Document Title	Specification of Timing Extensions for Adaptive Platform
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	968

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• No content changes
2023-11-23	R23-11	AUTOSAR Release Management	• Editorial changes • See Disclaimer note in "AP TR AdaptivePlatformReleaseOverview"
2022-11-24	R22-11	AUTOSAR Release Management	• Added SL-LET feature
2021-11-25	R21-11	AUTOSAR Release Management	• Corrected specification item numbers and constraint identifier numbers to make the number unique and indicated correct status by setting it to DRAFT • Corrected spelling errors
2020-11-30	R20-11	AUTOSAR Release Management	• Initial release

Disclaimer

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] Generic Structure Template
AUTOSAR_FO_TPS_GenericStructureTemplate

[6] Specification of Timing Extensions for Classic Platform
AUTOSAR_CP_TPS_TimingExtensions

[7] Methodology for Adaptive Platform
AUTOSAR_AP_TR_Methodology

[8] Glossary
AUTOSAR_FO_TR_Glossary

[9] 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. 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 an executable 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:

- *System Timing* and System View
- *Machine Timing* and Machine View
- *Executable Timing* and Executable View
- *Service Timing* and Service View

In particular, these events are used to specify:

- the usage and operation of services in timing views such VFB, System, Machine, Executable and Service 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.

3 Modeling

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

3.1 TimingExtensions

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 timing views are:

- **VfbTiming**: timing information related to the interaction of `AdaptiveApplicationSwComponentType`s at VFB level (3.1.1)
- **ExecutableTiming**: timing information related to an `Executable` (3.1.2)
- **SystemTiming**: timing information related to a `System`, utilizing information about topology, software deployment, and signal mapping (3.1.3)
- **ServiceTiming**: timing information related to a *service*, specifically `AdaptivePlatformServiceInstance` (3.1.4)
- **MachineTiming**: timing information related to a `Machine` (3.1.5)

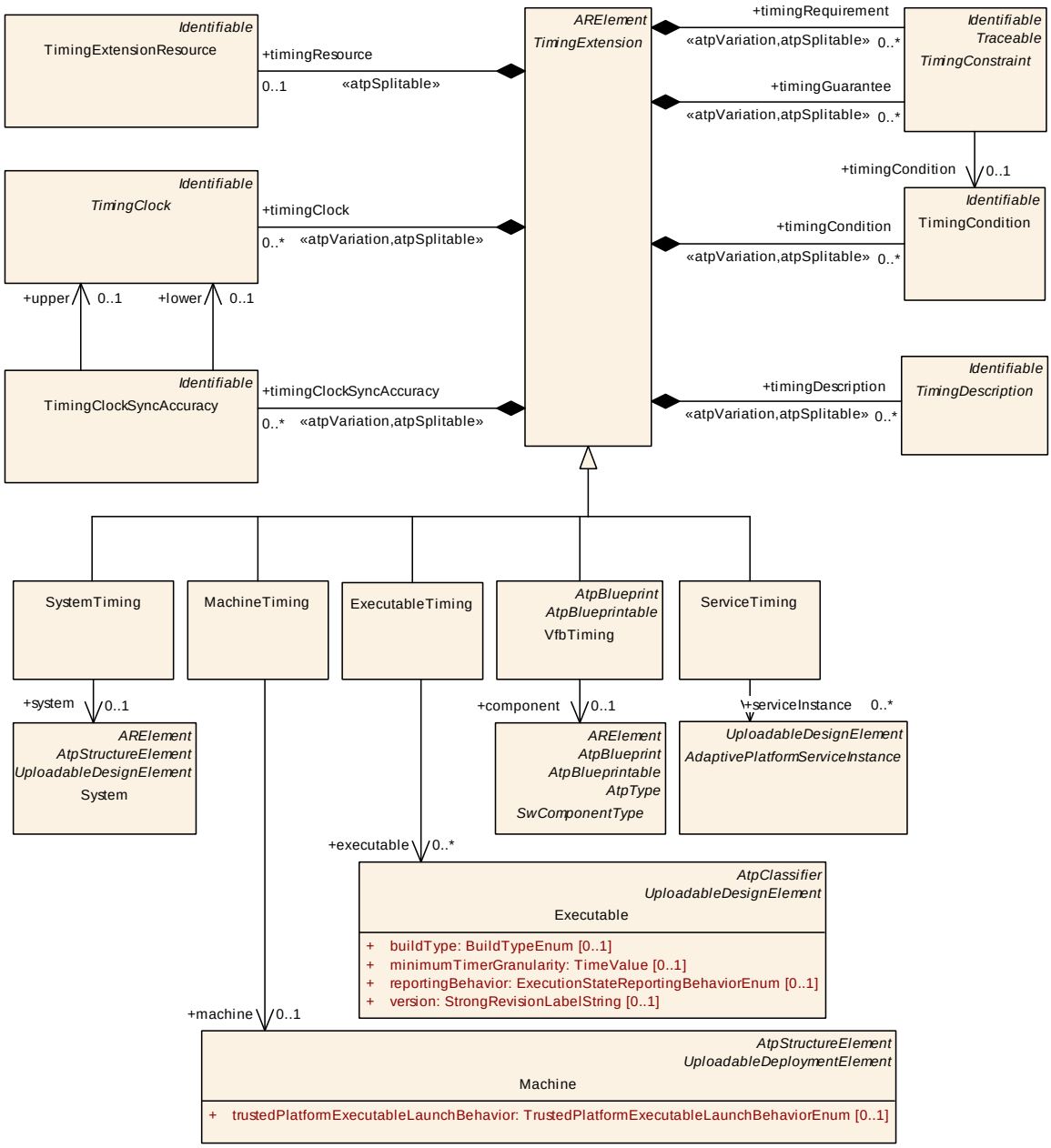


Figure 3.1: Timing Extensions top-level view

3.1.1 VfbTiming

AUTOSAR defines the *Virtual Functional Bus* [4] as a composition of `SwComponentPrototype`s 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. For these mentioned restrictions (irrelevance of the physical distribution, black box view), TimingDescriptions at VFB level should only refer to SwComponentTypes, PortPrototypes and their connections.

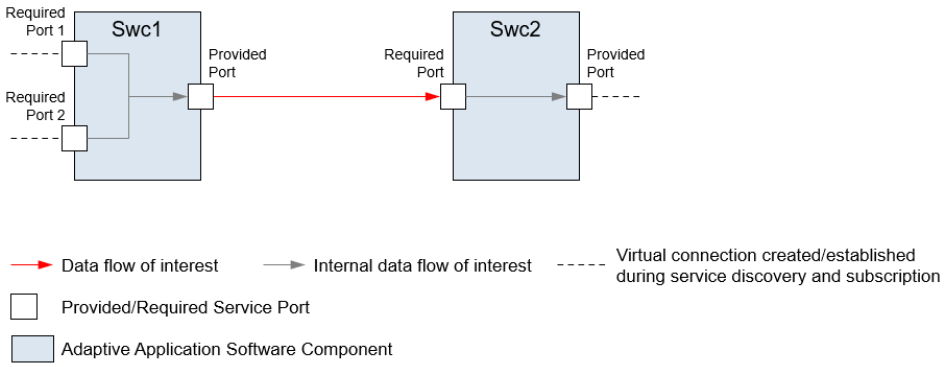


Figure 3.2: Example: Data flow in the scope of the VfbTiming view

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.

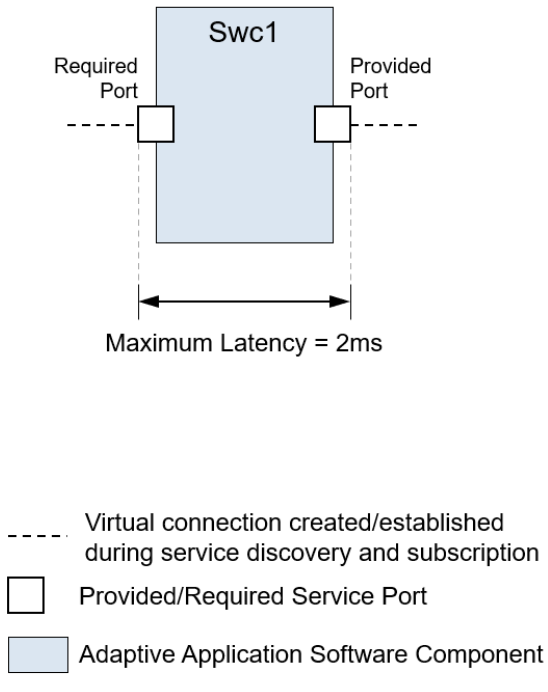


Figure 3.3: Example: Latency requirement

As an example, consider the timing constraint illustrated in Figure 3.3: ”From the point in time, where the value is received by AA named *Swc1*, until the point in time, where the newly calculated data value is sent via the provided service port, there shall be a maximum latency of 2 ms”. This would be attached to the timing description that refers to an `AdaptiveApplicationSwComponentType` called *Swc1*.

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

[TPS_TIMEX_00087] Purpose of `VfbTiming`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001

⌈The element `VfbTiming` aggregates all timing information, timing descriptions and timing constraints related to the `VFB` View.⌋

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

3.1.2 ExecutableTiming

[TPS_TIMEX_00064] Purpose of `ExecutableTiming`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00024

⌈The element `ExecutableTiming` aggregates all timing information, timing descriptions and timing constraints, that is related to the Executable View.⌋

[constr_6902] Existence of `ExecutableTiming.executable` ⌈For each `ExecutableTiming`, the reference to a `Executable` in the role `exe-`

cutable shall exist **at the time when the Executable Timing Description is complete.**⌋

Class	**ExecutableTiming**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingExtensions			
Note	This meta-class represents the timing view for one or more executables. **Tags:** atp.Status=draft atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *TimingExtension*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
executable	Executable	*	ref	This defines the scope of a ExecutableTiming. All corresponding timing descriptions and constraints shall be defined within this scope. **Tags:** atp.Status=draft

Table 3.2: ExecutableTiming

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 machines in order to configure the system. The main configuration result is the mapping of software components to Machines 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 AdaptiveApplicationSwComponentTypes, it is possible to use the same concepts that are available in the view VfbTiming 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.

