuatorCalculationTerminated</RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlEcuTiming/TimingChain4Seg3</SEGMENT-
            REF>
      </SEGMENT-REFS>
    </TIMING-DESCRIPTION-EVENT-CHAIN>
    <TIMING-DESCRIPTION-EVENT-CHAIN>
      <SHORT-NAME>TimingChain4AllSeg</SHORT-NAME>
      <STIMULUS-REF DEST="TD-EVENT-BSW-INTERNAL-BEHAVIOR">/Timing/EngineControlEcuTiming/CamShaftISRStarted</STIMULUS-
          REF>
      <RESPONSE-REF DEST="TD-EVENT-SWC-INTERNAL-BEHAVIOR">/Timing/EngineControlEcuTiming/
          IgnitionActuatorCalculationTerminated</RESPONSE-REF>
      <SEGMENT-REFS>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlEcuTiming/TimingChain4Seg1</SEGMENT-
            REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlEcuTiming/TimingChain4Seg2</SEGMENT-
            REF>
        <SEGMENT-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlEcuTiming/TimingChain4Seg3</SEGMENT-
            REF>
```

```
        </SEGMENT-REFS>
      </TIMING-DESCRIPTION-EVENT-CHAIN>
    </TIMING-DESCRIPTIONS>
    <TIMING-REQUIREMENTS>
      <LATENCY-TIMING-CONSTRAINT>
        <SHORT-NAME>TimingChain4AllSegLatency</SHORT-NAME>
        <LATENCY-CONSTRAINT-TYPE>REACTION</LATENCY-CONSTRAINT-TYPE>
        <SCOPE-REF DEST="TIMING-DESCRIPTION-EVENT-CHAIN">/Timing/EngineControlEcuTiming/TimingChain4AllSeg</SCOPE-REF>
        <MAXIMUM>
          <CSE-CODE>3</CSE-CODE>
          <CSE-CODE-FACTOR>3</CSE-CODE-FACTOR>
        </MAXIMUM>
      </LATENCY-TIMING-CONSTRAINT>
    </TIMING-REQUIREMENTS>
  </ECU-TIMING>
```

Listing C.25: Event Definitions and Constraints for Requirement 4

C.7.5 Application Notes - Engine Control - Requirement 5

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

Listing C.26: Event Definitions and Constraints for Requirement 5

D Mentioned Class Tables

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

Class	**ARPackage**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	AUTOSAR package, allowing to create top level packages to structure the contained ARElements. ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package. This is an extended version of MSR's SW-SYSTEM.			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ARPackage.arPackage, AUTOSAR.arPackage			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arPackage	ARPackage	*	aggr	This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arPackage.shortName, arPackage.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableElement	*	aggr	Elements that are part of this package **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=element.shortName, element.variation Point.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20
referenceBase	ReferenceBase	*	aggr	This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=referenceBase.shortLabel xml.sequenceOffset=10

Table D.1: ARPackage

Class	**AUTOSAR**			
Package	M2::AUTOSARTemplates::AutosarTopLevelStructure			
Note	Root element of an AUTOSAR description, also the root element in corresponding XML documents. **Tags:** xml.globalElement=true			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	AUTOSAR			
adminData	AdminData	0..1	aggr	This represents the administrative data of an Autosar file. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=adminData xml.sequenceOffset=10
arPackage	ARPackage	*	aggr	This is the top level package in an AUTOSAR model. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arPackage.shortName, arPackage.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
fileInfoComment	FileInfoComment	0..1	aggr	This represents a possibility to provide a structured comment in an AUTOSAR file. **Stereotypes:** atpStructuredComment **Tags:** xml.roleElement=true xml.sequenceOffset=-10 xml.typeElement=false
introduction	DocumentationBlock	0..1	aggr	This represents an introduction on the Autosar file. It is intended for example to represent disclaimers and legal notes. **Tags:** xml.sequenceOffset=20

Table D.2: AUTOSAR

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

Table D.3: AbstractEvent

Class	*AbstractRequiredPortPrototype* (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This abstract class provides the ability to become a required PortPrototype.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Subclasses	PRPortPrototype, RPortPrototype			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	***AbstractRequiredPortPrototype*** (abstract)			
requiredCom Spec	RPortComSpec	*	aggr	Required communication attributes, one for each interface element. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredComSpec

Table D.4: AbstractRequiredPortPrototype

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

Table D.5: ApplicationSwComponentType

Class	***AtomicSwComponentType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	An atomic software component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Subclasses	ApplicationSwComponentType, ComplexDeviceDriverSwComponentType, EcuAbstractionSwComponentType, NvBlockSwComponentType, SensorActuatorSwComponentType, ServiceProxySwComponentType, ServiceSwComponentType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
internalBehavior	SwcInternalBehavior	0..1	aggr	The SwcInternalBehaviors owned by an AtomicSwComponentType can be located in a different physical file. Therefore the aggregation is <<atpSplitable>>. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalBehavior.shortName, internalBehavior.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the AtomicSwComponentType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=symbolProps.shortName

Table D.6: AtomicSwComponentType

Class	***AtpInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::AbstractStructure			
Note	An M0 instance of a classifier may be represented as a tree rooted at that instance, where under each node come the sub-trees representing the instances which act as features under that node. An instance ref specifies a navigation path from any M0 tree-instance of the base (which is a classifier) to a leaf (which is an instance of the target).			
Base	*ARObject*			
Subclasses	AnyInstanceRef, ApplicationCompositeElementInPortInterfaceInstanceRef, ComponentInCompositionInstanceRef, ComponentInSystemInstanceRef, *DataPrototypeInPortInterfaceInstanceRef*, DataPrototypeInSystemInstanceRef, InnerDataPrototypeGroupInCompositionInstanceRef, InnerPortGroupInCompositionInstanceRef, InnerRunnableEntityGroupInCompositionInstanceRef, InstanceEventInCompositionInstanceRef, ModeDeclarationGroupPrototypeInSystemInstanceRef, *ModeGroupInAtomicSwcInstanceRef*, ModeInBswModuleDescriptionInstanceRef, ModeInSwcInstanceRef, OperationArgumentInComponentInstanceRef, *OperationInAtomicSwcInstanceRef*, OperationInSystemInstanceRef, PModeInSystemInstanceRef, ParameterDataPrototypeInSystemInstanceRef, ParameterInAtomicSWCTypeInstanceRef, PortGroupInSystemInstanceRef, *PortInCompositionTypeInstanceRef*, RModeInAtomicSwcInstanceRef, RteEventInCompositionInstanceRef, RteEventInEcuInstanceRef, RteEventInSystemInstanceRef, RunnableEntityInCompositionInstanceRef, SwcServiceDependencyInSystemInstanceRef, *TriggerInAtomicSwcInstanceRef*, TriggerInSystemInstanceRef, VariableAccessInEcuInstanceRef, VariableDataPrototypeInCompositionInstanceRef, VariableDataPrototypeInSystemInstanceRef, VariableInAtomicSWCTypeInstanceRef, *VariableInAtomicSwcInstanceRef*, VariableInComponentInstanceRef			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
atpBase	AtpClassifier	1	ref	This is the base from which the navigaion path starts. **Stereotypes:** atpAbstract; atpDerived
atpContextElement (ordered)	AtpPrototype	*	ref	This is one particular step in the navigation path. **Stereotypes:** atpAbstract
atpTarget	AtpFeature	1	ref	This is the target of the instance ref. In other words it is the terminal of the navigation path. **Stereotypes:** atpAbstract

Table D.7: AtpInstanceRef

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

Table D.8: Boolean

Class	***BswEvent*** (abstract)			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	Base class of various kinds of events which are used to trigger a BswModuleEntity of this BSW module or cluster. The event is local to the BSW module or cluster. The short name of the meta-class instance is intended as an input to configure the required API of the BSW Scheduler.			
Base	*ARObject*, *AbstractEvent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswInterruptEvent, BswOperationInvokedEvent, *BswScheduleEvent*			
Aggregated by	BswInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	***BswEvent*** (abstract)			
context Limitation	BswDistinguished Partition	*	ref	The existence of this reference indicates that the usage of the event is limited to the context of the referred Bsw DistinguishedPartitions.
disabledInMode	ModeDeclaration	*	iref	The modes, in which this event is disabled. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=disabledInMode.contextMode DeclarationGroup, disabledInMode.targetMode **InstanceRef implemented by:** ModeInBswModule DescriptionInstanceRef
startsOnEvent	BswModuleEntity	0..1	ref	The entity which is started by the event.

Table D.9: BswEvent

Class	**BswImplementation**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswImplementation			
Note	Contains the implementation specific information in addition to the generic specification (BswModule Description and BswBehavior). It is possible to have several different BswImplementations referring to the same BswBehavior. **Tags:** atp.recommendedPackage=BswImplementations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *Implementation*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arRelease Version	RevisionLabelString	0..1	attr	Version of the AUTOSAR Release on which this implementation is based. The numbering contains three levels (major, minor, revision) which are defined by AUTOSAR.
behavior	BswInternalBehavior	0..1	ref	The behavior of this implementation. This relation is made as an association because • it follows the pattern of the SWCT • since ARElement cannot be splitted, but we want supply the implementation later, the Bsw Implementation is not aggregated in BswBehavior
preconfigured Configuration	EcucModule ConfigurationValues	*	ref	Reference to the set of preconfigured (i.e. fixed) configuration values for this BswImplementation. If the BswImplementation represents a cluster of several modules, more than one EcucModuleConfigurationValues element can be referred (at most one per module), otherwise at most one such element can be referred. **Tags:** xml.roleWrapperElement=true
recommended Configuration	EcucModule ConfigurationValues	*	ref	Reference to one or more sets of recommended configuration values for this module or module cluster.

▽

△

Class	BswImplementation			
vendorApiInfix	Identifier	0..1	attr	In driver modules which can be instantiated several times on a single ECU, SRS_BSW_00347 requires that the names of files, APIs, published parameters and memory allocation keywords are extended by the vendorId and a vendor specific name. This parameter is used to specify the vendor specific name. In total, the implementation specific API name is generated as follows: <Module Name>_<vendorId>_ <vendorApiInfix>_<API name from SWS>. E.g. assuming that the vendorId of the implementer is 123 and the implementer chose a vendorApiInfix of "v11r456" an API name Can_Write defined in the SWS will translate to Can_123_v11r456_Write. This attribute is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1. See also SWS_BSW_00102.
vendorSpecific ModuleDef	EcucModuleDef	*	ref	Reference to • the vendor specific EcucModuleDef used in this Bsw Implementation if it represents a single module • several EcucModuleDefs used in this Bsw Implementation if it represents a cluster of modules • one or no EcucModuleDefs used in this Bsw Implementation if it represents a library **Tags:** xml.roleWrapperElement=true

Table D.10: BswImplementation

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

▽

△

Class	**BswInternalBehavior**			
bswPerInstance MemoryPolicy	BswPerInstance MemoryPolicy	*	aggr	Policy for a arTypedPerInstanceMemory The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswPerInstanceMemoryPolicy, bswPer InstanceMemoryPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
clientPolicy	BswClientPolicy	*	aggr	Policy for a requiredClientServerEntry. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=clientPolicy, clientPolicy.variationPoint.short Label vh.latestBindingTime=preCompileTime
distinguished Partition	BswDistinguished Partition	*	aggr	Indicates an abstract partition context in which the enclosing BswModuleEntity can be executed. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=distinguishedPartition.shortName, distinguishedPartition.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
entity	BswModuleEntity	*	aggr	A code entity for which the behavior is described **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=entity.shortName, entity.variationPoint.short Label vh.latestBindingTime=preCompileTime xml.sequenceOffset=5
event	BswEvent	*	aggr	An event required by this module behavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=event.shortName, event.variationPoint.short Label vh.latestBindingTime=preCompileTime xml.sequenceOffset=10
exclusiveArea Policy	BswExclusiveArea Policy	*	aggr	Policy for an ExclusiveArea in this BswInternalBehavior. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveAreaPolicy, exclusiveArea Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
includedData TypeSet	IncludedDataTypeSet	*	aggr	The includedDataTypeSet is used by a basic software module for its implementation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=includedDataTypeSet
includedMode Declaration GroupSet	IncludedMode DeclarationGroupSet	*	aggr	This aggregation represents the included Mode DeclarationGroups **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=includedModeDeclarationGroupSet

▽

△

Class	**BswInternalBehavior**			
internal TriggeringPoint	BswInternalTriggering Point	*	aggr	An internal triggering point. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalTriggeringPoint.shortName, internal TriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=2
internal TriggeringPoint Policy	BswInternalTriggering PointPolicy	*	aggr	Policy for an internalTriggeringPoint in this BswInternal Behavior.. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalTriggeringPointPolicy, internal TriggeringPointPolicy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
modeReceiver Policy	BswModeReceiver Policy	*	aggr	Implementation policy for the reception of mode switches. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeReceiverPolicy, modeReceiver Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25
modeSender Policy	BswModeSenderPolicy	*	aggr	Implementation policy for providing a mode group. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeSenderPolicy, modeSender Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
parameterPolicy	BswParameterPolicy	*	aggr	Policy for a perInstanceParameter in this BswInternal Behavior. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=parameterPolicy, parameterPolicy.variation Point.shortLabel vh.latestBindingTime=preCompileTime
perInstance Parameter	ParameterData Prototype	*	aggr	Describes a read only memory object containing characteristic value(s) needed by this BswInternal Behavior. The role name perInstanceParameter is chosen in analogy to the similar role in the context of SwcInternal Behavior. In contrast to constantMemory, this object is not allocated locally by the module's code, but by the BSW Scheduler and it is accessed from the BSW module via the BSW Scheduler API. The main use case is the support of software emulation of calibration data. The aggregation is subject to variability with the purpose to support implementation variants. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=perInstanceParameter.shortName, per InstanceParameter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45

▽

△

Class	**BswInternalBehavior**			
receptionPolicy	BswDataReception Policy	*	aggr	Data reception policy for inter-partition and/or inter-core communication. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=receptionPolicy, receptionPolicy.variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
releasedTrigger Policy	BswReleasedTrigger Policy	*	aggr	Policy for a releasedTrigger. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=releasedTriggerPolicy, releasedTrigger Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
schedulerName Prefix	BswSchedulerName Prefix	*	aggr	Optional definition of one or more prefixes to be used for the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=schedulerNamePrefix.shortName, scheduler NamePrefix.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
sendPolicy	BswDataSendPolicy	*	aggr	Policy for a providedData. The policy selects the options of the Schedule Manager API generation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=sendPolicy, sendPolicy.variationPoint.short Label vh.latestBindingTime=preCompileTime
service Dependency	BswService Dependency	*	aggr	Defines the requirements on AUTOSAR Services for a particular item. The aggregation is subject to variability with the purpose to support the conditional existence of ServiceNeeds. The aggregation is splitable in order to support that ServiceNeeds might be provided in later development steps. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceDependency.ident.shortName, serviceDependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
triggerDirect Implementation	BswTriggerDirect Implementation	*	aggr	Specifies a trigger to be directly implemented via OS calls. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=triggerDirectImplementation, triggerDirect Implementation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=15
variationPoint Proxy	VariationPointProxy	*	aggr	Proxy of a variation points in the C/C++ implementation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=variationPointProxy.shortName

Table D.11: BswInternalBehavior

Class	BswInterruptEntity			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	BSW module entity, which is designed to be triggered by an interrupt.			