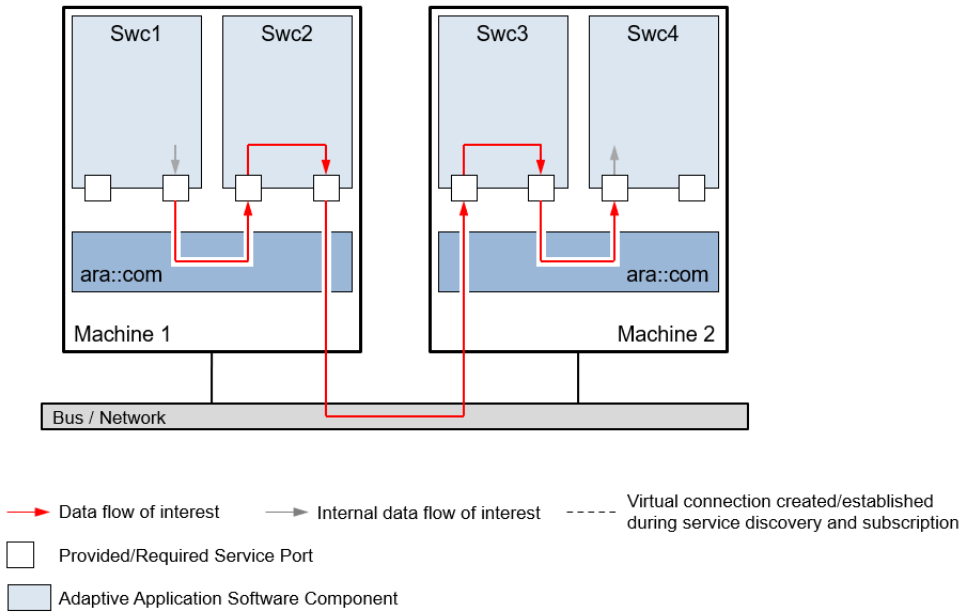


Figure 3.4: Example: Data flow in the scope of System Timing view

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 within a machine, or remote communication between machines across a communication bus. A system-specific timing description thus can refer to signals and frames sent across a physical network.

[TPS_TIMEX_00088] Purpose of `SystemTiming`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001

⌈The element `SystemTiming` aggregates all timing information, timing descriptions and timing constraints, that is related to the System View.⌋

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

▽

△

Class	SystemTiming			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
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

3.1.4 ServiceTiming

[TPS_TIMEX_00065] Purpose of `ServiceTiming`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00024

⌈The element `ServiceTiming` aggregates all timing information, timing descriptions and timing constraints, that is related to the Service View.⌋

[constr_6903] Existence of `ServiceTiming.serviceInstance` ⌈For each `ServiceTiming`, the reference to a `AdaptivePlatformServiceInstance` in the role `serviceInstance` shall exist **`at the time when the Service Timing Description is complete`**.⌋

Class	ServiceTiming			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingExtensions			
Note	This meta-class represents the timing view for one or more service instances. **Tags:** atp.Status=draft atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *TimingExtension*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
serviceInstance	AdaptivePlatform ServiceInstance	*	ref	This defines the scope of a ServiceTiming. All corresponding timing descriptions and constraints shall be defined within this scope. **Tags:** atp.Status=draft

Table 3.4: ServiceTiming

3.1.5 MachineTiming

[TPS_TIMEX_00063] Purpose of `MachineTiming`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00024

⌈The element `MachineTiming` aggregates all timing information, timing descriptions and timing constraints, that is related to the Machine View.⌋

[constr_6904] Existence of `MachineTiming.machine` ⌈For each `MachineTiming`, the reference to a `Machine` in the role `machine` shall exist **`at the time when the Machine Timing Description is complete`**.⌋

Class	**MachineTiming**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingExtensions			
Note	This meta-class represents the timing view for a machine. **Tags:** atp.Status=draft atp.recommendedPackage=TimingExtensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *TimingExtension*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
machine	Machine	0..1	ref	This defines the scope of a MachineTiming. All corresponding timing descriptions and constraints shall be defined within this scope. **Tags:** atp.Status=draft

Table 3.5: MachineTiming

3.2 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.

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 3.5.2) 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 3.5.1) 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.

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

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

3.3 Specifying Time Sets

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 process shall be periodically activated either at 1ms, 2ms, 5ms, 8ms, or 10ms. In other words, it is perfectly fine to decide that the process is activated every 8ms. Indeed, it is allowed to activate the process 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 process at {1, 2, 5, 8, 10} ms.

For the purpose of specifying time sets the timing extensions utilize the ”Variant Handling” capabilities specified and described in [5].

3.4 Timing Conditions

Please refer to [6] chapter "Timing Conditions".

3.5 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 3.5. These are detailed in the next sections.

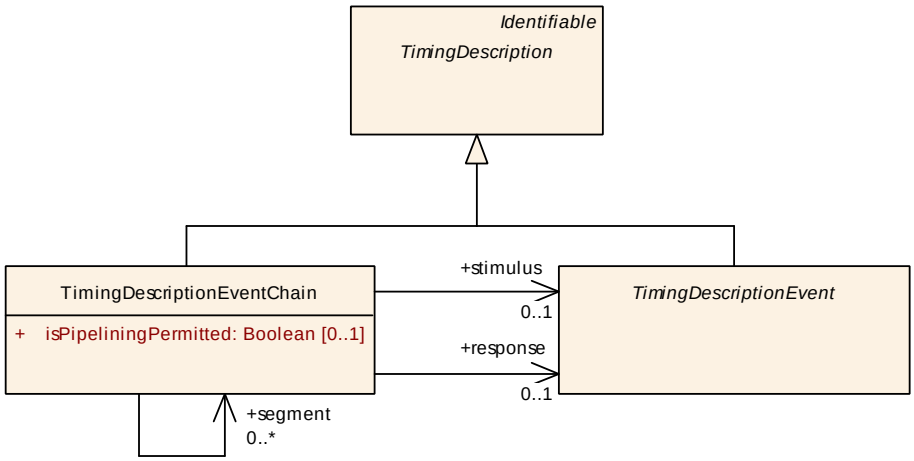


Figure 3.5: TimingDescription

3.5.1 TimingDescriptionEventChain

A timing event chain describes a causal order for a set of functionally dependent timing events. Each event chain defines at least the relationship between two differing events, its *stimulus* and *response* [constr_4515].

This means that if the stimulus event occurs then the response event occurs after or in other words the response event follows if and only if the stimulus event occurred before.

[TPS_TIMEX_00070] Purpose of `TimingDescriptionEventChain`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00004, RS_TIMEX_00005

⌈The element `TimingDescriptionEventChain` is used to specify a causal relationship between timing description events and their occurrences during the runtime of a system.⌋

Thus, by means of an event chain, the correlation between a stimulation of a system and its corresponding response can be explicitly described, and used as a formalized definition of the scope for timing constraints. This is important, because timing constraints refer to a specific part of the overall system's timing and need clear validity semantics.

[constr_4581] Specifying stimulus and response in `TimingDescriptionEventChain`

Status: DRAFT

⌈The references between `TimingDescriptionEventChain` and `TimingDescriptionEvent` playing the role `stimulus` and `response` shall not reference the same `TimingDescriptionEvent`.⌋

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

[TPS_TIMEX_00095] Standardized categorys of TimingDescriptionEventChain in Adaptive Platform

Status: DRAFT

⌈AUTOSAR standardizes the following categorys of TimingDescriptionEventChain and their semantics:

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

⌋

Please note constraint: [constr_4515] and specification items: [TPS_TIMEX_00111], [TPS_TIMEX_00114] in [6] shall apply here also.

3.5.1.1 Segments

[constr_4582] Specifying event chain segments

Status: DRAFT

⌈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_4583] Referencing no further event chain segments

Status: DRAFT

⌈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_4584] Specifying `stimulus` event and `response` event of first and last event chain segment

Status: DRAFT

⌈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.⌋

3.5.1.2 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.

3.5.1.2.1 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 3.6 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.

3.5.1.2.2 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 3.6 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).