Base	*ARObject*, *BswModuleEntity*, *ExecutableEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	BswInternalBehavior.entity			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
interrupt Category	BswInterruptCategory	0..1	attr	Category of the interrupt
interruptSource	String	0..1	attr	Allows a textual documentation of the intended interrupt source.

Table D.12: BswInterruptEntity

Class	BswModuleDescription			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswOverview			
Note	Root element for the description of a single BSW module or BSW cluster. In case it describes a BSW module, the short name of this element equals the name of the BSW module. **Tags:** atp.recommendedPackage=BswModuleDescriptions			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
bswModule Dependency	BswModuleDependency	*	aggr	Describes the dependency to another BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswModuleDependency.shortName, bswModuleDependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
bswModule Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswModuleDocumentation, bswModuleDocumentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=6
expectedEntry	BswModuleEntry	*	ref	Indicates an entry which is required by this module. Replacement of outgoingCallback / requiredEntry. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=expectedEntry.bswModuleEntry, expectedEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
implemented Entry	BswModuleEntry	*	ref	Specifies an entry provided by this module which can be called by other modules. This includes "main" functions, interrupt routines, and callbacks. Replacement of providedEntry / expectedCallback. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=implementedEntry.bswModuleEntry, implementedEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**BswModuleDescription**			
internalBehavior	BswInternalBehavior	*	aggr	The various BswInternalBehaviors associated with a Bsw ModuleDescription can be distributed over several physical files. Therefore the aggregation is <<atp Splitable>>. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=internalBehavior.shortName xml.sequenceOffset=65
moduleId	PositiveInteger	0..1	attr	Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen. **Tags:** xml.sequenceOffset=5
providedClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module provides a client server entry which can be called from another partition or core.This entry is declared locally to this context and will be connected to the requiredClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=providedClientServerEntry.shortName, providedClientServerEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
providedData	VariableDataPrototype	*	aggr	Specifies a data prototype provided by this module in order to be read from another partition or core.The providedData is declared locally to this context and will be connected to the requiredData of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=providedData.shortName, provided Data.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
providedMode Group	ModeDeclarationGroup Prototype	*	aggr	A set of modes which is owned and provided by this module or cluster. It can be connected to the required ModeGroups of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with modes provided via ports by an associated ServiceSwComponentType, EcuAbstraction SwComponentType or ComplexDeviceDriverSw ComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=providedModeGroup.shortName, provided ModeGroup.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25

▽

△

Class	**BswModuleDescription**			
releasedTrigger	Trigger	*	aggr	A Trigger released by this module or cluster. It can be connected to the requiredTriggers of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with Triggers provided via ports by an associated ServiceSwComponentType, Ecu AbstractionSwComponentType or ComplexDeviceDriver SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=releasedTrigger.shortName, released Trigger.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=35
requiredClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module requires a client server entry which can be implemented on another partition or core.This entry is declared locally to this context and will be connected to the providedClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredClientServerEntry.shortName, requiredClientServerEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
requiredData	VariableDataPrototype	*	aggr	Specifies a data prototype required by this module in oder to be provided from another partition or core.The required Data is declared locally to this context and will be connected to the providedData of another or the same module via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredData.shortName, required Data.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
requiredMode Group	ModeDeclarationGroup Prototype	*	aggr	Specifies that this module or cluster depends on a certain mode group. The requiredModeGroup is local to this context and will be connected to the providedModeGroup of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredModeGroup.shortName, required ModeGroup.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=30
requiredTrigger	Trigger	*	aggr	Specifies that this module or cluster reacts upon an external trigger.This requiredTrigger is declared locally to this context and will be connected to the providedTrigger of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredTrigger.shortName, required Trigger.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

Table D.13: BswModuleDescription

Class	***BswModuleEntity*** (abstract)			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	Specifies the smallest code fragment which can be described for a BSW module or cluster within AUTOSAR.			
Base	*ARObject*, *ExecutableEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswCalledEntity, BswInterruptEntity, BswSchedulableEntity			
Aggregated by	BswInternalBehavior.entity			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
accessedModeGroup	ModeDeclarationGroupPrototype	*	ref	A mode group which is accessed via API call by this entity. It shall be a ModeDeclarationGroupPrototype required by this module or cluster. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=accessedModeGroup.modeDeclarationGroupPrototype, accessedModeGroup.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
activationPoint	BswInternalTriggeringPoint	*	ref	Activation point used by the module entity to activate one or more internal triggers. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=activationPoint.bswInternalTriggeringPoint, activationPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
callPoint	BswModuleCallPoint	*	aggr	A call point used in the code of this entity. The variability of this association is especially targeted at debug scenarios: It is possible to have one variant calling into the AUTOSAR debug module and another one which doesn't. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=callPoint.shortName, callPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataReceivePoint	BswVariableAccess	*	aggr	The data is received via the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReceivePoint.shortName, dataReceivePoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataSendPoint	BswVariableAccess	*	aggr	The data is sent via the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataSendPoint.shortName, dataSendPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
implementedEntry	BswModuleEntry	0..1	ref	The entry which is implemented by this module entity.
issuedTrigger	Trigger	*	ref	A trigger issued by this entity via BSW Scheduler API call. It shall be a BswTrigger released (i.e. owned) by this module or cluster. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=issuedTrigger.trigger, issuedTrigger.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

△

Class	***BswModuleEntity*** (abstract)			
managedMode Group	ModeDeclarationGroup Prototype	*	ref	A mode group which is managed by this entity. It shall be a ModeDeclarationGroupPrototype provided by this module or cluster. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=managedModeGroup.modeDeclaration GroupPrototype, managedModeGroup.variation Point.shortLabel vh.latestBindingTime=preCompileTime
schedulerName Prefix	BswSchedulerName Prefix	0..1	ref	A prefix to be used in generated names for the Bsw ModuleScheduler in the context of this BswModuleEntity, for example entry point prototypes, macros for dealing with exclusive areas, header file names. Details are defined in the SWS RTE. The prefix supersedes default rules for the prefix of those names.

Table D.14: BswModuleEntity

Class	**BswModuleEntry**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces			
Note	This class represents a single API entry (C-function prototype) into the BSW module or cluster. The name of the C-function is equal to the short name of this element with one exception: In case of multiple instances of a module on the same CPU, special rules for "infixes" apply, see description of class BswImplementation. **Tags:** atp.recommendedPackage=BswModuleEntrys			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument (ordered)	SwServiceArg	*	aggr	An argument belonging to this BswModuleEntry. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=argument.shortName, argument.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=45
bswEntryKind	BswEntryKindEnum	0..1	attr	This describes whether the entry is concrete or abstract. If the attribute is missing the entry is considered as concrete. **Tags:** xml.sequenceOffset=40
callType	BswCallType	0..1	attr	The type of call associated with this service. **Tags:** xml.sequenceOffset=25
execution Context	BswExecutionContext	0..1	attr	Specifies the execution context which is required (in case of entries into this module) or guaranteed (in case of entries called from this module) for this service. **Tags:** xml.sequenceOffset=30
function Prototype Emitter	NameToken	0..1	attr	This attribute is used to control the generation of function prototypes. If set to "RTE", the RTE generates the function prototypes in the Module Interlink Header File.

▽

△

Class	**BswModuleEntry**			
isReentrant	Boolean	0..1	attr	Reentrancy from the viewpoint of function callers: • true: Enables the service to be invoked again, before the service has finished. • false: It is prohibited to invoke the service again before is has finished. **Tags:** xml.sequenceOffset=15
isSynchronous	Boolean	0..1	attr	Synchronicity from the viewpoint of function callers: • true: This calls a synchronous service, i.e. the service is completed when the call returns. • false: The service (on semantical level) may not be complete when the call returns. **Tags:** xml.sequenceOffset=20
returnType	SwServiceArg	0..1	aggr	The return type belonging to this bswModuleEntry. **Tags:** xml.sequenceOffset=40
role	Identifier	0..1	attr	Specifies the role of the entry in the given context. It shall be equal to the standardized name of the service call, especially in cases where no ServiceIdentifier is specified, e.g. for callbacks. Note that the ShortName is not always sufficient because it maybe vendor specific (e.g. for callbacks which can have more than one instance). **Tags:** xml.sequenceOffset=10
serviceId	PositiveInteger	0..1	attr	Refers to the service identifier of the Standardized Interfaces of AUTOSAR basic software. For non-standardized interfaces, it can optionally be used for proprietary identification. **Tags:** xml.sequenceOffset=5
swServiceImpl Policy	SwServiceImplPolicy Enum	0..1	attr	Denotes the implementation policy as a standard function call, inline function or macro. This has to be specified on interface level because it determines the signature of the call. **Tags:** xml.sequenceOffset=35

Table D.15: BswModuleEntry

Class	**BswTimingEvent**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	A recurring BswEvent driven by a time period.			
Base	*ARObject*, *AbstractEvent*, *BswEvent*, *BswScheduleEvent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	BswInternalBehavior.event			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
period	TimeValue	0..1	attr	Requirement for the time period (in seconds) by which this event is triggered.

Table D.16: BswTimingEvent

Class	**ClientServerOperation**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An operation declared within the scope of a client/server interface.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationInterface.command, *AtpClassifier*.atpFeature, ClientServerInterface.operation, DiagnosticDataElementInterface.read, DiagnosticDataIdentifierInterface.read, DiagnosticDataIdentifierInterface.write, DiagnosticRoutineInterface.requestResult, DiagnosticRoutineInterface.start, DiagnosticRoutineInterface.stop, PhmRecoveryActionInterface.recovery, ServiceInterface.method			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this ClientServerOperation **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=argument.shortName, argument.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime
diagArgIntegrity	Boolean	0..1	attr	This attribute shall only be used in the implementation of diagnostic routines to support the case where input and output arguments are allocated in a shared buffer and might unintentionally overwrite input arguments by tentative write operations to output arguments. This situation can happen during sliced execution or while output parameters are arrays (call by reference). The value true means that the ClientServerOperation is aware of the usage of a shared buffer and takes precautions to avoid unintentional overwrite of input arguments. If the attribute does not exist or is set to false the Client ServerOperation does not have to consider the usage of a shared buffer.
possibleError	ApplicationError	*	ref	Possible errors that may by raised by the referring operation.

Table D.17: ClientServerOperation

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

Table D.18: ComplexDeviceDriverSwComponentType

Class	ComponentInCompositionInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs			
Note	The ComponentInCompositionInstanceRef points to a concrete SwComponentPrototype within a CompositionSwComponentType.			
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	DiagnosticJ1939SwMapping.swComponentPrototype, EOCEventRef.component, EOCExecutableEntityRef.component, ExecutionTimeConstraint.component, *TDEventSwc*.component, *TDEventVfb*.component			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
base	CompositionSwComponentType	0..1	ref	**Stereotypes:** atpDerived **Tags:** xml.sequenceOffset=10
context Component (ordered)	SwComponentPrototype	*	ref	The context for the scope of this timing event. **Tags:** xml.sequenceOffset=20
target Component	SwComponentPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30

Table D.19: ComponentInCompositionInstanceRef

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

▽

△

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

▽

△

Class	**CompositionSwComponentType**			
physical Dimension Mapping	PhysicalDimension MappingSet	0..1	ref	This reference identifies the `PhysicalDimensionMappingSet` that is applicable in the context of the enclosing `CompositionSwComponentType`. The `PhysicalDimensionMapping`s contained in the `PhysicalDimensionMappingSet` shall be taken into account for the assessment of the compatibility of `PhysicalDimension`s in the context of creation of a `PortInterfaceMapping` in the scope of the `CompositionSwComponentType`.

Table D.20: CompositionSwComponentType

Class	**CpSoftwareCluster**			
Package	M2::AUTOSARTemplates::SystemTemplate::SoftwareCluster			
Note	This meta class provides the ability to define a CP Software Cluster. Each CP Software Cluster can be integrated and build individually. It defines the sub-set of hierarchical tree(s) of Software Components belonging to this CP Software Cluster. Resources required or provided by this CP Software Cluster are given in the according mappings. **Tags:** atp.recommendedPackage=CpSoftwareClusters			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
softwareCluster Id	PositiveInteger	0..1	attr	This attribute represents the value of the id of the corresponding CP software cluster.
swComponent Assignment	SwComponent PrototypeAssignment	*	aggr	This is the collection of SwComponentPrototype Assignments **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swComponentAssignment, swComponent Assignment.variationPoint.shortLabel vh.latestBindingTime=postBuild
swComposition	CompositionSw ComponentType	*	ref	Software Components in the context of a CompositionSw ComponentType belonging to this CP Software Cluster. This reference can be used to describe the belonging SWCs when the CP Software Cluster is described out of the context of a System, e.g. reusable CP Software Cluster. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swComposition.compositionSwComponent Type, swComposition.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Table D.21: CpSoftwareCluster

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

Table D.22: EcuAbstractionSwComponentType

Class	EcucValueCollection			
Package	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
Note	This represents the anchor point of the ECU configuration description. **Tags:** atp.recommendedPackage=EcucValueCollections			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ecucValue	EcucModule ConfigurationValues	*	ref	References to the configuration of individual software modules that are present on this ECU. atpVariation: [RS_ECUC_00079] **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=ecucValue.ecucModuleConfigurationValues, ecucValue.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
ecuExtract	System	0..1	ref	Represents the extract of the System Configuration that is relevant for the ECU configured with that ECU Configuration Description.

Table D.23: EcucValueCollection

Class	***ExecutableEntity*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	Abstraction of executable code.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*BswModuleEntity*, RunnableEntity			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
activation Reason	ExecutableEntity ActivationReason	*	aggr	If the ExecutableEntity provides at least one activation Reason element the RTE resp. BSW Scheduler shall provide means to read the activation vector of this executable entity execution. If no activationReason element is provided the feature of being able to determine the activating RTEEvent is disabled for this ExecutableEntity.

▽

△

Class	***ExecutableEntity*** (abstract)			
canEnter	ExclusiveArea	*	ref	This means that the executable entity can enter/leave the referenced exclusive area through explicit API calls. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=canEnter.exclusiveArea, canEnter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
exclusiveAreaNestingOrder	ExclusiveAreaNestingOrder	*	ref	This represents the set of ExclusiveAreaNestingOrders recognized by this ExecutableEntity.
minimumStartInterval	TimeValue	0..1	attr	Specifies the time in seconds by which two consecutive starts of an ExecutableEntity are guaranteed to be separated.
reentrancyLevel	ReentrancyLevelEnum	0..1	attr	The reentrancy level of this ExecutableEntity. See the documentation of the enumeration type ReentrancyLevelEnum for details. Please note that nonReentrant interfaces can have also reentrant or multicoreReentrant implementations, and reentrant interfaces can also have multicoreReentrant implementations.
runsInside	ExclusiveArea	*	ref	The executable entity runs completely inside the referenced exclusive area. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=runsInside.exclusiveArea, runsInside.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this code entity. Via an association to the same SwAddrMethod, it can be specified that several code entities (even of different modules or components) shall be located in the same memory without already specifying the memory section itself.

Table D.24: ExecutableEntity

Primitive	**Float**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	An instance of Float is an element from the set of real numbers. **Tags:** xml.xsd.customType=FLOAT xml.xsd.type=double

Table D.25: Float

Class	<<atpMixedString>> ***FormulaExpression*** (abstract)
Package	M2::AUTOSARTemplates::GenericStructure::FormulaLanguage
Note	This class represents the syntax of the formula language. The class is modeled as an abstract class in order to be specialized into particular use cases. For each use case the referable objects might be specified in the specialization.
Base	*ARObject*
Subclasses	CompuGenericMath, EcucConditionFormula, EcucParameterDerivationFormula, *FMFormulaByFeaturesAndAttributes*, *SwSystemconstDependentFormula*, TDEventOccurrenceExpressionFormula, TimingConditionFormula

▽

△

Class	<<atpMixedString>> ***FormulaExpression*** (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
atpReference	Referrable	*	ref	The referable object shall yield a numerical / boolean value. **Stereotypes:** atpAbstract
atpString Reference	Referrable	*	ref	The referable object shall yield a string value. **Stereotypes:** atpAbstract

Table D.26: FormulaExpression

Class	***Frame*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Data frame which is sent over a communication medium. This element describes the pure Layout of a frame sent on a channel.			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	*AbstractEthernetFrame*, CanFrame, FlexrayFrame, *LinFrame*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
frameLength	Integer	0..1	attr	The used length (in bytes) of the referencing frame. Should not be confused with a static byte length reserved for each frame by some platforms (e.g. FlexRay). The frameLength of zero bytes is allowed. Please consider also TPS_SYST_02255.
pduToFrame Mapping	PduToFrameMapping	*	aggr	A frames layout as a sequence of Pdus. atpVariation: The content of a frame can be variable. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=pduToFrameMapping.shortName, pduToFrameMapping.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table D.27: Frame

Class	***FrameTriggering*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The FrameTriggering describes the instance of a frame sent on a channel and defines the manner of triggering (timing information) and identification of a frame on the channel, on which it is sent. For the same frame, if FrameTriggerings exist on more than one channel of the same cluster the fan-out/in is handled by the Bus interface.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CanFrameTriggering, EthernetFrameTriggering, FlexrayFrameTriggering, LinFrameTriggering			
Aggregated by	*PhysicalChannel*.frameTriggering			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
frame	Frame	0..1	ref	One frame can be triggered several times, e.g. on different channels. If a frame has no frame triggering, it won't be sent at all. A frame triggering has assigned exactly one frame, which it triggers.

▽

△

Class	***FrameTriggering*** (abstract)			
framePort	FramePort	*	ref	References to the FramePort on every ECU of the system which sends and/or receives the frame. References for both the sender and the receiver side shall be included when the system is completely defined.
pduTriggering	PduTriggering	*	ref	This reference provides the relationship to the Pdu Triggerings that are implemented by the FrameTriggering. The reference is optional since no PduTriggering can be defined for NmPdus and XCP Pdus. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=pduTriggering.pduTriggering, pdu Triggering.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table D.28: FrameTriggering

Class	**GlobalTimeDomain**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the ability to define a global time domain. **Tags:** atp.recommendedPackage=GlobalTimeDomains			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
debounceTime	TimeValue	0..1	attr	Defines the minimum amount of time between two time sync messages are transmitted.
domainId	PositiveInteger	0..1	attr	This represents the ID of the GlobalTimeDomain used in the network messages sent on behalf of global time management.
gateway	GlobalTimeGateway	*	aggr	A GlobalTimeGateway may exist in the context of a GlobalTimeDomain to actively update the global time information as it is routed from one GlobalTimeDomain to another. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=gateway.shortName, gateway.variation Point.shortLabel vh.latestBindingTime=postBuild
globalTime CorrectionProps	GlobalTimeCorrection Props	0..1	aggr	Defintion of attributes for rate and offset correction.
globalTime Domain Property	AbstractGlobalTime DomainProps	0..1	aggr	Additional properties of the GlobalTimeDomain. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=globalTimeDomainProperty, globalTime DomainProperty.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	**GlobalTimeDomain**			
globalTime Master	GlobalTimeMaster	0..1	aggr	This represents the single master of a GlobalTime Domain. A GlobalTimeDomain may have no GlobalTime Domain.master, e.g. when it gets its time from a GPS receiver. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=globalTimeMaster.shortName, globalTime Master.variationPoint.shortLabel vh.latestBindingTime=postBuild
globalTimeSub Domain	GlobalTimeDomain	*	ref	By this means it is possible to create a hierarchy of sub Domains where one global time domain can declare one or more other global time domains as its subDomains. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=globalTimeSubDomain.globalTimeDomain, globalTimeSubDomain.variationPoint.shortLabel vh.latestBindingTime=postBuild
network SegmentId	NetworkSegment Identification	0..1	aggr	Defines the numerical identification of a GlobalTime sub domain.
offsetTime Domain	GlobalTimeDomain	0..1	ref	Reference to a synchronized time domain this offset time domain is based on. The reference source is the offset time domain. The reference target is the synchronized time domain.
pduTriggering	PduTriggering	0..1	ref	This PduTriggering will be taken to transmit the global time information from a GlobalTimeMaster to a the associated GlobalTimeSlaves. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=pduTriggering.pduTriggering, pdu Triggering.variationPoint.shortLabel vh.latestBindingTime=postBuild
slave	GlobalTimeSlave	*	aggr	This represents the collections of slaves of the Global TimeDomain. A GlobalTimeDomain may have no Global TimeDomain.slaves, e.g. when it propagates its time directly to sub domains. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=slave.shortName, slave.variationPoint.short Label vh.latestBindingTime=postBuild
syncLoss Timeout	TimeValue	0..1	attr	This attribute describes the timeout for the situation that the time synchronization gets lost in the scope of the time domain.

Table D.29: GlobalTimeDomain

Class	***IPdu*** (abstract)
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication
Note	The IPdu (Interaction Layer Protocol Data Unit) element is used to sum up all Pdus that are routed by the PduR.
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*
Subclasses	ContainerIPdu, DcmIPdu, GeneralPurposeIPdu, ISignalIPdu, J1939DcmIPdu, MultiplexedIPdu, NPdu, SecuredIPdu, UserDefinedIPdu
Aggregated by	ARPackage.element

▽

△

Class	***IPdu*** (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
containedIPdu Props	ContainedIPduProps	0..1	aggr	Defines whether this IPdu may be collected inside a ContainerIPdu.

Table D.30: IPdu

Class	**ISignal**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Signal of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal is sent in different SignalIPdus to multiple receivers. To support the RTE "signal fan-out" each SignalIPdu contains ISignals. If the same System Signal is to be mapped into several SignalIPdus there is one ISignal needed for each ISignalToIPduMapping. ISignals describe the Interface between the Precompile configured RTE and the potentially Postbuild configured Com Stack (see ECUC Parameter Mapping). In case of the SystemSignalGroup an ISignal shall be created for each SystemSignal contained in the SystemSignalGroup. **Tags:** atp.recommendedPackage=ISignals			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
data Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignal. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataTransformation.dataTransformation, dataTransformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
dataTypePolicy	DataTypePolicyEnum	0..1	attr	With the aggregation of SwDataDefProps an ISignal specifies how it is represented on the network. This representation follows a particular policy. Note that this causes some redundancy which is intended and can be used to support flexible development methodology as well as subsequent integrity checks. If the policy "networkRepresentationFromComSpec" is chosen the network representation from the ComSpec that is aggregated by the PortPrototype shall be used. If the "override" policy is chosen the requirements specified in the PortInterface and in the ComSpec are not fulfilled by the networkRepresentationProps. In case the System Description doesn't use a complete Software Component Description (VFB View) the "legacy" policy can be chosen.

▽

△

Class	**ISignal**			
initValue	ValueSpecification	0..1	aggr	Optional definition of a ISignal's initValue in case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. This value can be used to configure the Signal's "Init Value". If a full DataMapping exist for the SystemSignal this information may be available from a configured Sender ComSpec and ReceiverComSpec. In this case the initvalues in SenderComSpec and/or ReceiverComSpec override this optional value specification. Further restrictions apply from the RTE specification.
iSignalProps	ISignalProps	0..1	aggr	Additional optional ISignal properties that may be stored in different files. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=iSignalProps
iSignalType	ISignalTypeEnum	0..1	attr	This attribute defines whether this iSignal is an array that results in a UINT8_N / UINT8_DYN ComSignalType in the COM configuration or a primitive type.
length	UnlimitedInteger	0..1	attr	Size of the signal in bits. The size needs to be derived from the mapped VariableDataPrototype according to the mapping of primitive DataTypes to BaseTypes as used in the RTE. Indicates maximum size for dynamic length signals. The ISignal length of zero bits is allowed.
network Representation Props	SwDataDefProps	0..1	aggr	Specification of the actual network representation. The usage of SwDataDefProps for this purpose is restricted to the attributes compuMethod and baseType. The optional baseType attributes "memAllignment" and "byteOrder" shall not be used. The attribute "dataTypePolicy" in the SystemTemplate element defines whether this network representation shall be ignored and the information shall be taken over from the network representation of the ComSpec. If "override" is chosen by the system integrator the network representation can violate against the requirements defined in the PortInterface and in the network representation of the ComSpec. In case that the System Description doesn't use a complete Software Component Description (VFB View) this element is used to configure "ComSignalDataInvalid Value" and the Data Semantics. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=networkRepresentationProps
systemSignal	SystemSignal	0..1	ref	Reference to the System Signal that is supposed to be transmitted in the ISignal.
timeout Substitution Value	ValueSpecification	0..1	aggr	Defines and enables the ComTimeoutSubstituition for this ISignal.

▽

△

Class	**ISignal**			
transformation ISignalProps	TransformationISignal Props	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignal specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignals are described in the TransformationTechnology class. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=transformationISignalProps

Table D.31: ISignal

Class	**ISignalTriggering**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	A ISignalTriggering allows an assignment of ISignals to physical channels.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*PhysicalChannel*.iSignalTriggering			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iSignal	ISignal	0..1	ref	This reference shall be used if an ISignal is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignalTriggering-ISignalGroup reference.
iSignalGroup	ISignalGroup	0..1	ref	This reference shall be used if an ISignalGroup is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignal Triggering-ISignal reference.
iSignalPort	ISignalPort	*	ref	References to the ISignalPort on every ECU of the system which sends and/or receives the ISignal. References for both the sender and the receiver side shall be included when the system is completely defined.