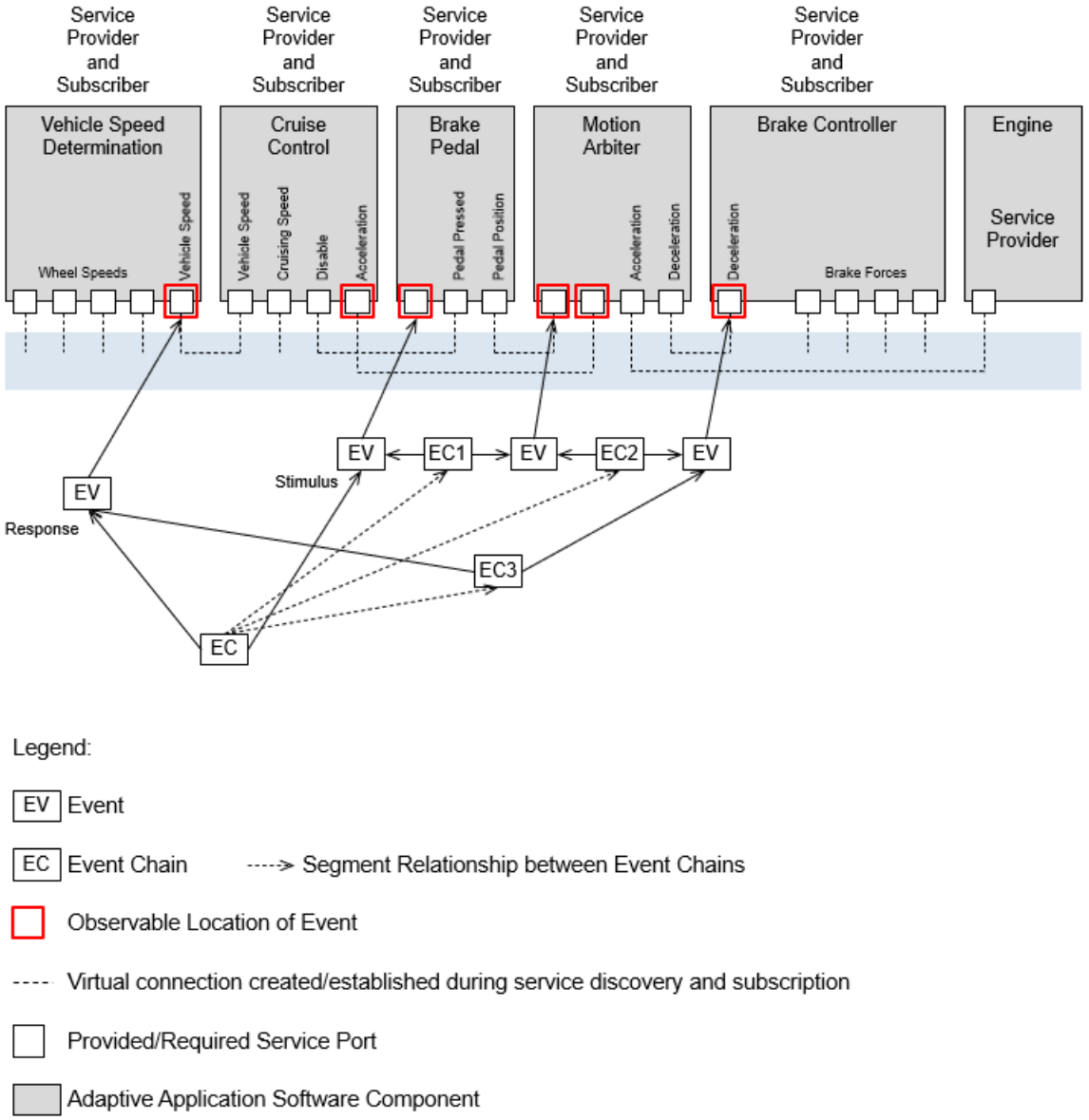


Figure 3.6: Example of a composed and decomposed event chain

3.5.1.3 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.

3.5.1.3.1 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.

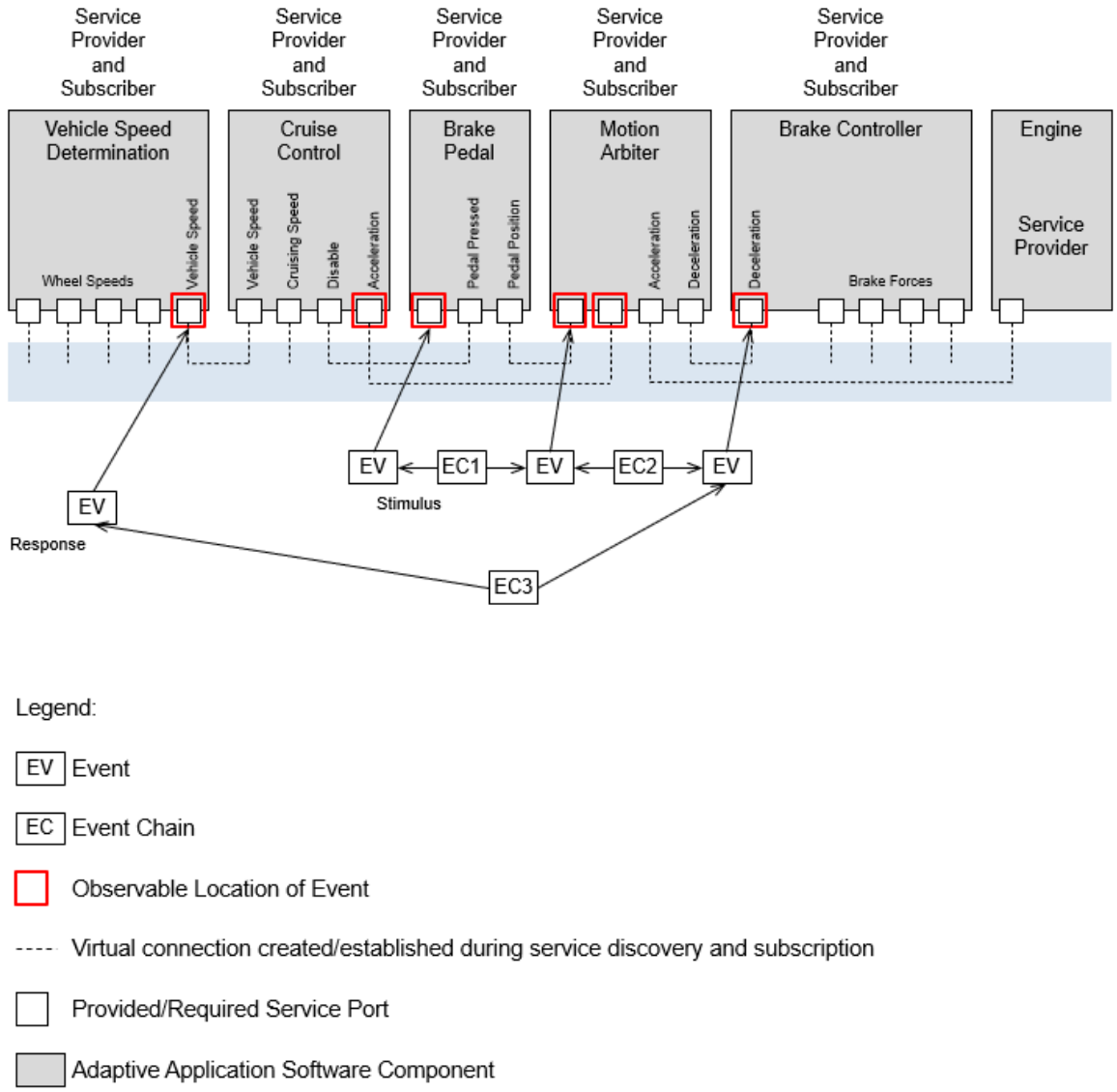


Figure 3.7: Example of the ”Sequence” pattern

An example for this pattern is depicted in Figure 3.7. The event chains *EC1* through *EC3* define a causal relationship of events observed at a port of the AA called *Brake Pedal* and a port of the AA called *Vehicle Speed Determination*.

3.5.1.3.2 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 3.8, which shows a path that forks because the AA *Brake Controller* calculates the brake force value for each wheel (*EC5* through *EC8*).

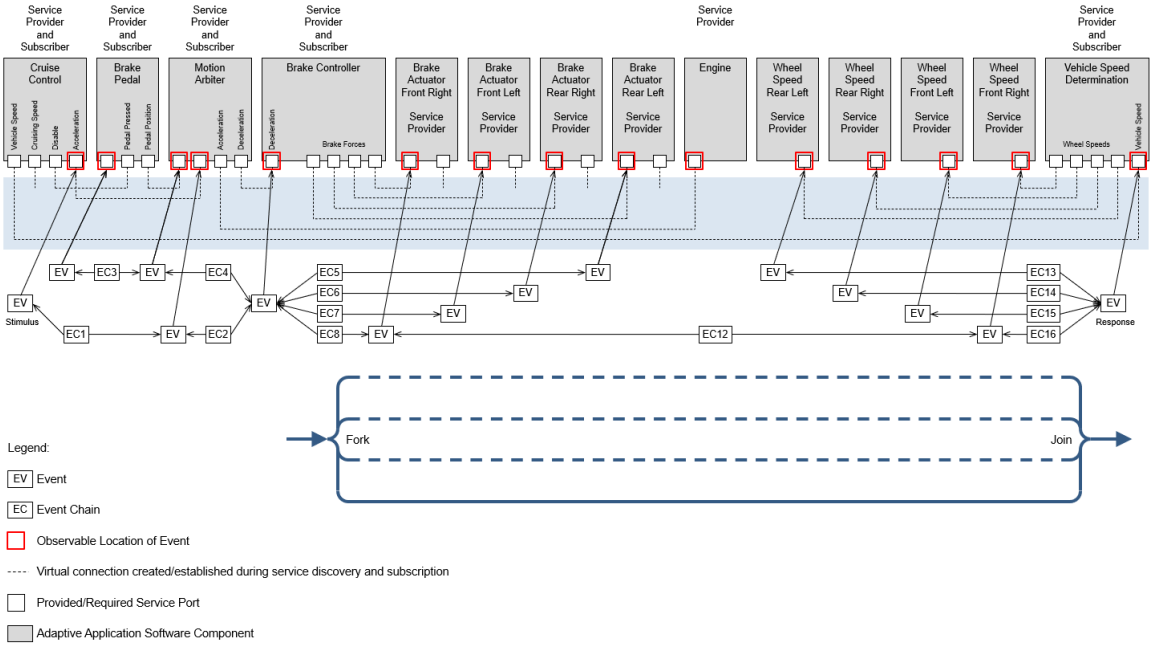


Figure 3.8: Example of the ”Fork” and ”Join” pattern

3.5.1.3.3 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 3.8 which shows a path that joins because the AA *Vehicle Speed Determination* aggregates the wheel speed values from individual wheels (*EC13* through *EC16*).