Table D.32: ISignalTriggering

Class	***Identifiable*** (abstract)
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**Identifiable** (abstract)			
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstanceFilter*, *AbstractServiceInstance*, AppOsTask ProxyToEcuTaskProxyMapping, ApplicationEndpoint, ApplicationError, ApplicationPartitionToEcuPartition Mapping, AppliedStandard, AsynchronousServerCallResultPoint, *AtpBlueprint*, *AtpBlueprintable*, *Atp Classifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BinaryManifest AddressableObject*, BinaryManifestItemDefinition, *BinaryManifestResource*, BinaryManifestResource Definition, BlockState, BswInternalTriggeringPoint, BswModuleDependency, *BuildActionEntity*, Build ActionEnvironment, CanTpAddress, CanTpChannel, CanTpNode, Chapter, ClassContentConditional, ClientIdDefinition, ClientServerOperation, Code, *CollectableElement*, ComManagementMapping, *Comm ConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, *CouplingElementAbstractDetails*, CouplingPort, *CouplingPortAbstractShaper*, *CouplingPortStructuralElement*, *CpSoftwareClusterResource*, CpSoftwareClusterResourceToApplication PartitionMapping, CpSoftwareClusterToApplicationPartitionMapping, CpSoftwareClusterToEcuInstance Mapping, CpSoftwareClusterToResourceMapping, *CryptoServiceMapping*, DataPrototypeGroup, Data Transformation, DdsCpDomain, DdsCpPartition, DdsCpQosProfile, DdsCpTopic, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnected Indicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, DiagnosticFunctionInhibit Source, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DltApplication, DltArgument, Dlt LogChannel, DltMessage, DoIpInterface, DoIpLogicAddress, DoIpRoutingActivation, ECUMapping, *EOC ExecutableEntityRefAbstract*, EcuPartition, EcucContainerValue, *EcucDefinitionElement*, Ecuc DestinationUriDef, EcucEnumerationLiteralDef, EcucQuery, EcucValidationCondition, EndToEnd Protection, EthernetWakeupSleepOnDatalineConfig, EventHandler, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMap Element, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FlatInstanceDescriptor, FlexrayArTpNode, FlexrayTpConnectionControl, FlexrayTpNode, FlexrayTpPduPool, *FrameTriggering*, GeneralParameter, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HeapUsage*, HwAttribute Def, HwAttributeLiteralDef, HwPin, HwPinGroup, *IEEE1722TpAcfBus*, *IEEE1722TpAcfBusPart*, IPSec Rule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignalTriggering, *IdentCaption*, ImpositionTime, InternalTriggeringPoint, J1939SharedAddressCluster, J1939TpNode, Keyword, LifeCycleState, Lin ScheduleTable, LinTpNode, Linker, MacMulticastGroup, MacSecKayParticipant, McDataInstance, MemorySection, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, NetworkEndpoint, *Nm Cluster*, NmEcu, *NmNode*, NvBlockDescriptor, *PackageableElement*, ParameterAccess, PduActivation RoutingGroup, PduToFrameMapping, PduTriggering, PerInstanceMemory, *PhysicalChannel*, Port ElementToCommunicationResourceMapping, PortGroup, *PortInterfaceMapping*, PossibleErrorReaction, ResourceConsumption, RootSwCompositionPrototype, RptComponent, RptContainer, RptExecutable Entity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RteEventIn CompositionSeparation, RteEventInCompositionToOsTaskProxyMapping, RteEventInSystemSeparation, RteEventInSystemToOsTaskProxyMapping, RunnableEntityGroup, *SdgAttribute*, SdgClass, Secure CommunicationAuthenticationProps, SecureCommunicationFreshnessProps, SecurityEventContext Props, *ServerCallPoint*, *ServiceNeeds*, SignalServiceTranslationElementProps, SignalServiceTranslation EventProps, SignalServiceTranslationProps, SocketAddress, SomeipTpChannel, *SpecElement Reference*, *StackUsage*, StaticSocketConnection, StructuredReq, SwGenericAxisParamType, SwService Arg, SwcServiceDependency, SwcToApplicationPartitionMapping, SwcToEcuMapping, SwcToImpl Mapping, SwitchAsynchronousTrafficShaperGroupEntry, SwitchFlowMeteringEntry, SwitchStreamFilter ActionDestPortModification, SwitchStreamFilterEntry, SwitchStreamFilterRule, SwitchStreamGateEntry, SwitchStreamIdentification, SystemMapping, SystemSignalGroupToCommunicationResourceMapping, SystemSignalToCommunicationResourceMapping, TDCpSoftwareClusterMapping, TDCpSoftware ClusterResourceMapping, TcpOptionFilterList, *TimingClock*, TimingClockSyncAccuracy, Timing Condition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, TimingModeInstance, Tls CryptoCipherSuite, TlsCryptoCipherSuiteProps, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, *TransformationProps*, TransformationTechnology, Trigger, VariableAccess, VariationPoint Proxy, ViewMap, VlanConfig, WaitPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=adminData xml.sequenceOffset=-40

▽

△

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

Table D.33: Identifiable

Class	***InternalBehavior*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	Common base class (abstract) for the internal behavior of both software components and basic software modules/clusters.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswInternalBehavior, SwcInternalBehavior			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	***InternalBehavior*** (abstract)			
constant Memory	ParameterData Prototype	*	aggr	Describes a read only memory object containing characteristic value(s) implemented by this Internal Behavior. The shortName of ParameterDataPrototype has to be equal to the ”C’ identifier of the described constant. The characteristic value(s) might be shared between Sw ComponentPrototypes of the same SwComponentType. The aggregation of constantMemory is subject to variability with the purpose to support variability in the software component or module implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=constantMemory.shortName, constant Memory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for the particular InternalBehavior **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the particular InternalBehavior **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dataTypeMapping
exclusiveArea	ExclusiveArea	*	aggr	This specifies an ExclusiveArea for this InternalBehavior. The exclusiveArea is local to the component resp. module. The aggregation of ExclusiveAreas is subject to variability. Note: the number of ExclusiveAreas might vary due to the conditional existence of RunnableEntities or BswModuleEntities. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveArea.shortName, exclusive Area.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
exclusiveArea NestingOrder	ExclusiveAreaNesting Order	*	aggr	This represents the set of ExclusiveAreaNestingOrder owned by the InternalBehavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveAreaNestingOrder.shortName, exclusiveAreaNestingOrder.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	***InternalBehavior*** (abstract)			
staticMemory	VariableDataPrototype	*	aggr	Describes a read and writeable static memory object representing measurerment variables implemented by this software component. The term "static" is used in the meaning of "non-temporary" and does not necessarily specify a linker encapsulation. This kind of memory is only supported if supportsMultipleInstantiation is FALSE. The shortName of the VariableDataPrototype has to be equal with the ”C’ identifier of the described variable. The aggregation of staticMemory is subject to variability with the purpose to support variability in the software component’s implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=staticMemory.shortName, staticMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table D.34: InternalBehavior

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

Table D.35: ModeDeclaration

Class	**ModeDeclarationGroupPrototype**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, BswModuleDescription.providedModeGroup, BswModuleDescription.requiredModeGroup, FirewallStateSwitchInterface.firewallStateMachine, FunctionGroupSet.functionGroup, ModeSwitchInterface.modeGroup, Process.processStateMachine, StateManagementStateNotification.stateMachine			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swCalibrationAccess	SwCalibrationAccessEnum	0..1	attr	This allows for specifying whether or not the enclosing ModeDeclarationGroupPrototype can be measured at run-time.
type	ModeDeclarationGroup	0..1	tref	The "collection of ModeDeclarations" (= ModeDeclarationGroup) supported by a component **Stereotypes:** isOfType

Table D.36: ModeDeclarationGroupPrototype

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

Table D.37: Numerical

Class	**PPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port providing a certain port interface.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provided Interface	PortInterface	0..1	tref	The interface that this port provides. **Stereotypes:** isOfType

Table D.38: PPortPrototype

Class	**PduTriggering**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The PduTriggering describes on which channel the IPdu is transmitted. The Pdu routing by the PduR is only allowed for subclasses of IPdu. Depending on its relation to entities such channels and clusters it can be unambiguously deduced whether a fan-out is handled by the Pdu router or the Bus Interface. If the fan-out is specified between different clusters it shall be handled by the Pdu Router. If the fan-out is specified between different channels of the same cluster it shall be handled by the Bus Interface.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*PhysicalChannel*.pduTriggering			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iPdu	Pdu	0..1	ref	Reference to the Pdu for which the PduTriggering is defined. One I-Pdu can be triggered on different channels (PduR fan-out). The Pdu routing by the PduR is only allowed for subclasses of IPdu. Nevertheless is the reference to the Pdu element necessary since the PduTriggering element is also used to specify the sending and receiving connections to Ecu Ports.
iPduPort	IPduPort	*	ref	References to the IPduPort on every ECU of the system which sends and/or receives the I-PDU. References for both the sender and the receiver side shall be included when the system is completely defined.

△

Class	PduTriggering			
iSignal Triggering	ISignalTriggering	*	ref	This reference provides the relationship to the ISignal Triggerings that are implemented by the PduTriggering. The reference is optional since no ISignalTriggering can be defined for DCM and Multiplexed Pdus. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=iSignalTriggering.iSignalTriggering, iSignal Triggering.variationPoint.shortLabel vh.latestBindingTime=postBuild
secOcCrypto Mapping	SecOcCryptoService Mapping	0..1	ref	This reference identifies the crypto profile applicable to the usage (send, receive) of the also referenced Secured IPdu. Obviously, this reference is only applicable if the Pdutriggering also references a SecuredIPdu in the role i Pdu.
triggerIPduSend Condition	TriggerIPduSend Condition	*	aggr	Defines the trigger for the Com_TriggerIPDUSend API call. Only if all defined TriggerIPduSendConditions evaluate to true (AND associated) the Com_Trigger IPDUSend API shall be called.

Table D.39: PduTriggering

Class	*PhysicalChannel* (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	A physical channel is the transmission medium that is used to send and receive information between communicating ECUs. Each CommunicationCluster has at least one physical channel. Bus systems like CAN and LIN only have exactly one PhysicalChannel. A FlexRay cluster may have more than one PhysicalChannels that may be used in parallel for redundant communication. An ECU is part of a cluster if it contains at least one controller that is connected to at least one channel of the cluster.#			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanPhysicalChannel*, EthernetPhysicalChannel, FlexrayPhysicalChannel, LinPhysicalChannel, UserDefinedPhysicalChannel			
Aggregated by	*CommunicationCluster*.physicalChannel			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
comm Connector	Communication Connector	*	ref	Reference to the ECUInstance via a Communication Connector to which the channel is connected. atpVariation: Variable assignment of Physical Channels to different CommunicationConnectors is expressed with this variation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=commConnector.communicationConnector, commConnector.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	PhysicalChannel (abstract)			
frameTriggering	FrameTriggering	*	aggr	One frame triggering is defined for exactly one channel. Channels may have assigned an arbitrary number of frame triggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings shall be variable, too. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=frameTriggering.shortName, frame Triggering.variationPoint.shortLabel vh.latestBindingTime=postBuild
iSignal Triggering	ISignalTriggering	*	aggr	One ISignalTriggering is defined for exactly one channel. Channels may have assigned an arbitrary number of ISignaltriggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings shall be variable, too. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=iSignalTriggering.shortName, iSignal Triggering.variationPoint.shortLabel vh.latestBindingTime=postBuild
managed Physical Channel	PhysicalChannel	*	ref	Reference between a channel with role managing channel and a channel with role managed channel.
pduTriggering	PduTriggering	*	aggr	One PduTriggering is defined for exactly one channel. Channels may have assigned an arbitrary number of I-Pdu triggerings. atpVariation: If signals/PDUs/frames are variable, the corresponding triggerings shall be variable, too. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=pduTriggering.shortName, pdu Triggering.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table D.40: PhysicalChannel

Class	PortPrototype (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/ server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.

▽

△

Class	***PortPrototype*** (abstract)			
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table D.41: PortPrototype

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

Table D.42: PortPrototypeBlueprint

Class	**RPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
mayBe Unconnected	Boolean	0..1	attr	If set to true, this attribute indicates that the enclosing RPortPrototype may be left unconnected and that this aspect has explicitly been considered in the software-component's design.
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType

Table D.43: RPortPrototype

Class	**RTEEvent** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	Abstract base class for all RTE-related events			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AsynchronousServerCallReturnsEvent, BackgroundEvent, DataReceiveErrorEvent, DataReceivedEvent, DataSendCompletedEvent, DataWriteCompletedEvent, ExternalTriggerOccurredEvent, InitEvent, InternalTriggerOccurredEvent, ModeSwitchedAckEvent, OperationInvokedEvent, OsTaskExecutionEvent, SwcModeManagerErrorEvent, SwcModeSwitchEvent, TimingEvent, TransformerHardErrorEvent			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
disabledMode	ModeDeclaration	*	iref	Reference to the Modes that disable the Event. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=disabledMode.contextPort, disabledMode.contextModeDeclarationGroupPrototype, disabledMode.targetModeDeclaration **InstanceRef implemented by:** RModeInAtomicSwcInstanceRef
startOnEvent	RunnableEntity	0..1	ref	The referenced RunnableEntity starts when the corresponding RTEEvent is raised.

Table D.44: RTEEvent

Class	**Referrable** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	*ARObject*			
Subclasses	*AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, BswVariableAccess, CouplingPortTrafficClassAssignment, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, ExclusiveAreaNestingOrder, *HwDescriptionEntity*, *ImplementationProps*, LinSlaveConfigIdent, ModeTransition, *MultilanguageReferrable*, PncMappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, SocketConnectionBundle, TimeSyncServerConfiguration, TpConnectionIdent			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortNameFragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** xml.sequenceOffset=-90

Table D.45: Referrable

Class	**ResourceConsumption**			
Package	M2::AUTOSARTemplates::CommonStructure::ResourceConsumption			
Note	Description of consumed resources by one implementation of a software.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuResourceEstimation.bswResourceEstimation, EcuResourceEstimation.rteResourceEstimation, *Implementation*.resourceConsumption, StateDependentStartupConfig.resourceConsumption			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**ResourceConsumption**			
accessCount Set	AccessCountSet	*	aggr	Set of access count values **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=accessCountSet, accessCountSet.variation Point.shortLabel vh.latestBindingTime=preCompileTime
executionTime	ExecutionTime	*	aggr	Collection of the execution time descriptions for this implementation. The aggregation of executionTime is subject to variability with the purpose to support the conditional existence of runnable entities. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=executionTime.shortName, execution Time.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
heapUsage	HeapUsage	*	aggr	Collection of the heap memory allocated by this implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=heapUsage.shortName, heap Usage.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
memorySection	MemorySection	*	aggr	An abstract memory section required by this Implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=memorySection.shortName, memory Section.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
sectionName Prefix	SectionNamePrefix	*	aggr	A prefix to be used for the memory section symbol in the code. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=sectionNamePrefix.shortName, section NamePrefix.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
stackUsage	StackUsage	*	aggr	Collection of the stack memory usage for each runnable entity of this implementation. The aggregation of Stack Usage is subject to variability with the purpose to support the conditional existence of runnable entities. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=stackUsage.shortName, stack Usage.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table D.46: ResourceConsumption

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

Table D.47: RootSwCompositionPrototype

Class	**RunnableEntity**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
Note	A RunnableEntity represents the smallest code-fragment that is provided by an AtomicSwComponent Type and are executed under control of the RTE. RunnableEntities are for instance set up to respond to data reception or operation invocation on a server.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *ExecutableEntity*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.runnable			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument (ordered)	RunnableEntity Argument	*	aggr	This represents the formal definition of a an argument to a RunnableEntity.

▽

△

Class	**RunnableEntity**			
asynchronous ServerCall ResultPoint	AsynchronousServer CallResultPoint	*	aggr	The server call result point admits a runnable to fetch the result of an asynchronous server call. The aggregation of AsynchronousServerCallResultPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes and the variant existence of server call result points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=asynchronousServerCallResultPoint.shortName, asynchronousServerCallResultPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
canBeInvoked Concurrently	Boolean	0..1	attr	If the value of this attribute is set to "true" the enclosing RunnableEntity can be invoked concurrently (even for one instance of the corresponding AtomicSwComponent Type). This implies that it is the responsibility of the implementation of the RunnableEntity to take care of this form of concurrency.
dataRead Access	VariableAccess	*	aggr	RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataReadAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataReadAccess in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReadAccess.shortName, dataReadAccess.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataReceive PointBy Argument	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of an argument in the function signature. The aggregation of dataReceivePointByArgument is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data receive points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReceivePointByArgument.shortName, dataReceivePointByArgument.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**RunnableEntity**			
dataReceive PointByValue	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of the return value. The aggregation of dataReceivePointBy Value is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of data receive points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReceivePointByValue.shortName, data ReceivePointByValue.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataSendPoint	VariableAccess	*	aggr	RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataSendPoint is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data send points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataSendPoint.shortName, dataSend Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataWrite Access	VariableAccess	*	aggr	RunnableEntity has implicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataWriteAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataWriteAccess in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataWriteAccess.shortName, dataWrite Access.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
external TriggeringPoint	ExternalTriggeringPoint	*	aggr	The aggregation of ExternalTriggeringPoint is subject to variability with the purpose to support the conditional existence of trigger ports or the variant existence of external triggering points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=externalTriggeringPoint.ident.shortName, externalTriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
internal TriggeringPoint	InternalTriggeringPoint	*	aggr	The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalTriggeringPoint.shortName, internal TriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**RunnableEntity**			
modeAccess Point	ModeAccessPoint	*	aggr	The runnable has a mode access point. The aggregation of ModeAccessPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode access points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeAccessPoint.ident.shortName, mode AccessPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
modeSwitch Point	ModeSwitchPoint	*	aggr	The runnable has a mode switch point. The aggregation of ModeSwitchPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode switch points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeSwitchPoint.shortName, modeSwitch Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
parameter Access	ParameterAccess	*	aggr	The presence of a ParameterAccess implies that a RunnableEntity needs read only access to a Parameter DataPrototype which may either be local or within a Port Prototype. The aggregation of ParameterAccess is subject to variability with the purpose to support the conditional existence of parameter ports and component local parameters as well as the variant existence of Parameter Access (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=parameterAccess.shortName, parameter Access.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
readLocal Variable	VariableAccess	*	aggr	The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of read LocalVariable (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=readLocalVariable.shortName, readLocal Variable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
serverCallPoint	ServerCallPoint	*	aggr	The RunnableEntity has a ServerCallPoint. The aggregation of ServerCallPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes or the variant existence of server call points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serverCallPoint.shortName, serverCall Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	RunnableEntity			
symbol	CIdentifier	0..1	attr	The symbol describing this RunnableEntity's entry point. This is considered the API of the RunnableEntity and is required during the RTE contract phase.
waitPoint	WaitPoint	*	aggr	The WaitPoint associated with the RunnableEntity.
writtenLocal Variable	VariableAccess	*	aggr	The presence of a writtenLocalVariable implies that a RunnableEntity needs write access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of writtenLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of written LocalVariable (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=writtenLocalVariable.shortName, written LocalVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table D.48: RunnableEntity

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

Table D.49: SenderReceiverInterface

Class	SensorActuatorSwComponentType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The SensorActuatorSwComponentType introduces the possibility to link from the software representation of a sensor/actuator to its hardware description provided by the ECU Resource Template. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
sensorActuator	HwDescriptionEntity	0..1	ref	Reference from the Sensor Actuator Software Component Type to the description of the actual hardware.

Table D.50: SensorActuatorSwComponentType

Class	SwComponentPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	Role of a software component within a composition.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.component			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
type	SwComponentType	0..1	tref	Type of the instance. **Stereotypes:** isOfType

Table D.51: SwComponentPrototype

Class	*SwComponentType* (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for AUTOSAR software components.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AtomicSwComponentType*, CompositionSwComponentType, ParameterSwComponentType			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
consistency Needs	ConsistencyNeeds	*	aggr	This represents the collection of ConsistencyNeeds owned by the enclosing SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=consistencyNeeds.shortName, consistency Needs.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
port	PortPrototype	*	aggr	The PortPrototypes through which this SwComponent Type can communicate. The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=port.shortName, port.variationPoint.short Label vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=portGroup.shortName, portGroup.variation Point.shortLabel vh.latestBindingTime=preCompileTime
swcMapping Constraint	SwComponentMapping Constraints	*	ref	Reference to constraints that are valid for this Sw ComponentType.
swComponent Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swComponentDocumentation, sw ComponentDocumentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10
unitGroup	UnitGroup	*	ref	This allows for the specification of which UnitGroups are relevant in the context of referencing SwComponentType.

Table D.52: SwComponentType

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

Table D.53: SwSystemconstantValueSet

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

△

Class	**SwcInternalBehavior**			
exclusiveArea Policy	SwcExclusiveArea Policy	*	aggr	Options how to generate the ExclusiveArea related APIs. When no SwcExclusiveAreaPolicy is specified for an ExclusiveArea the default values apply. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveAreaPolicy, exclusiveArea Policy.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
explicitInter Runnable Variable	VariableDataPrototype	*	aggr	Implement state message semantics for establishing communication among runnables of the same component. The aggregation of explicitInterRunnable Variable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=explicitInterRunnableVariable.shortName, explicitInterRunnableVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
implicitInter Runnable Variable	VariableDataPrototype	*	aggr	Implement state message semantics for establishing communication among runnables of the same component. The aggregation of implicitInterRunnable Variable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=implicitInterRunnableVariable.shortName, implicitInterRunnableVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
includedData TypeSet	IncludedDataTypeSet	*	aggr	The includedDataTypeSet is used by a software component for its implementation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=includedDataTypeSet
includedMode Declaration GroupSet	IncludedMode DeclarationGroupSet	*	aggr	This aggregation represents the included Mode DeclarationGroups **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=includedModeDeclarationGroupSet
instantiation DataDefProps	InstantiationDataDef Props	*	aggr	The purpose of this is that within the context of a given SwComponentType some data def properties of individual instantiations can be modified. The aggregation of InstantiationDataDefProps is subject to variability with the purpose to support the conditional existence of Port Prototypes and component local memories like "per InstanceParameter" or "arTypedPerInstanceMemory". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=instantiationDataDefProps, instantiationData DefProps.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**SwcInternalBehavior**			
perInstance Memory	PerInstanceMemory	*	aggr	Defines a per-instance memory object needed by this software component. The aggregation of PerInstance Memory is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=perInstanceMemory.shortName, perInstance Memory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
perInstance Parameter	ParameterData Prototype	*	aggr	Defines parameter(s) or characteristic value(s) that needs to be available for each instance of the software-component. This is typically only useful if supportsMultipleInstantiation is set to "true". The aggregation of perInstanceParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=perInstanceParameter.shortName, per InstanceParameter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
portAPIOption	PortAPIOption	*	aggr	Options for generating the signature of port-related calls from a runnable to the RTE and vice versa. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=portAPIOption, portAPIOption.variation Point.shortLabel vh.latestBindingTime=preCompileTime
runnable	RunnableEntity	*	aggr	This is a RunnableEntity specified for the particular Swc InternalBehavior. The aggregation of RunnableEntity is subject to variability with the purpose to support the conditional existence of RunnableEntities. Note: the number of RunnableEntities might vary due to the conditional existence of Port Prototypes using DataReceivedEvents or due to different scheduling needs of algorithms. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=runnable.shortName, runnable.variation Point.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**SwcInternalBehavior**			
service Dependency	SwcService Dependency	*	aggr	Defines the requirements on AUTOSAR Services for a particular item. The aggregation of SwcServiceDependency is subject to variability with the purpose to support the conditional existence of ports as well as the conditional existence of ServiceNeeds. The SwcServiceDependency owned by an SwcInternal Behavior can be located in a different physical file in order to support that SwcServiceDependency might be provided in later development steps or even by different expert domain (e.g OBD expert for Obd related Service Needs) tools. Therefore the aggregation is <<atp Splitable>>. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceDependency.shortName, service Dependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
shared Parameter	ParameterData Prototype	*	aggr	Defines parameter(s) or characteristic value(s) shared between SwComponentPrototypes of the same Sw ComponentType The aggregation of sharedParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=sharedParameter.shortName, shared Parameter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
supports Multiple Instantiation	Boolean	0..1	attr	Indicate whether the corresponding software-component can be multiply instantiated on one ECU. In this case the attribute will result in an appropriate component API on programming language level (with or without instance handle).
variationPoint Proxy	VariationPointProxy	*	aggr	Proxy of a variation points in the C/C++ implementation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=variationPointProxy.shortName

Table D.54: SwcInternalBehavior

Class	**System**			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The top level element of the System Description. The System description defines five major elements: Topology, Software, Communication, Mapping and Mapping Constraints. The System element directly aggregates the elements describing the Software, Mapping and Mapping Constraints; it contains a reference to an ASAM FIBEX description specifying Communication and Topology. **Tags:** atp.recommendedPackage=Systems			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**System**			
clientId DefinitionSet	ClientIdDefinitionSet	*	ref	Set of Client Identifiers that are used for inter-ECU client-server communication in the System.
containerIPdu HeaderByte Order	ByteOrderEnum	0..1	attr	Defines the byteOrder of the header in ContainerIPdus.
ecuExtract Version	RevisionLabelString	0..1	attr	Version number of the Ecu Extract.
fibexElement	FibexElement	*	ref	Reference to ASAM FIBEX elements specifying Communication and Topology. All Fibex Elements used within a System Description shall be referenced from the System Element. atpVariation: In order to describe a product-line, all Fibex Elements can be optional. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=fibexElement.fibexElement, fibex Element.variationPoint.shortLabel vh.latestBindingTime=postBuild
interpolation Routine MappingSet	InterpolationRoutine MappingSet	*	ref	This reference identifies the InterpolationRoutineMapping Sets that are relevant in the context of the enclosing System.
j1939Shared AddressCluster	J1939SharedAddress Cluster	*	aggr	Collection of J1939Clusters that share a common address space for the routing of messages. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=j1939SharedAddressCluster.shortName, j1939SharedAddressCluster.variationPoint.shortLabel vh.latestBindingTime=postBuild
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects (mapping of SW components to ECUs, mapping of data elements to signals, and mapping constraints). In order to support OEM / Tier 1 interaction and shared development for one common System this aggregation is atpSplitable and atpVariation. The content of System Mapping can be provided by several parties using different names for the SystemMapping. This element is not required when the System description is used for a network-only use-case. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=mapping.shortName, mapping.variation Point.shortLabel vh.latestBindingTime=postBuild
pncVector Length	PositiveInteger	0..1	attr	Length of the partial networking request release information vector (in bytes).
pncVectorOffset	PositiveInteger	0..1	attr	Absolute offset (with respect to the NM-PDU) of the partial networking request release information vector that is defined in bytes as an index starting with 0.