3.5.1.3.4 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 3.9 which shows that an event observed at a required port of the AA *Motion Arbiter* leads to an occurrence of an event either at the port called

Deceleration of the AA *Brake Controller*, or at the port called *Acceleration* of the AA *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 AA *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.

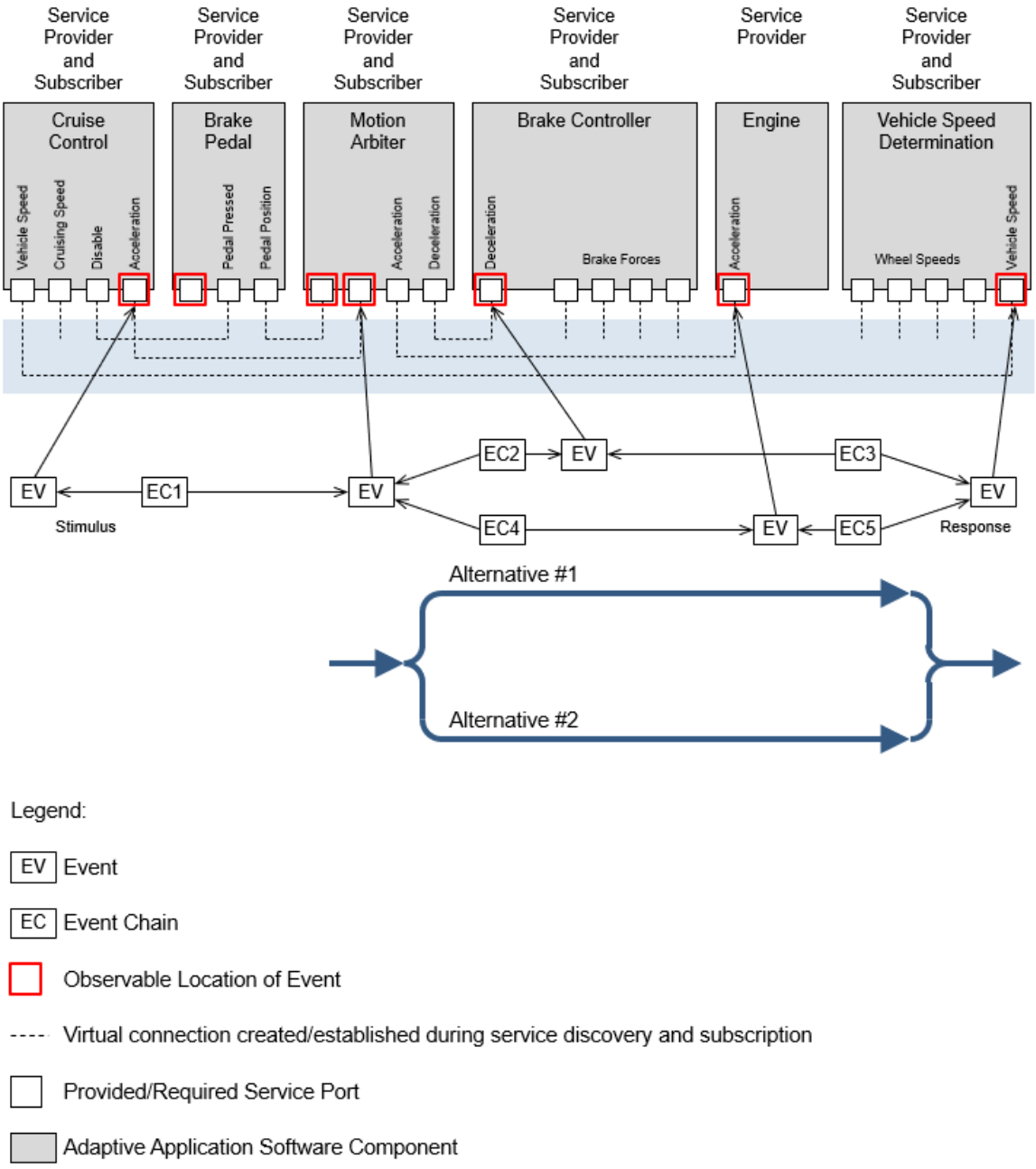


Figure 3.9: Example of the "Alternative" pattern

3.5.1.3.5 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 3.10 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_4581]. As a consequence a cycle consists of at least two event chains.

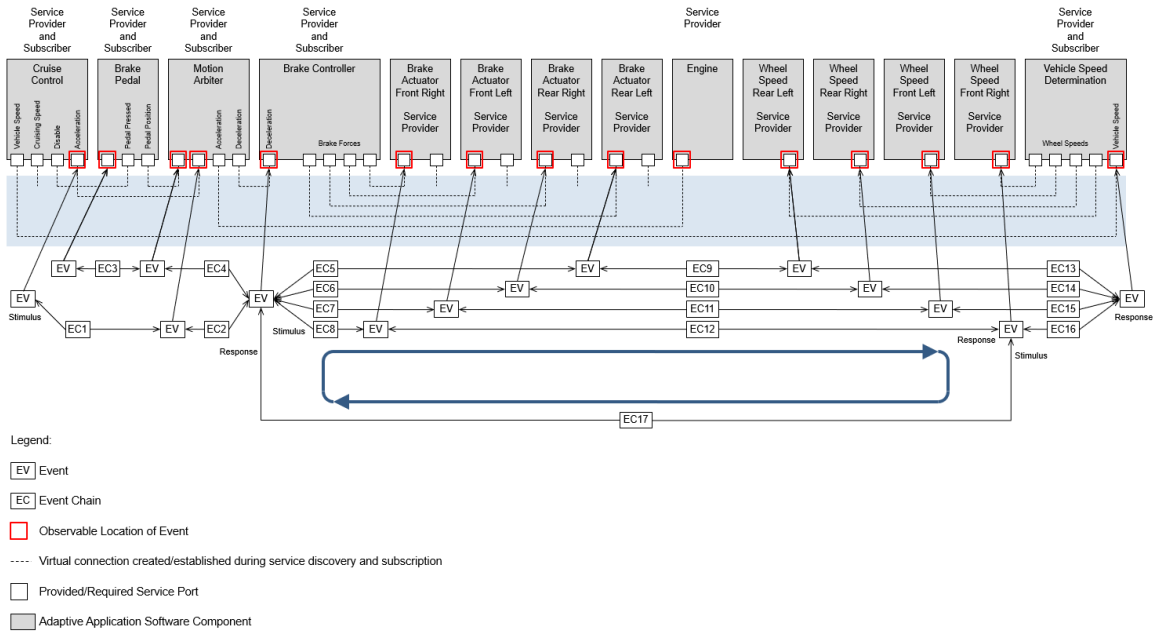


Figure 3.10: Example of the ”Cycle” pattern

3.5.2 TimingDescriptionEvent

[TPS_TIMEX_00069] Purpose of `TimingDescriptionEvent`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001

⌈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.⌋

For example, this can be the start of a service or the different steps in executing an executable.

An overview of the different event types is given in Figure 3.11. These are described in more detail in the following sub-sections.

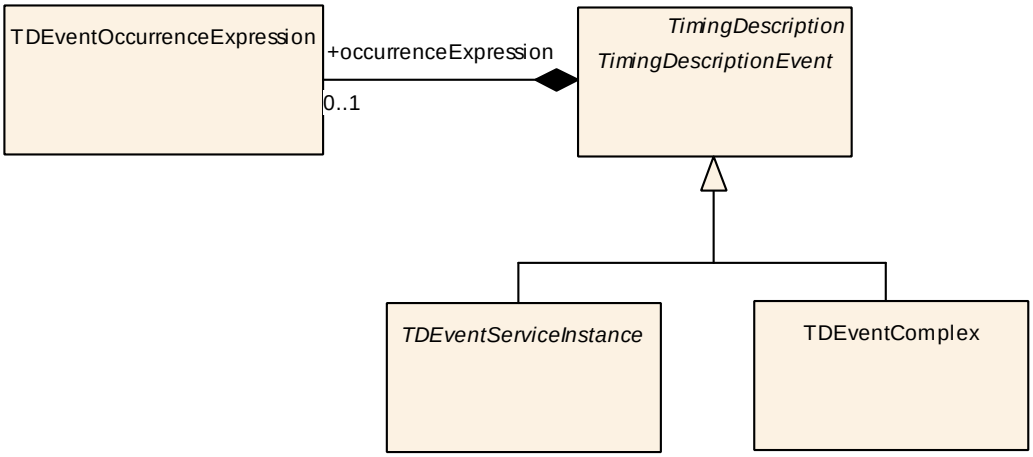


Figure 3.11: Overview of the different types of timing events

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

[TPS_TIMEX_00094] Standardized `category`s of `TimingDescriptionEvent` in Adaptive Platform

Status: DRAFT

⌈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.
- `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

⌋

Please note constraint: [constr_4559] in [6] shall apply here also.

Also note that information regarding the occurrence of a `TimingDescriptionEvent` is described separately in 3.6.1.