▽

△

Class	System			
rootSoftware Composition	RootSwComposition Prototype	0..1	aggr	Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case. atpVariation: The RootSwCompositionPrototype can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=rootSoftwareComposition.shortName, root SoftwareComposition.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
swCluster	CpSoftwareCluster	*	ref	CP Software Clusters of this System **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swCluster.cpSoftwareCluster, sw Cluster.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
system Documentation	Chapter	*	aggr	Possibility to provide additional documentation while defining the System. The System documentation can be composed of several chapters. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=systemDocumentation.shortName, system Documentation.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=-10
systemVersion	RevisionLabelString	0..1	attr	Version number of the System Description.

Table D.55: System

Class	*TDEventBsw* (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventBsw			
Note	This is used to describe timing events related to BSW modules.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	TDEventBswModeDeclaration, TDEventBswModule			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
bswModule Description	BswModuleDescription	0..1	ref	The scope of this timing event.

Table D.56: TDEventBsw

Class	*TDEventSLLET* (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventSLLET			
Note	Used to describe SL-LET (System-Level) timing events. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	TDEventSLLETPort			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.57: TDEventSLLET

Class	**TDLETZoneClock**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingClock			
Note	Describes a LET zone clock. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingClock*			
Aggregated by	*TimingExtension*.timingClock			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
accuracyExt	MultidimensionalTime	0..1	aggr	External synchronization accuracy within the LET Zone/ Zone Clock. **Tags:** atp.Status=draft
accuracyInt	MultidimensionalTime	0..1	aggr	Internal synchronization accuracy within the LET Zone/ Zone Clock. **Tags:** atp.Status=draft

Table D.58: TDLETZoneClock

Class	***TimingClock*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingClock			
Note	Describes an abstract clock. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	TDLETZoneClock			
Aggregated by	*TimingExtension*.timingClock			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
platformTime Base	GlobalTimeDomain	0..1	ref	Refers to a physical time base reference on the respective platform level **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=platformTimeBase.globalTimeDomain, platformTimeBase.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table D.59: TimingClock

Class	**TimingClockSyncAccuracy**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingClock			
Note	Describes the synchronization accuracy between exactly two TDClocks. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*TimingExtension*.timingClockSyncAccuracy			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
accuracy	MultidimensionalTime	0..1	aggr	Synchronization accuracy, treated as zero if not given. **Tags:** atp.Status=draft
lower	TimingClock	0..1	ref	References a target clock
upper	TimingClock	0..1	ref	References a source clock

Table D.60: TimingClockSyncAccuracy

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.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Traceable*			
Subclasses	AgeConstraint, *EventTriggeringConstraint*, ExecutionOrderConstraint, ExecutionTimeConstraint, LatencyTimingConstraint, OffsetTimingConstraint, SynchronizationPointConstraint, SynchronizationTimingConstraint			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
timingCondition	TimingCondition	0..1	ref	A timing condition the timing constraint depends on. In other words it specifies the condition the timing constraint holds.

Table D.61: TimingConstraint

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.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*TimingDescriptionEvent*, TimingDescriptionEventChain			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table D.62: TimingDescription

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

Table D.63: TimingDescriptionEvent

Class	TimingEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	This event is used to start RunnableEntities that shall be executed periodically.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
offset	TimeValue	0..1	attr	The value makes an assumption about the time offset of the first activation of the RunnableEntity triggered by the mapped TimingEvent relative to the periodic activation of the time base of this TimingEvent. Unit: second.
period	TimeValue	0..1	attr	Period of timing event in seconds. The value of this attribute shall be greater than zero.

Table D.64: TimingEvent

Class	***TimingExtension*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingExtensions			
Note	The abstract parent class of the different template specific timing extensions. Depending on the specific timing extension the timing descriptions and timing constraints, that can be used to specify the timing behavior, are restricted.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	BswCompositionTiming, BswModuleTiming, EcuTiming, SwcTiming, SystemTiming, VfbTiming			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
timingClock	TimingClock	*	aggr	A list of abstract model Clocks. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingClock.shortName, timingClock.variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=postBuild
timingClockSyncAccuracy	TimingClockSyncAccuracy	*	aggr	A list of accuracies - which may be used to specify synchronizations from one model clock to another model clock. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingClockSyncAccuracy.shortName, timingClockSyncAccuracy.variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=postBuild
timingCondition	TimingCondition	*	aggr	The timing condition specifies a specific condition. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingCondition.shortName, timingCondition.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	***TimingExtension*** (abstract)			
timing Description	TimingDescription	*	aggr	The timing descriptions that belong to a specific timing specification. In order to support different timing description variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingDescription.shortName, timing Description.variationPoint.shortLabel vh.latestBindingTime=postBuild
timing Guarantee	TimingConstraint	*	aggr	The timing constraints that belong to a specific timing specification in the role of a timing guarantee. In order to support different timing constraint variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingGuarantee.shortName, timing Guarantee.variationPoint.shortLabel vh.latestBindingTime=postBuild
timing Requirement	TimingConstraint	*	aggr	The timing constraints that belong to a specific timing specification in the role of a timing requirement. In order to support different timing constraint variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingRequirement.shortName, timing Requirement.variationPoint.shortLabel vh.latestBindingTime=postBuild
timingResource	TimingExtension Resource	0..1	aggr	The timing resource contains all instance references referred from within a timing condition formula of a timing view. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=timingResource.shortName

Table D.65: TimingExtension

Class	**Trigger**			
Package	M2::AUTOSARTemplates::CommonStructure::TriggerDeclaration			
Note	A trigger which is provided (i.e. released) or required (i.e. used to activate something) in the given context.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, BswModuleDescription.releasedTrigger, BswModuleDescription.requiredTrigger, ServiceInterface.trigger, TriggerInterface.trigger			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swImplPolicy	SwImplPolicyEnum	0..1	attr	This attribute, when set to value queued, allows for a queued processing of Triggers.
triggerPeriod	MultidimensionalTime	0..1	aggr	Optional definition of a period in case of a periodically (time or angle) driven external trigger.

Table D.66: Trigger

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

Table D.67: VariableAccess

Class	VariableDataPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	A VariableDataPrototype represents a formalized generic piece of information that is typically mutable by the application software layer. VariableDataPrototype is used in various contexts and the specific context gives the otherwise generic VariableDataPrototype a dedicated semantics.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationInterface.indication, *AtpClassifier*.atpFeature, BswInternalBehavior.arTypedPerInstanceMemory, BswModuleDescription.providedData, BswModuleDescription.requiredData, BulkNvDataDescriptor.bulkNvBlock, *InternalBehavior*.staticMemory, NvBlockDescriptor.ramBlock, NvDataInterface.nvData, SenderReceiverInterface.dataElement, ServiceInterface.event, SwcInternalBehavior.arTypedPerInstanceMemory, SwcInternalBehavior.explicitInterRunnableVariable, SwcInternalBehavior.implicitInterRunnableVariable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

Table D.68: VariableDataPrototype

E Splitable Elements in the Scope of this Document

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

Each entry in the following table consists of the identification of the specific model element itself and the applicable value of the tagged value atp.Splitkey.

For more information about the concept of splitable model elements and how these shall be treated please refer to [6].

Name of splitable element	*Splitkey*
TDCpSoftwareClusterMappingSet.tdCpSoftwareClusterResourceToTdMapping	tdCpSoftwareClusterResourceToTdMapping.shortName, tdCpSoftwareClusterResourceToTdMapping.variationPoint.shortLabel
TDCpSoftwareClusterMappingSet.tdCpSoftwareClusterToTdMapping	tdCpSoftwareClusterToTdMapping.shortName, tdCpSoftwareClusterToTdMapping.variationPoint.shortLabel
TimingClock.platformTimeBase	platformTimeBase.globalTimeDomain, platformTimeBase.variationPoint.shortLabel
TimingExtension.timingClock	timingClock.shortName, timingClock.variationPoint.shortLabel
TimingExtension.timingClockSyncAccuracy	timingClockSyncAccuracy.shortName, timingClockSyncAccuracy.variationPoint.shortLabel
TimingExtension.timingCondition	timingCondition.shortName, timingCondition.variationPoint.shortLabel
TimingExtension.timingDescription	timingDescription.shortName, timingDescription.variationPoint.shortLabel
TimingExtension.timingGuarantee	timingGuarantee.shortName, timingGuarantee.variationPoint.shortLabel
TimingExtension.timingRequirement	timingRequirement.shortName, timingRequirement.variationPoint.shortLabel
TimingExtension.timingResource	timingResource.shortName
TimingExtensionResource.timingArgument	timingArgument.shortName, timingArgument.variationPoint.shortLabel
TimingExtensionResource.timingMode	timingMode.shortName, timingMode.variationPoint.shortLabel
TimingExtensionResource.timingVariable	timingVariable.shortName, timingVariable.variationPoint.shortLabel

Table E.1: Usage of splitable elements

F Variation Points in the Scope of this Document

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

Each entry in the following table consists of the identification of the model element itself and the applicable value of the tagged value vh.latestBindingTime.

For more information about the concept of variation points and how model elements that contain variation points shall be treated please refer to [6].

Variation Point	*Latest Binding Time*
TDCpSoftwareClusterMappingSet.tdCpSoftwareClusterResourceToTdMapping	postBuild
TDCpSoftwareClusterMappingSet.tdCpSoftwareClusterToTdMapping	postBuild
TimingClock.platformTimeBase	postBuild
TimingExtension.timingClock	postBuild
TimingExtension.timingClockSyncAccuracy	postBuild
TimingExtension.timingCondition	postBuild
TimingExtension.timingDescription	postBuild
TimingExtension.timingGuarantee	postBuild
TimingExtension.timingRequirement	postBuild
TimingExtensionResource.timingArgument	postBuild
TimingExtensionResource.timingMode	postBuild
TimingExtensionResource.timingVariable	postBuild

Table F.1: Usage of variation points

G Change History

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

G.1 Change History of this document according to AUTOSAR Release R4.2.1

G.1.1 Added Specification Items in Release R4.2.1

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

Table G.1: Added Traceables in 4.2.1

G.1.2 Changed Specification Items in Release R4.2.1

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

Table G.2: Changed Traceables in 4.2.1

G.1.3 Deleted Specification Items in Release R4.2.1

none

G.1.4 Added Constraints in Release R4.2.1

Number	Heading
[constr_4544]	Specifying patternLength, patternJitter and patternPeriod
[constr_4545]	Referring either ExecutableEntitys or AbstractEvents
[constr_4546]	Setting the attribute isEvent
[constr_4547]	Setting the attribute permitMultipleReferencesToEE
[constr_4548]	EOCEventRef shall reference AbstractEvent in Ordinary Execution Order Constraint
[constr_4549]	EOCEventRef shall reference AbstractEvent in Hierarchical Execution Order Constraint

[constr_4550]	A Hierarchical Execution Order Constraint shall have an unambiguous root EOCExecutableEntityRefGroup

Table G.3: Added Constraints in R4.2.1

G.1.5 Changed Constraints in Release R4.2.1

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

Table G.4: Changed Constraints in R4.2.1

G.1.6 Deleted Constraints in Release R4.2.1

none

G.2 Change History of this document according to AUTOSAR Release R4.2.2

G.2.1 Added Specification Items in Release R4.2.2

none

G.2.2 Changed Specification Items in Release R4.2.2

none

G.2.3 Deleted Specification Items in Release R4.2.2

none

G.2.4 Added Constraints in Release R4.2.2

none

G.2.5 Changed Constraints in Release R4.2.2

none

G.2.6 Deleted Constraints in Release R4.2.2

none

G.3 Change History of this document according to AUTOSAR Release R4.3.0

G.3.1 Added Specification Items in Release R4.3.0

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

Table G.5: Added Traceables in 4.3.0

G.3.2 Changed Specification Items in Release R4.3.0

none

G.3.3 Added Constraints in Release R4.3.0

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

Table G.6: Added Constraints in R4.3.0

G.3.4 Changed Constraints in Release R4.3.0

Number	Heading
[constr_4501]	Application rule for the occurrence expression

Table G.7: Added Constraints in R4.3.0

G.3.5 Deleted Constraints in Release R4.3.0

none

G.4 Change History of this document according to AUTOSAR Release R4.3.1

G.4.1 Added Specification Items in Release R4.3.1

none

G.4.2 Changed Specification Items in Release R4.3.1

none

G.4.3 Added Constraints in Release R4.3.1

none

G.4.4 Changed Constraints in Release R4.3.1

none

G.4.5 Deleted Constraints in Release R4.3.1

none

G.5 Change History of this document according to AUTOSAR Release R4.4.0

G.5.1 Added Specification Items in Release R4.4.0

Number	Heading
[TPS_TIMEX_-00053]	Purpose of `BswCompositionTiming`
[TPS_TIMEX_-00054]	Purpose of `SynchronizationPointConstraint`
[TPS_TIMEX_-00055]	Purpose of `letInterval`

[TPS_TIMEX_-00056]	Attribute `category` used in context of `TimingDescriptionEvent` and LET
[TPS_TIMEX_-00057]	Attribute `category` used in context of `TimingDescriptionEventChain` and LET

Table G.8: Added Traceables in 4.4.0

G.5.2 Changed Specification Items in Release R4.4.0

none

G.5.3 Added Constraints in Release R4.4.0

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

Table G.9: Added Constraints in R4.4.0

G.5.4 Changed Constraints in Release R4.4.0

none

G.5.5 Deleted Constraints in Release R4.4.0

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

Table G.10: Deleted Constraints in R4.4.0

G.6 Change History of this document according to AUTOSAR Release R19-11

G.6.1 Added Specification Items in Release 19-11

none

G.6.2 Changed Specification Items in Release 19-11

none

G.6.3 Added Constraints in Release 19-11

none

G.6.4 Changed Constraints in Release 19-11

none

G.6.5 Deleted Constraints in Release 19-11

none

G.7 Change History of this document according to AUTOSAR Release R20-11

G.7.1 Added Specification Items in R20-11

Number	Heading
[TPS_TIMEX_00066]	Purpose of TDCpSoftwareClusterMappingSet
[TPS_TIMEX_00067]	Purpose of TDCpSoftwareClusterMapping
[TPS_TIMEX_00068]	Purpose of TDCpSoftwareClusterResourceMapping

Table G.11: Added Traceables in R20-11

G.7.2 Changed Specification Items in R20-11

none

G.7.3 Deleted Specification Items in R20-11

none

G.7.4 Added Constraints in R20-11

Number	Heading
[constr_4561]	Usage of the category value DISPATCH_ENTRY_POINT in TimingDescriptionEvent
[constr_4562]	Usage of the category value DISPATCH_EXIT_POINT in TimingDescriptionEvent
[constr_4563]	TDCpSoftwareClusterMapping shall reference only dispatcher or LET interval
[constr_4564]	TDCpSoftwareClusterResourceMapping shall reference only dispatcher or LET interval
[constr_4565]	Consistency of TDCpSoftwareClusterMapping and TDCpSoftwareClusterResourceMapping
[constr_4566]	SystemTiming describing timing of software clusters and category of System
[constr_4567]	Reference provider of TDCpSoftwareClusterMapping
[constr_4568]	Reference requestor of TDCpSoftwareClusterMapping

Table G.12: Added Constraints in R20-11

G.7.5 Changed Constraints in R20-11

none

G.7.6 Deleted Constraints in R20-11

none

G.8 Change History of this document according to AUTOSAR Release R21-11

G.8.1 Added Specification Items in R21-11

Number	Heading
[TPS_TIMEX_00001]	Purpose of TimingDescriptionEvent
[TPS_TIMEX_00002]	Purpose of TimingDescriptionEventChain
[TPS_TIMEX_00003]	EventTriggeringConstraint specifies occurrence behavior respectively model
[TPS_TIMEX_00004]	LatencyTimingConstraint specifies latency constraints
[TPS_TIMEX_00005]	AgeConstraint to specify age constraints
[TPS_TIMEX_00006]	SynchronizationTimingConstraint specifies synchronicity constraints

▽

△

Number	Heading
[TPS_TIMEX_00007]	ExecutionOrderConstraint specifies sequence of executing executable entities
[TPS_TIMEX_00008]	ExecutionTimeConstraint to specify execution time constraints
[TPS_TIMEX_00009]	Optional use of timing extensions
[TPS_TIMEX_00010]	PeriodicEventTriggering specifies periodic occurrences of events
[TPS_TIMEX_00011]	SporadicEventTriggering specifies sporadic occurrences of events
[TPS_TIMEX_00012]	ConcretePatternEventTriggering specifies concrete pattern of occurrences of events
[TPS_TIMEX_00013]	BurstPatternEventTriggering specifies burst of occurrences of events
[TPS_TIMEX_00014]	ArbitraryEventTriggering specifies arbitrary occurrences of an event
[TPS_TIMEX_00015]	OffsetTimingConstraint specifies offset between occurrences of events
[TPS_TIMEX_00016]	Purpose of TDEventVfb
[TPS_TIMEX_00017]	TDEventVariableDataPrototype specifies events observable at sender/receiver ports
[TPS_TIMEX_00018]	TDEventOperation specifies events observable at client/server ports.
[TPS_TIMEX_00019]	TDEventModeDeclaration specifies events observable at mode ports.
[TPS_TIMEX_00020]	TDEventSwcInternalBehavior specifies observable events of runnable entities
[TPS_TIMEX_00021]	Purpose of TDEventCom
[TPS_TIMEX_00022]	TDEventISignal specifies events related to the exchange of I-Signals
[TPS_TIMEX_00023]	TDEventIPdu specifies events related to the exchange of I-PDUs
[TPS_TIMEX_00024]	TDEventFrame specifies events related to the exchange of network frames
[TPS_TIMEX_00025]	TDEventFrClusterCycleStart specifies the event related to the start of a FlexRay communication cycle
[TPS_TIMEX_00026]	TDEventTTCanCycleStart specifies the event related to the start of a TTCAN communication cycle
[TPS_TIMEX_00027]	Purpose of TDEventComplex
[TPS_TIMEX_00028]	TDEventBswInternalBehavior specifies observable events of BSW module entities
[TPS_TIMEX_00029]	Purpose of TDEventBsw
[TPS_TIMEX_00030]	TDEventBswModule specifies observable events when basic software entries are called
[TPS_TIMEX_00031]	TDEventBswModeDeclaration specifies observable events in case of BSW mode communication
[TPS_TIMEX_00032]	Purpose of VfbTiming
[TPS_TIMEX_00033]	Purpose of SwcTiming
[TPS_TIMEX_00034]	Purpose of SystemTiming
[TPS_TIMEX_00035]	Purpose of BswModuleTiming
[TPS_TIMEX_00036]	Purpose of EcuTiming
[TPS_TIMEX_00037]	TimingConstraint is a Traceable
[TPS_TIMEX_00038]	Purpose of EOCExecutableEntityRefAbstract

▽

△

Number	Heading
[TPS_TIMEX_00039]	`TDEventTrigger` specifies events observable at trigger ports
[TPS_TIMEX_00040]	Blueprinting `VfbTiming`
[TPS_TIMEX_00041]	Purpose of `EOCExecutableEntityRefGroup`
[TPS_TIMEX_00042]	Purpose of `TDEventVfbPort`
[TPS_TIMEX_00043]	Purpose of `TDEventVfbReference`
[TPS_TIMEX_00044]	Purpose of `TDEventSwc`
[TPS_TIMEX_00045]	Purpose of `TDEventSwcInternalBehaviorReference`
[TPS_TIMEX_00046]	Purpose of `EOCExecutableEntityRef`
[TPS_TIMEX_00047]	Purpose of `ExecutionOrderConstraintTypeEnum`
[TPS_TIMEX_00048]	Purpose of `EOCEventRef`
[TPS_TIMEX_00049]	Purpose of `TimingCondition`
[TPS_TIMEX_00050]	Purpose of `TimingConditionFormula`
[TPS_TIMEX_00051]	Purpose of `TimingExtensionResource`
[TPS_TIMEX_00052]	`TDEventFrameEthernet` specifies events related to the exchange of Ethernet frames
[TPS_TIMEX_00053]	Purpose of `BswCompositionTiming`
[TPS_TIMEX_00054]	`SynchronizationPointConstraint` explicitly specifies a synchronization point between executable entities
[TPS_TIMEX_00055]	Purpose of the attribute `letInterval`
[TPS_TIMEX_00056]	Attribute `category` used in context of `TimingDescriptionEvent` and LET
[TPS_TIMEX_00057]	Attribute `category` used in context of `TimingDescriptionEventChain` and LET
[TPS_TIMEX_00066]	Purpose of `TDCpSoftwareClusterMappingSet`
[TPS_TIMEX_00067]	Purpose of `TDCpSoftwareClusterMapping`
[TPS_TIMEX_00068]	Purpose of `TDCpSoftwareClusterResourceMapping`

Table G.13: Added Specification Items in R21-11

G.8.2 Changed Specification Items in R21-11

none

G.8.3 Deleted Specification Items in R21-11

none

G.8.4 Added Constraints in R21-11

Number	Heading
[constr_4500]	Restricted usage of functions
[constr_4501]	Application rule for the occurrence expression in TDEventComplex
[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
[constr_4508]	TDEventVfb shall reference PortPrototypeBlueprint only in Blueprints
[constr_4509]	Only VfbTiming shall be a Blueprint
[constr_4510]	Specifying references to RunnableEntity and VariableAccess
[constr_4511]	Validity of referencing RunnableEntity
[constr_4512]	Validity of referencing VariableAccess
[constr_4513]	SynchronizationTimingConstraint shall reference at least two events
[constr_4514]	SynchronizationTimingConstraint shall reference at least two event chains
[constr_4515]	Specifying stimulus and response in TimingDescriptionEventChain
[constr_4516]	Specifying event chain segments
[constr_4517]	Referencing no further event chain segments
[constr_4518]	Specifying stimulus event and response event of first and last event chain segment
[constr_4519]	Specifying patternLength
[constr_4520]	Specifying attribute synchronizationConstraintType
[constr_4521]	Specifying attribute synchronizationConstraintType
[constr_4522]	SynchronizationTimingConstraint shall either reference events or event chains
[constr_4523]	Specifying attributes maxCycles and maxSlots
[constr_4524]	Referencing TimingDescriptionEvent
[constr_4525]	Precedence of successor relationships successor and directSuccessor
[constr_4526]	Specifying maxCycles and maxSlots in a Repetitive Execution Order Constraint
[constr_4527]	Referencing TimingDescriptionEvent in a Repetitive Execution Order Constraint
[constr_4528]	The *root* EOCExecutableEntityRefGroup shall reference only EOCExecutableEntityRefGroups
[constr_4529]	Number of nested elements referenced by the *root* EOCExecutableEntityRefGroup
[constr_4530]	An EOCExecutableEntityRefGroup representing a cycle shall reference only EOCExecutableEntityRefs respectively EOCEventRefs

△

Number	Heading
[constr_4531]	Number of nested elements referenced by EOCExecutableEntityRefGroup representing a cycle
[constr_4532]	Successor relationship is not self-referencing
[constr_4533]	Maximum number of successor relationships
[constr_4534]	Maximum number of directSuccessor relationships
[constr_4536]	Compatible recurrence of any ExecutableEntity
[constr_4537]	References among elements in an ExecutionOrderConstraint
[constr_4538]	Hierarchical Execution Order Constraint: EOCExecutableEntityRef, EOCEventRef, and EOCExecutableEntityRefGroup shall be target or source of a successor relationship
[constr_4539]	The successor relationships successor and directSuccessor shall not be used
[constr_4540]	maxCycles and maxSlots shall not be zero
[constr_4541]	EOCExecutableEntityRef shall reference ExecutableEntity in Ordinary Execution Order Constraint
[constr_4542]	EOCExecutableEntityRef shall reference ExecutableEntity in Hierarchical Execution Order Constraint
[constr_4543]	Maximum value of the parameter minimumInterArrivalTime
[constr_4544]	Specifying patternLength, patternJitter and patternPeriod
[constr_4545]	Referring either ExecutableEntitys or AbstractEvents
[constr_4546]	Setting the attribute isEvent
[constr_4547]	Setting the attribute permitMultipleReferencesToEE
[constr_4548]	EOCEventRef shall reference AbstractEvent in Ordinary Execution Order Constraint
[constr_4549]	EOCEventRef shall reference AbstractEvent in Hierarchical Execution Order Constraint
[constr_4550]	A Hierarchical Execution Order Constraint shall have an unambiguous root EOCExecutableEntityRefGroup
[constr_4551]	Use only Numericals in TDEventOccurrenceExpression
[constr_4552]	Restricted usage of AutosarVariableInstance for Content Filter
[constr_4553]	Usage of optional attribute ignoreOrderAllowed
[constr_4554]	Usage of optional directed association/reference letInterval
[constr_4555]	Usage of the category value LET_RELEASE in TimingDescriptionEvent
[constr_4556]	Usage of the category value LET_TERMINATE in TimingDescriptionEvent
[constr_4557]	Usage of the category value LET_INTERVAL in TimingDescriptionEventChain
[constr_4558]	Applicability of LET semantics
[constr_4559]	category of TimingDescriptionEvent shall not be extended
[constr_4560]	category of TimingDescriptionEventChain shall not be extended
[constr_4561]	Usage of the category value DISPATCH_ENTRY_POINT in TimingDescriptionEvent
[constr_4562]	Usage of the category value DISPATCH_EXIT_POINT in TimingDescriptionEvent

▽

△

Number	Heading
[constr_4563]	`TDCpSoftwareClusterMapping` shall reference only dispatchers or LET intervals
[constr_4564]	`TDCpSoftwareClusterResourceMapping` shall reference only dispatchers or LET intervals
[constr_4565]	Consistency of `TDCpSoftwareClusterMapping` and `TDCpSoftwareClusterResourceMapping`
[constr_4566]	`SystemTiming` describing timing of software clusters and category of `System`
[constr_4567]	Reference `provider` of `TDCpSoftwareClusterMapping`
[constr_4568]	Reference `requestor` of `TDCpSoftwareClusterMapping`

Table G.14: Added Constraints in R21-11

G.8.5 Changed Constraints in R21-11

none

G.8.6 Deleted Constraints in R21-11

none

G.9 Change History of this document according to AUTOSAR Release R22-11

G.9.1 Added Specification Items in R22-11

Number	Heading
[TPS_TIMEX_00100]	Optionality of `accuracyInt`
[TPS_TIMEX_00101]	Semantics of `accuracyInt`
[TPS_TIMEX_00102]	Optionality of `LatencyTimingConstraint.minimum` used in an LET interval
[TPS_TIMEX_00103]	Optionality of `accuracyExt`
[TPS_TIMEX_00104]	Semantics of `accuracyExt`
[TPS_TIMEX_00105]	Purpose of `TimingClockSyncAccuracy`
[TPS_TIMEX_00106]	Purpose of `TimingClockSyncAccuracy.upper`
[TPS_TIMEX_00107]	Purpose of `TimingClockSyncAccuracy.lower`
[TPS_TIMEX_00108]	Usage of the attribute `accuracy` of `TimingClockSyncAccuracy`
[TPS_TIMEX_00109]	Optionality of `accuracy`

▽

△

Number	Heading
[TPS_TIMEX_00110]	Standardized `category`s of `TimingDescriptionEventChain`
[TPS_TIMEX_00111]	Semantics of `TimingDescriptionEventChain.stimulus`
[TPS_TIMEX_00112]	Representation of an LET interval `release`
[TPS_TIMEX_00113]	Representation of an LET interval `terminate`
[TPS_TIMEX_00114]	Semantics of `TimingDescriptionEventChain.response`
[TPS_TIMEX_00115]	Representation of an SL-LET interval `release`
[TPS_TIMEX_00116]	Representation of an SL-LET interval `terminate`
[TPS_TIMEX_00117]	Representation of the *recurrence* of an LET interval
[TPS_TIMEX_00118]	Usage of `TimingClock.platformTimeBase`
[TPS_TIMEX_00119]	Purpose of `TDLETZoneClock`
[TPS_TIMEX_00120]	Purpose of `TDEventSLLET`
[TPS_TIMEX_00121]	Representation of the *duration* of an LET interval
[TPS_TIMEX_00122]	Application of a `PeriodicEventTriggering` constraint used to specify the *recurrence* of an LET interval
[TPS_TIMEX_00123]	Purpose of `TimingClock`
[TPS_TIMEX_00124]	Purpose of `TDEventSLLETPort`
[TPS_TIMEX_00125]	Representation of an SL-LET interval in a `TimingDescriptionEventChain`
[TPS_TIMEX_00126]	Representation of an SL-LET interval in a `TimingDescriptionEvent`

Table G.15: Added Specification Items in R22-11

G.9.2 Changed Specification Items in R22-11

Number	Heading
[TPS_TIMEX_00056]	Standardized `category`s of `TimingDescriptionEvent` in Classic Platform