3.5.2.1 TDEventVfb

[TPS_TIMEX_00082] Purpose of `TDEventVfb`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001

⌈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.⌋

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

- `VfbTiming` 3.1.1
- `SystemTiming` 3.1.3

Class	***TDEventVfb*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventVfb			
Note	A `TimingDescriptionEvent` occuring on a Virtual Functional Bus (VFB) `PortPrototype`.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	*TDEventVfbPort*, TDEventVfbPortGroup, TDEventVfbReference			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
component	SwComponentPrototype	0..1	iref	The context for the scope of this timing event. **InstanceRef implemented by:** ComponentInCompositionInstanceRef

Table 3.6: TDEventVfb

[TPS_TIMEX_00092] Purpose of `TDEventVfbPort`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00019

⌈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.⌋

Class	***TDEventVfbPort*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescriptionEvents::TDEventVfb			
Note	A `TimingDescriptionEvent` occuring on a `PortPrototype`.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	TDEventModeDeclaration, TDEventOperation, TDEventTrigger, TDEventVariableDataPrototype			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	***TDEventVfbPort*** (abstract)			
port	PortPrototype	0..1	ref	port on which the TimingEvent shall apply **Tags:** atp.Status=obsolete
portPrototype	PortPrototype	0..1	iref	PortPrototype on which the TimingEvent occurs **Tags:** atp.Status=draft xml.typeElement=true **InstanceRef implemented by:** PortInCompositionType InstanceRef
portPrototype Blueprint	PortPrototypeBlueprint	0..1	ref	port on which the TimingEvent shall apply (in the context of an AUTOSAR blueprint)

Table 3.7: TDEventVfbPort

[TPS_TIMEX_00093] Purpose of TDEventVfbReference

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00019

⌈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.⌋

Class	**TDEventVfbReference**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	Reference to "other" TimingDescriptionEvents. These other TimingDescriptionEvents may be specified in other views and re-used in this view.			
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 3.8: TDEventVfbReference

[TPS_TIMEX_00083] TDEventVariableDataPrototype specifies events observable at sender/receiver ports

Status: DRAFT
Upstream requirements: RS_TIMEX_00001

⌈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.⌋

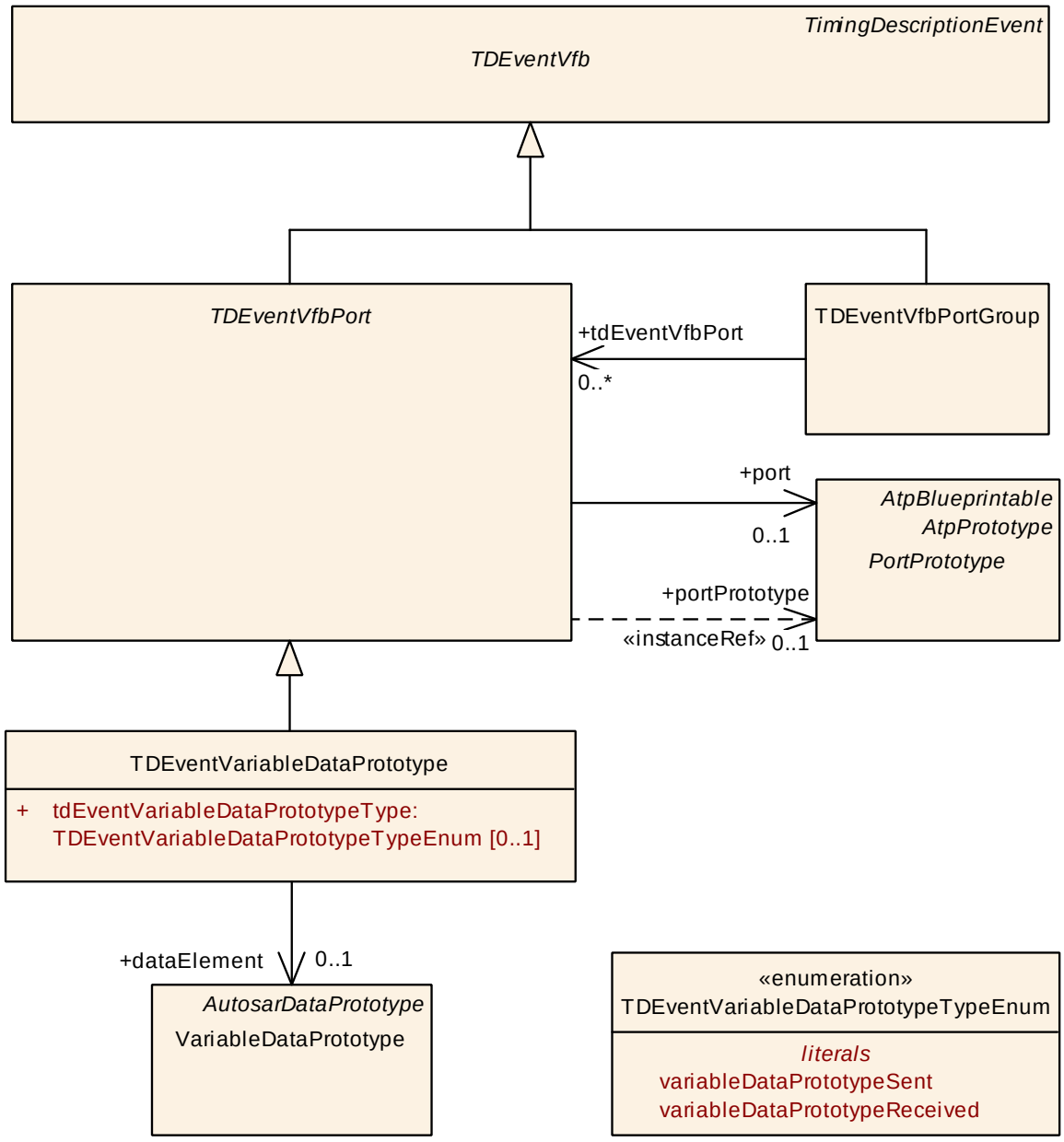


Figure 3.12: Variable Data Prototype

Class	**TDEventVariableDataPrototype**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	A `TimingDescriptionEvent` triggered by the sending/receiving of a `VariableDataPrototype` in a `SenderReceiverInterface` on VFB level.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventVfb*, *TDEventVfbPort*, *Timing Description*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	VariableDataPrototype	0..1	ref	The referenced `VariableDataPrototype` from a `SenderReceiverInterface`.

△

Class	**TDEventVariableDataPrototype**			
tdEventVariable DataPrototype Type	TDEventVariableData PrototypeTypeEnum	0..1	attr	The specific type of this timing event.

Table 3.9: TDEventVariableDataPrototype

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 3.10: TDEventVariableDataPrototypeTypeEnum

[TPS_TIMEX_00084] `TDEventOperation` specifies events observable at client/server ports.

Status: DRAFT
Upstream requirements: RS_TIMEX_00001

⌈The element `TDEventOperation` is used to specify events, namely the invocation of operations and their completion, observable at required and provided client/server ports.⌋

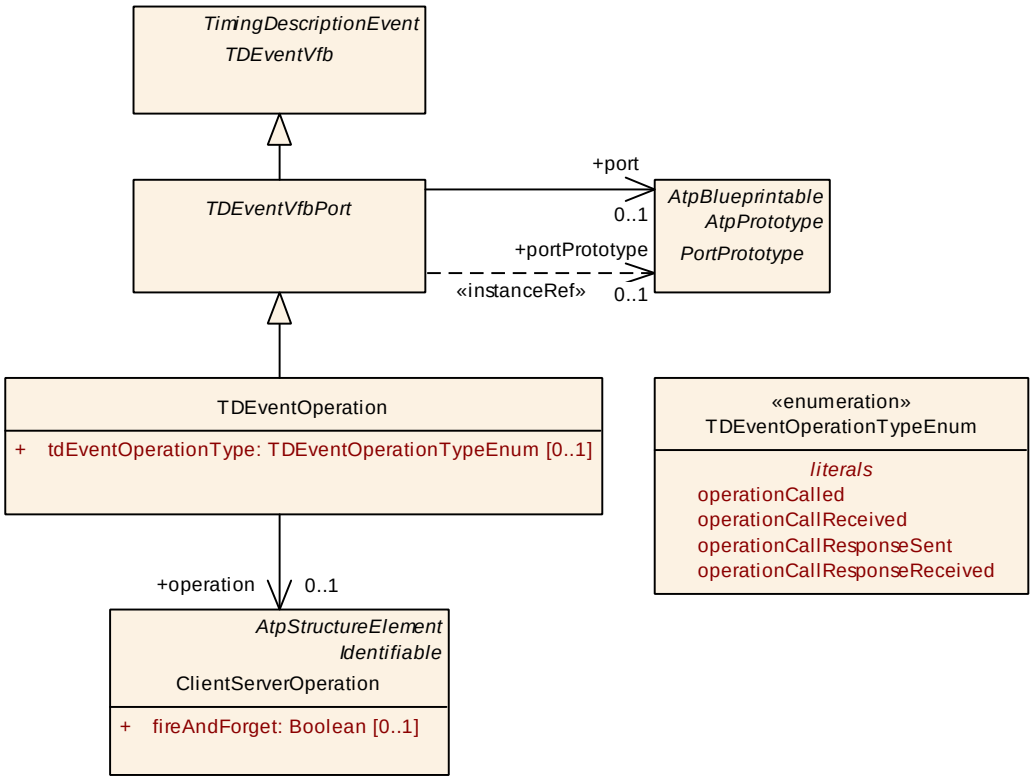


Figure 3.13: Operation

Class	**TDEventOperation**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	A `TimingDescriptionEvent` triggered by the sending/receiving of a `ClientServerOperation` in a `ClientServerInterface` on 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 `ClientServerOperation` from a `ClientServerInterface`.
tdEvent OperationType	TDEventOperationType Enum	0..1	attr	The specific type of this timing event.

Table 3.11: TDEventOperation

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

△

Enumeration	TDEventOperationTypeEnum
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 3.12: TDEventOperationTypeEnum

[TPS_TIMEX_00085] `TDEventModeDeclaration` specifies events observable at mode ports.