Table G.16: Changed Specification Items in R22-11

G.9.3 Deleted Specification Items in R22-11

none

G.9.4 Added Constraints in R22-11

Number	Heading
[constr_6816]	Restricted usage of TimingDescriptionEventChain.isPipeliningPermitted in TimingDescriptionEventChain
[constr_6817]	Restricted usage of TimingDescriptionEvent.clockReference
[constr_6818]	Existence of EventTriggeringConstraint.event
[constr_6819]	Existence of PeriodicEventTriggering.jitter
[constr_6820]	Existence of PeriodicEventTriggering.minimumInterArrivalTime
[constr_6821]	Existence of PeriodicEventTriggering.period
[constr_6822]	Existence of SporadicEventTriggering.maximumInterArrivalTime
[constr_6823]	Existence of SporadicEventTriggering.minimumInterArrivalTime
[constr_6824]	Existence of ConcretePatternEventTriggering.patternLength
[constr_6825]	Existence of BurstPatternEventTriggering.maxNumberOfOccurrences
[constr_6826]	Existence of BurstPatternEventTriggering.minimumInterArrivalTime
[constr_6827]	Existence of BurstPatternEventTriggering.patternLength
[constr_6828]	Existence of ArbitraryEventTriggering.minimumDistance
[constr_6829]	Existence of ArbitraryEventTriggering.maximumDistance
[constr_6830]	Existence of ConfidenceInterval.lowerBound
[constr_6831]	Existence of ConfidenceInterval.propability
[constr_6832]	Existence of ConfidenceInterval.upperBound
[constr_6833]	Existence of ExecutionOrderConstraint.orderedElement
[constr_6834]	Existence of EOCExecutableEntityRefGroup.nestedElement
[constr_6835]	Existence of ExecutionTimeConstraint.executionTimeType
[constr_6836]	Existence of ExecutionTimeConstraint.executable
[constr_6837]	Existence of LatencyTimingConstraint.latencyConstraintType
[constr_6838]	Existence of LatencyTimingConstraint.maximum
[constr_6839]	Existence of LatencyTimingConstraint.minimum
[constr_6840]	Existence of LatencyTimingConstraint.nominal
[constr_6841]	Existence of LatencyTimingConstraint.scope
[constr_6842]	Existence of OffsetTimingConstraint.maximum
[constr_6843]	Existence of OffsetTimingConstraint.minimum
[constr_6844]	Existence of OffsetTimingConstraint.source
[constr_6845]	Existence of OffsetTimingConstraint.target
[constr_6846]	Existence of SynchronizationTimingConstraint.synchronizationConstraintType
[constr_6847]	Existence of SynchronizationTimingConstraint.tolerance
[constr_6848]	Existence of VfbTiming.component
[constr_6849]	Existence of SystemTiming.system
[constr_6850]	Existence of BswModuleTiming.behavior

▽

△

Number	Heading
[constr_6851]	Existence of BswCompositionTiming.implementation
[constr_6852]	Existence of EcuTiming.ecuConfiguration
[constr_6853]	Existence of ModeInBswInstanceRef. contextModeDeclarationGroupPrototype
[constr_6854]	Existence of ModeInBswInstanceRef.targetModeDeclaration
[constr_6855]	Existence of ModeInSwcInstanceRef. contextModeDeclarationGroupPrototype
[constr_6856]	Existence of ModeInSwcInstanceRef.contextPort
[constr_6857]	Existence of ModeInSwcInstanceRef.targetModeDeclaration
[constr_6858]	Existence of TDEventBswInternalBehavior. tdEventBswInternalBehaviorType
[constr_6859]	Existence of TDEventBswInternalBehavior.bswModuleEntity
[constr_6860]	Existence of TDEventBswModule.tdEventBswModuleType
[constr_6861]	Existence of TDEventBswModule.bswModuleEntry
[constr_6862]	Existence of TDEventBswModeDeclaration. tdEventBswModeDeclarationType
[constr_6863]	Existence of TDEventBswModeDeclaration.modeDeclaration
[constr_6864]	Existence of TDEventISignal.tdEventType
[constr_6865]	Existence of TDEventISignal.iSignal
[constr_6866]	Existence of TDEventISignal.physicalChannel
[constr_6867]	Existence of TDEventIPdu.tdEventType
[constr_6868]	Existence of TDEventIPdu.iPdu
[constr_6869]	Existence of TDEventIPdu.physicalChannel
[constr_6870]	Existence of TDEventFrame.tdEventType
[constr_6871]	Existence of TDEventFrame.frame
[constr_6872]	Existence of TDEventFrame.physicalChannel
[constr_6873]	Existence of TDEventFrameEthernet.tdEventType
[constr_6874]	Existence of TDHeaderIdRange.maxHeaderId
[constr_6875]	Existence of TDHeaderIdRange.minHeaderId
[constr_6876]	Existence of TDEventCycleStart.cycleRepetition
[constr_6877]	Existence of TDEventFrClusterCycleStart.frCluster
[constr_6878]	Existence of TDEventTTCanCycleStart.ttCanCluster
[constr_6879]	Existence of TDEventOccurrenceExpression.formula
[constr_6880]	Existence of AutosarVariableInstance.variableInstance
[constr_6881]	Existence of AutosarOperationArgumentInstance. operationArgumentInstance
[constr_6882]	Existence of TDEventSwcInternalBehavior. tdEventSwcInternalBehaviorType
[constr_6883]	Existence of TDEventSwcInternalBehavior.runnable
[constr_6884]	Existence of TDEventSwcInternalBehaviorReference. referencedTDEventSwc

▽

△

Number	Heading
[constr_6885]	Existence of TDEventVfbPort.isExternal
[constr_6886]	Existence of TDEventVfbReference.referencedTDEventVfb
[constr_6887]	Existence of TDEventVariableDataPrototype.tdEventVariableDataPrototypeType
[constr_6888]	Existence of TDEventVariableDataPrototype.dataElement
[constr_6889]	Existence of TDEventOperation.tdEventOperationType
[constr_6890]	Existence of TDEventOperation.operation
[constr_6891]	Existence of TDEventModeDeclaration.tdEventModeDeclarationType
[constr_6892]	Existence of TDEventModeDeclaration.modeDeclaration
[constr_6893]	Existence of TDEventTrigger.tdEventTriggerType
[constr_6894]	Existence of TDEventTrigger.trigger
[constr_6895]	Existence of TimingDescriptionEventChain.response
[constr_6896]	Existence of TimingDescriptionEventChain.stimulus
[constr_6897]	Existence of TimingDescriptionEventChain.segment
[constr_6898]	Existence of ConcretePatternEventTriggering.offset
[constr_6899]	Existence of ModeInSwcInstanceRef.base

Table G.17: Added Constraints in R22-11

G.9.5 Changed Constraints in R22-11

Number	Heading
[constr_4559]	Restriction of TimingDescriptionEvent.category
[constr_4560]	Restriction of TimingDescriptionEventChain.category

Table G.18: Changed Constraints in R22-11

G.9.6 Deleted Constraints in R22-11

none

G.10 Change History of this document according to AUTOSAR Release R23-11

G.10.1 Added Specification Items in R23-11

Number	Heading
[TPS_TIMEX_00127]	SystemTiming of classic platform software clusters
[TPS_TIMEX_00128]	Default letDataExchangeParadigm
[TPS_TIMEX_00129]	Representation of the root in a Hierarchical Execution Order Constraint

Table G.19: Added Specification Items in R23-11

G.10.2 Changed Specification Items in R23-11

Number	Heading
[TPS_TIMEX_00005]	Semantics of an AgeConstraint
[TPS_TIMEX_00009]	Optionality of Timing Extensions
[TPS_TIMEX_00066]	Purpose of TDCpSoftwareClusterMappingSet
[TPS_TIMEX_00067]	Purpose of TDCpSoftwareClusterMapping
[TPS_TIMEX_00068]	Purpose of TDCpSoftwareClusterResourceMapping
[TPS_TIMEX_00115]	Representation of an SL-LET interval release
[TPS_TIMEX_00116]	Representation of an SL-LET interval terminate

Table G.20: Changed Specification Items in R23-11

G.10.3 Deleted Specification Items in R23-11

Number	Heading
[TPS_TIMEX_00037]	TimingConstraint is a Traceable
[TPS_TIMEX_00040]	Blueprinting VfbTiming
[TPS_TIMEX_00126]	Representation of an SL-LET interval in a TimingDescriptionEvent

Table G.21: Deleted Specification Items in R23-11

G.10.4 Added Constraints in R23-11

Number	Heading
[constr_6900]	Dual existence of TDEventVfb.port and TDEventVfb.portPrototypeBlueprint
[constr_6901]	Existence of TDEventBsw.bswModuleDescription
[constr_6906]	Conformity of stimulus and response in a TimingDescriptionEventChain
[constr_6907]	Restriction of EOCExecutableEntityRefGroup.triggeringEvent
[constr_6908]	Restriction of EOCExecutableEntityRefGroup.letDataExchangeParadigm
[constr_6909]	Singleton ROOT_GROUP in a Hierarchical Execution Order Constraint
[constr_6910]	Referencing **from** a ROOT_GROUP in a Hierarchical Execution Order Constraint
[constr_6911]	Referencing **to** a ROOT_GROUP in a Hierarchical Execution Order Constraint
[constr_6912]	Mandatory specification of LET interval **recurrence**
[constr_6913]	Restriction on RTEEvents used in an LET interval
[constr_6914]	Restriction of the port context of an AgeConstraint
[constr_6915]	Affinity of ISignal in TDEventISignal
[constr_6916]	Affinity of Frame in TDEventFrame
[constr_6917]	Affinity of IPdu in TDEventIPdu

Table G.22: Added Constraints in R23-11

G.10.5 Changed Constraints in R23-11

Number	Heading
[constr_4504]	Restriction of the scope of an AgeConstraint
[constr_4508]	Existence of TDEventVfbPort.portPrototypeBlueprint
[constr_4523]	Restriction of maxCycleRepetitions and maxSlotsPerCycle to Repetitive Execution Order Constraint
[constr_4547]	Restriction of ExecutionOrderConstraint.permitMultipleReferencesToEE
[constr_4554]	Restriction of the referenced TimingDescriptionEventChain for a letInterval
[constr_4561]	Usage of the category value DISPATCH_ENTRY_POINT in TimingDescriptionEvent
[constr_4562]	Usage of the category value DISPATCH_EXIT_POINT in TimingDescriptionEvent
[constr_4563]	TDCpSoftwareClusterMapping shall reference only dispatchers or LET intervals
[constr_4564]	TDCpSoftwareClusterResourceMapping shall reference only dispatchers or LET intervals
[constr_4565]	Consistency of TDCpSoftwareClusterMapping and TDCpSoftwareClusterResourceMapping

▽

△

Number	Heading
[constr_4567]	Reference `provider` of `TDCpSoftwareClusterMapping`
[constr_4568]	Reference `requestor` of `TDCpSoftwareClusterMapping`

Table G.23: Changed Constraints in R23-11

G.10.6 Deleted Constraints in R23-11

Number	Heading
[constr_4501]	Application rule for the occurrence expression in `TDEventComplex`
[constr_4509]	Only `VfbTiming` shall be a Blueprint
[constr_4524]	Referencing `TimingDescriptionEvent`
[constr_4553]	Usage of optional attribute `ignoreOrderAllowed`
[constr_4555]	Usage of the category value `LET_RELEASE` in `TimingDescriptionEvent`
[constr_4556]	Usage of the category value `LET_TERMINATE` in `TimingDescriptionEvent`
[constr_4557]	Usage of the category value `LET_INTERVAL` in `TimingDescriptionEventChain`
[constr_4558]	Applicability of LET semantics
[constr_4566]	`SystemTiming` describing timing of software clusters and category of `System`
[constr_6840]	Existence of `LatencyTimingConstraint.nominal`

Table G.24: Deleted Constraints in R23-11