Status: DRAFT
Upstream requirements: RS_TIMEX_00001

⌈The element `TDEventModeDeclaration` is used to specify events, namely initiation and propagation of mode changes, observable at required and provided mode ports.⌋

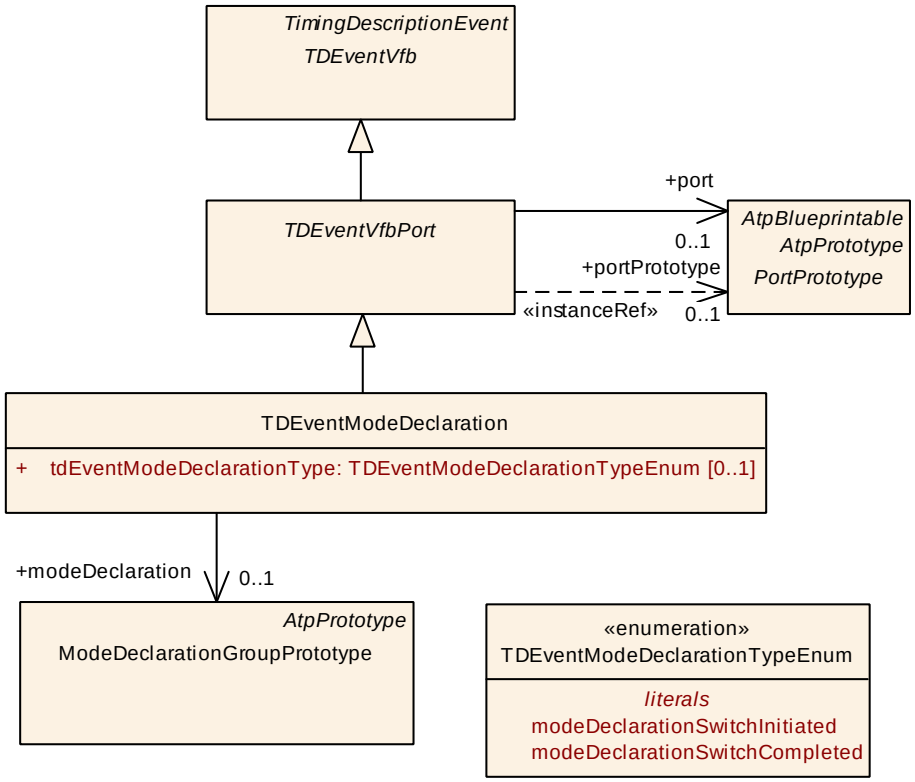


Figure 3.14: Mode Declaration

Class	**TDEventModeDeclaration**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	A `TimingDescriptionEvent` triggered by a mode switch in a `ModeSwitchInterface` on 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 `ModeDeclarationGroupPrototype` from a{ModeSwitchInterface}].
tdEventMode DeclarationType	TDEventMode DeclarationTypeEnum	0..1	attr	The specific type of this timing event.

Table 3.13: TDEventModeDeclaration

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 3.14: TDEventModeDeclarationTypeEnum

[TPS_TIMEX_00090] `TDEventTrigger` specifies events observable at trigger ports

Status: DRAFT
Upstream requirements: RS_TIMEX_00001

⌈The element `TDEventTrigger` is used to specify events, namely the activation and release of triggers, observable at required and provided trigger ports.⌋

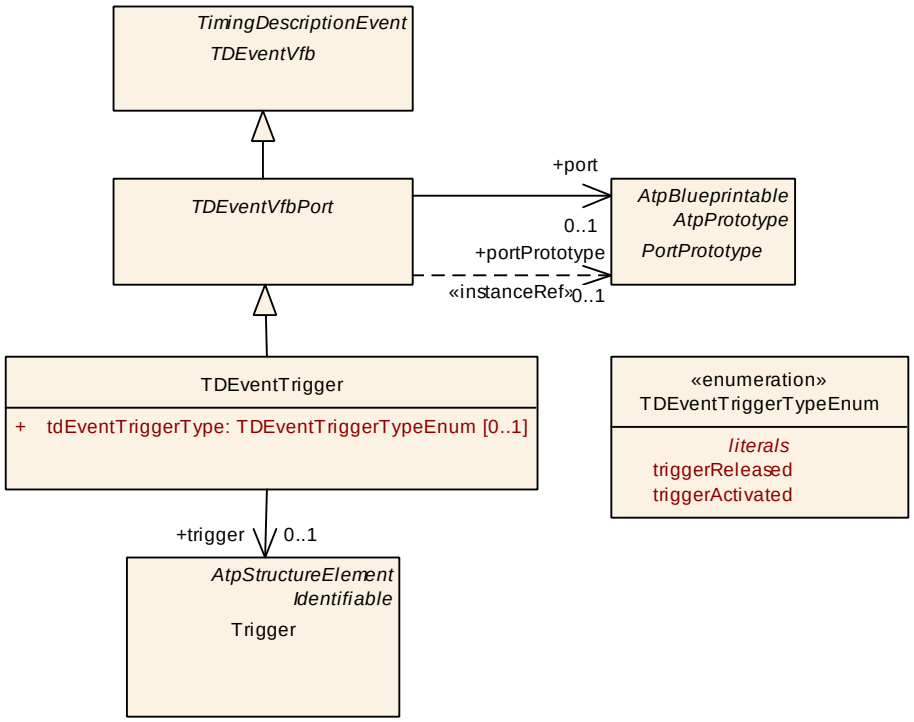


Figure 3.15: Trigger

Class	**TDEventTrigger**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventVfb			
Note	A `TimingDescriptionEvent` triggered by a `Trigger` in a `TriggerInterface` on 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 referenced `Trigger` from a `TriggerInterface`.

Table 3.15: TDEventTrigger

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 3.16: TDEventTriggerTypeEnum

3.5.2.2 TDEventServiceInstance

[TPS_TIMEX_00058] Purpose of `TDEventServiceInstance`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00024

⌈The element `TDEventServiceInstance` and its specializations are used to describe the occurrences of an event which are observed at a specific location in the Service view.⌋

Events related to the adaptive service can be used during the specification of:

- `VfbTiming` 3.1.1
- `SystemTiming` 3.1.3
- `ServiceTiming` 3.1.4

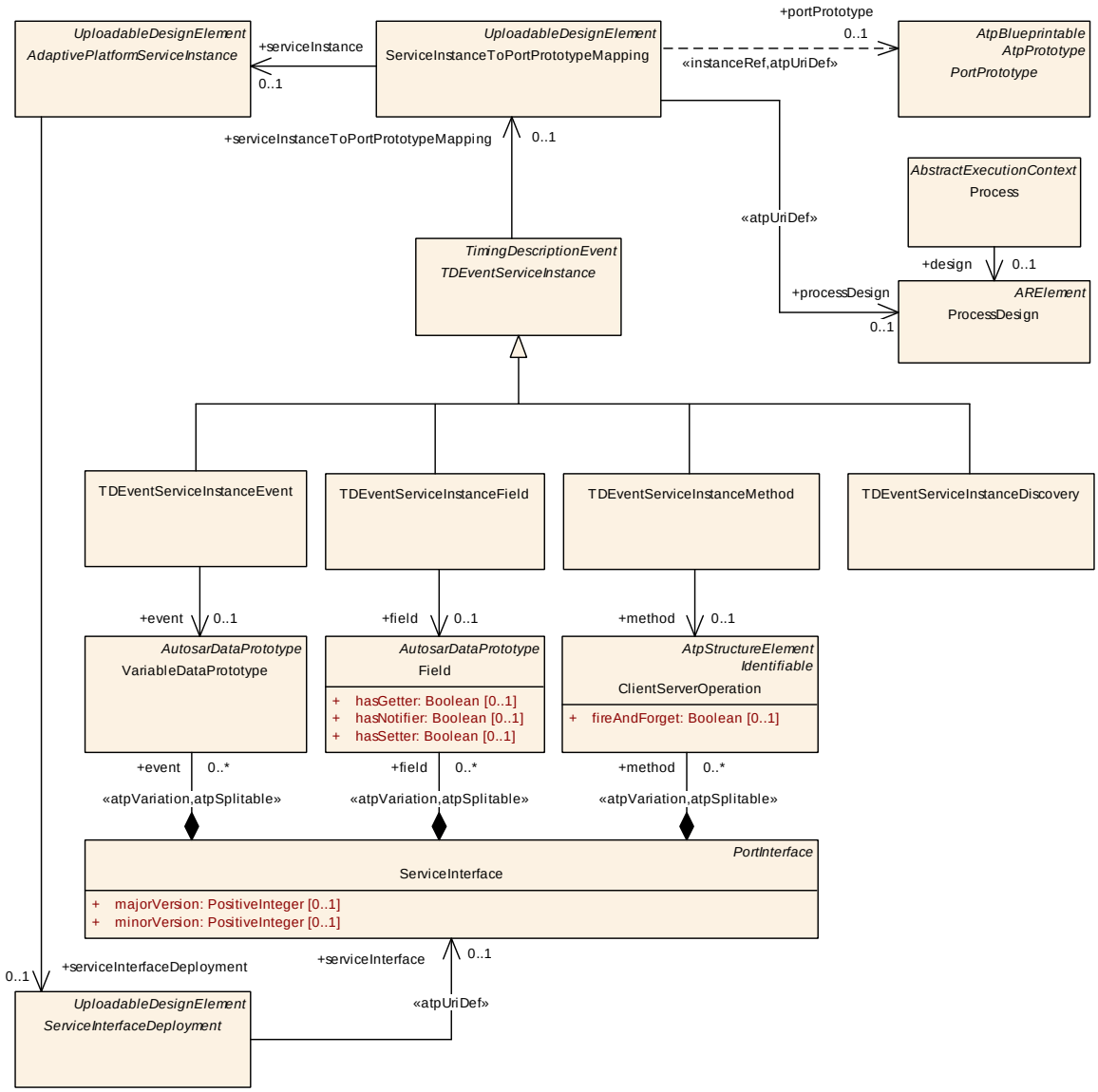


Figure 3.16: Adaptive Service events

Class	***TDEventServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingDescription::TimingDescription Events::TDEventServiceInstance			
Note	This is the abstract parent class to describe specific timing description event types for service-oriented communication. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*, *TimingDescriptionEvent*			
Subclasses	TDEventServiceInstanceDiscovery, TDEventServiceInstanceEvent, TDEventServiceInstanceField, TD EventServiceInstanceMethod			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	***TDEventServiceInstance*** (abstract)			
serviceInstance ToPortPrototype Mapping	ServiceInstanceToPort PrototypeMapping	0..1	ref	The service and the port this service is provided. **Tags:** atp.Status=draft

Table 3.17: TDEventServiceInstance

[TPS_TIMEX_00059] Purpose of `TDEventServiceInstanceEvent`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00024

⌈The element `TDEventServiceInstanceEvent` is used to describe the occurrences of an event which are observed at a specific location in the Service view.⌋

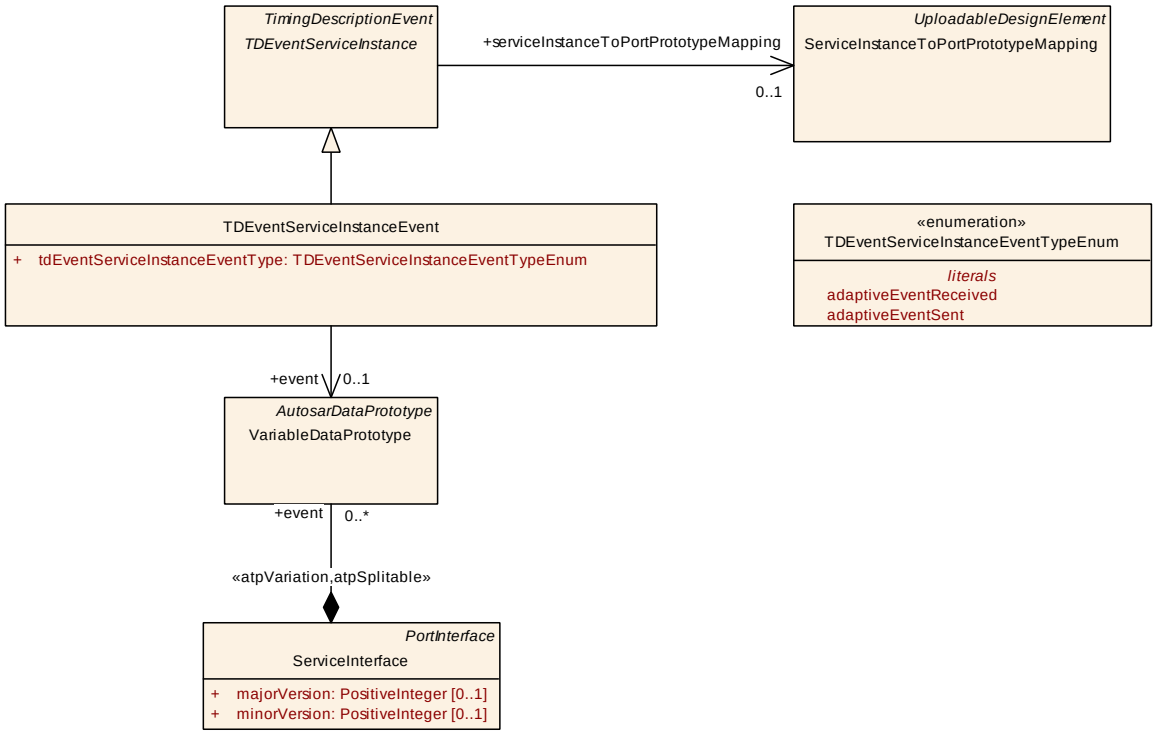


Figure 3.17: Adaptive Service Event

Class	**TDEventServiceInstanceEvent**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingDescription::TimingDescription Events::TDEventServiceInstance::TDEventServiceInstanceEvent			
Note	This is used to describe timing description events related to events of a service. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventServiceInstance*, *Timing Description*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**TDEventServiceInstanceEvent**			
event	VariableDataPrototype	0..1	ref	The event provided by the service. **Tags:** atp.Status=draft
tdEventService InstanceEvent Type	TDEventService InstanceEventType Enum	1	attr	The specific type of this timing event. **Tags:** atp.Status=draft

Table 3.18: TDEventServiceInstanceEvent

Enumeration	**TDEventServiceInstanceEventTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingDescription::TimingDescription Events::TDEventServiceInstance::TDEventServiceInstanceEvent
Note	This is used to describe the specific event type of a TDEventServiceInstanceEvent. **Tags:** atp.Status=draft
Aggregated by	TDEventServiceInstanceEvent.tdEventServiceInstanceEventType
Literal	***Description***
adaptiveEvent Received	A point in time where an event required by a service subscriber is received through the service port of the service subscriber. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=draft
adaptiveEventSent	A point in time where an event provided by a service is sent through the service port of the service provider. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=draft

Table 3.19: TDEventServiceInstanceEventTypeEnum

[TPS_TIMEX_00060] Purpose of `TDEventServiceInstanceField`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00024

⌈The element `TDEventServiceInstanceField` is used to describe the occurrences of an event which are observed at a specific location in the Service view.⌋

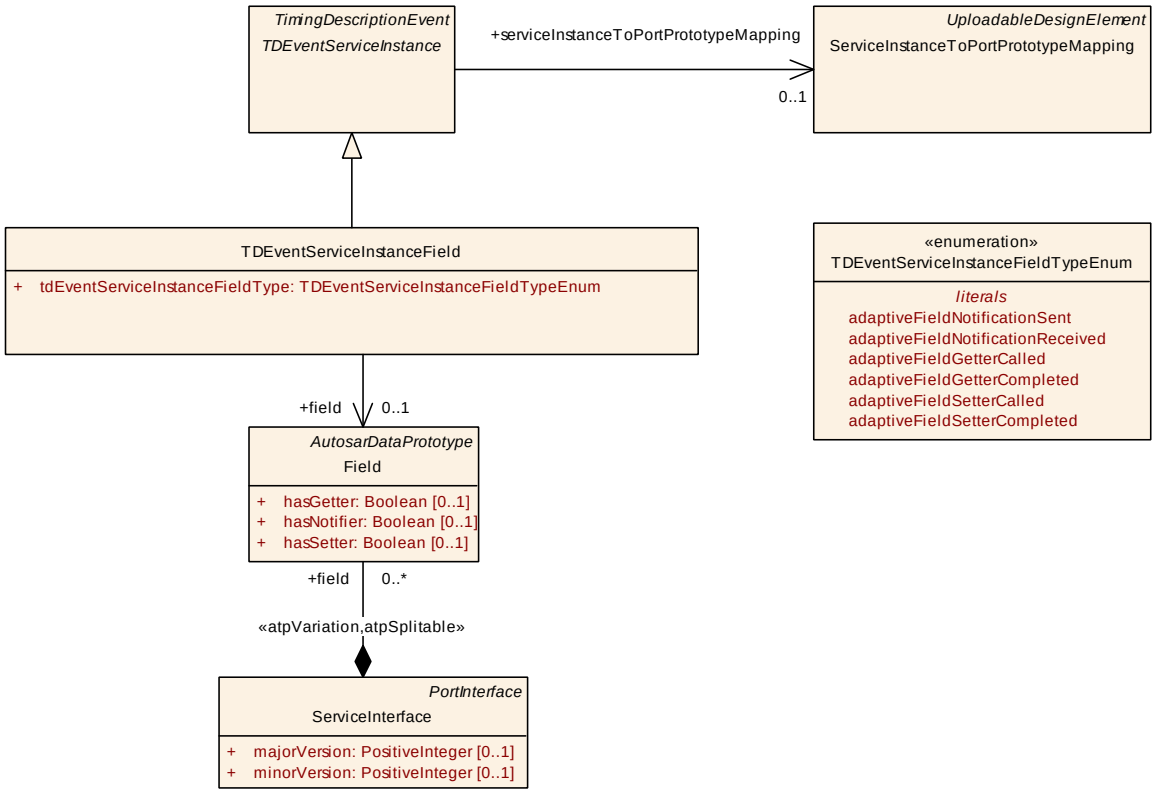


Figure 3.18: Adaptive Service Field

Class	**TDEventServiceInstanceField**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingDescription::TimingDescriptionEvents::TDEventServiceInstance::TDEventServiceInstanceField			
Note	This is used to describe timing description events related to fields of a service. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventServiceInstance*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
field	Field	0..1	ref	The field provided by the service. **Tags:** atp.Status=draft
tdEventServiceInstanceFieldType	TDEventServiceInstanceFieldTypeEnum	1	attr	The specific type of this timing event. **Tags:** atp.Status=draft

Table 3.20: TDEventServiceInstanceField

Enumeration	**TDEventServiceInstanceFieldTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingDescription::TimingDescriptionEvents::TDEventServiceInstance::TDEventServiceInstanceField
Note	This is used to describe the specific event type of a TDEventServiceInstanceField. **Tags:** atp.Status=draft
Aggregated by	TDEventServiceInstanceField.tdEventServiceInstanceFieldType

△

Enumeration	**TDEventServiceInstanceFieldTypeEnum**
Literal	***Description***
adaptiveFieldGetter Called	A point in time where a field getter of a service is called by a service subscriber through the service port of the service subscriber. **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=draft
adaptiveFieldGetter Completed	A point in time where a field getter of a service is completed and the result of the field getter is received through the service subscriber's service port. **Tags:** atp.EnumerationLiteralIndex=3 atp.Status=draft
adaptiveField Notification Received	A point in time where a field notification required by a service subscriber is received through the service port of the service subscriber. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=draft
adaptiveField NotificationSent	A point in time where a field notification provided by a service is sent through the service port of the service provider. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=draft
adaptiveFieldSetter Called	A point in time where a field setter of a service is called by a service subscriber through the service port of the service subscriber. **Tags:** atp.EnumerationLiteralIndex=4 atp.Status=draft
adaptiveFieldSetter Completed	A point in time where a field setter of a service is completed and the result of the field setter is received through the service subscriber's service port. **Tags:** atp.EnumerationLiteralIndex=5 atp.Status=draft

Table 3.21: TDEventServiceInstanceFieldTypeEnum

[TPS_TIMEX_00061] Purpose of `TDEventServiceInstanceMethod`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00024

⌈The element `TDEventServiceInstanceMethod` is used to describe the occurrences of an event which are observed at a specific location in the Service view.⌋

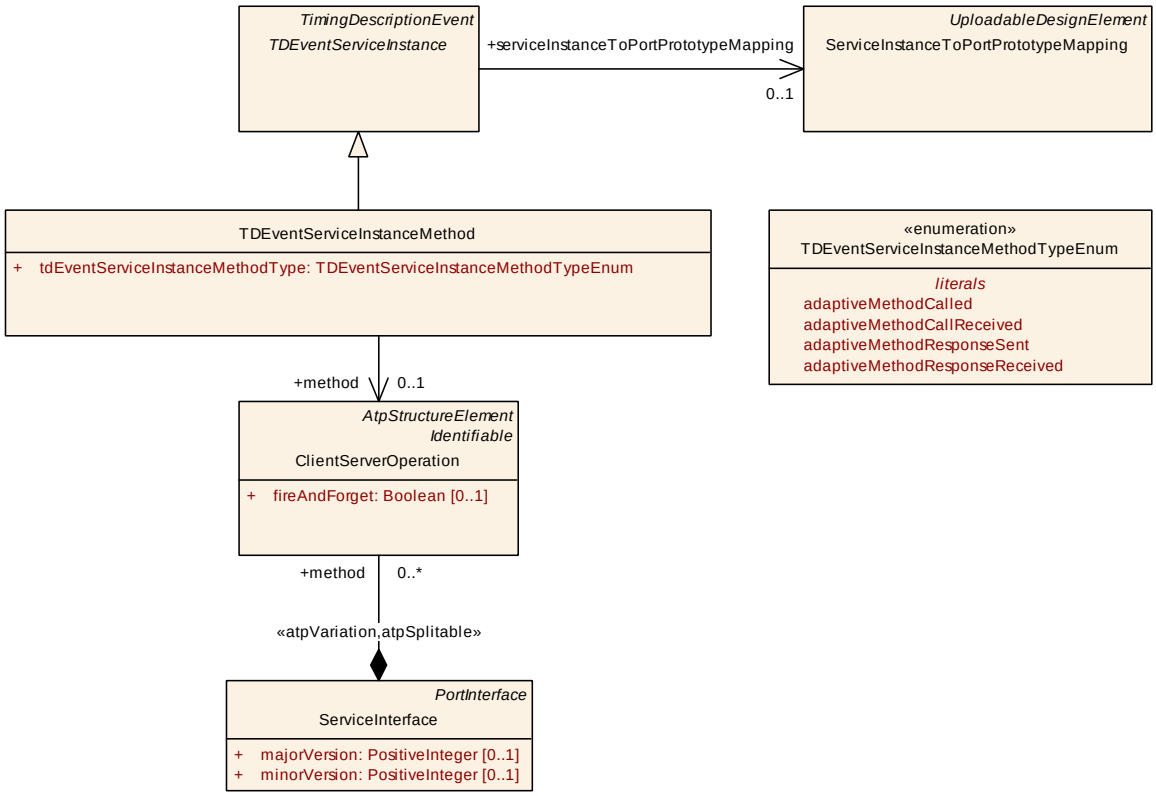


Figure 3.19: Adaptive Service Method

Class	**TDEventServiceInstanceMethod**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingDescription::TimingDescriptionEvents::TDEventServiceInstance::TDEventServiceInstanceMethod			
Note	This is used to describe timing description events related to methods of a service. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventServiceInstance*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
method	ClientServerOperation	0..1	ref	The method provided by the service. **Tags:** atp.Status=draft
tdEventServiceInstanceMethodType	TDEventServiceInstanceMethodTypeEnum	1	attr	The specific type of this timing event. **Tags:** atp.Status=draft

Table 3.22: TDEventServiceInstanceMethod

Enumeration	**TDEventServiceInstanceMethodTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingDescription::TimingDescriptionEvents::TDEventServiceInstance::TDEventServiceInstanceMethod
Note	This is used to describe the specific event type of a TDEventServiceInstanceMethod. **Tags:** atp.Status=draft
Aggregated by	TDEventServiceInstanceMethod.tdEventServiceInstanceMethodType

△

Enumeration	**TDEventServiceInstanceMethodTypeEnum**
Literal	***Description***
adaptiveMethod Called	A point in time where a method of a service is called through the service subscriber's service port. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=draft
adaptiveMethodCall Received	A point in time where a method call of a service is received through the service provider's service port. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=draft
adaptiveMethod ResponseReceived	A point in time where a response of a method call of a service is received through the service subscribers's service port. **Tags:** atp.EnumerationLiteralIndex=3 atp.Status=draft
adaptiveMethod ResponseSent	A point in time where a response of a method call of a service is sent through the service provider's service port. **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=draft

Table 3.23: TDEventServiceInstanceMethodTypeEnum

[TPS_TIMEX_00062] Purpose of `TDEventServiceInstanceDiscovery`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00024

⌈The element `TDEventServiceInstanceDiscovery` is used to describe the occurrences of an event which are observed at a specific location in the Service view.⌋

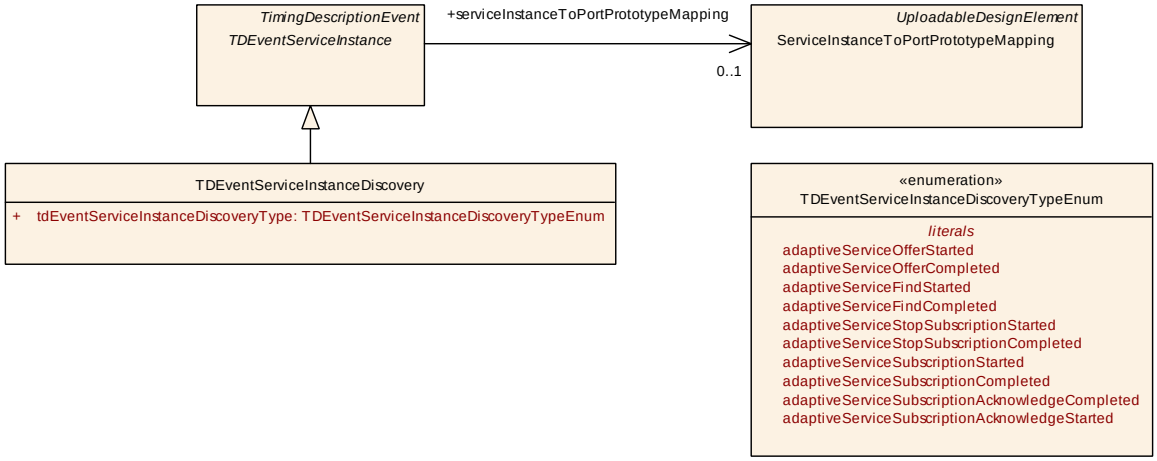


Figure 3.20: Adaptive Service Discovery

Class	**TDEventServiceInstanceDiscovery**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingDescription::TimingDescriptionEvents::TDEventServiceInstance::TDEventServiceInstanceService			
Note	This is used to describe timing description events related to different phases of service discovery. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TDEventServiceInstance*, *TimingDescription*, *TimingDescriptionEvent*			
Aggregated by	*TimingExtension*.timingDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tdEventServiceInstanceDiscoveryType	TDEventServiceInstanceDiscoveryTypeEnum	1	attr	The specific type of this timing event. **Tags:** atp.Status=draft

Table 3.24: TDEventServiceInstanceDiscovery

Enumeration	**TDEventServiceInstanceDiscoveryTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::Timing::TimingDescription::TimingDescriptionEvents::TDEventServiceInstance::TDEventServiceInstanceService
Note	This is used to describe the specific event type of a TDEventServiceInstanceDiscovery. **Tags:** atp.Status=draft
Aggregated by	TDEventServiceInstanceDiscovery.tdEventServiceInstanceDiscoveryType
Literal	***Description***
adaptiveServiceFindCompleted	A point in time where a service subscriber completes to find a needed service. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=draft
adaptiveServiceFindStarted	A point in time where a service subscriber starts to find a needed service. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=draft
adaptiveServiceOfferCompleted	A point in time where a service provider completes to offer a needed service. **Tags:** atp.EnumerationLiteralIndex=3 atp.Status=draft
adaptiveServiceOfferStarted	A point in time where a service provider starts to offer a needed service. **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=draft
adaptiveServiceStopSubscriptionCompleted	A point in time where a service subscriber completes to stop subscribing to a needed service. **Tags:** atp.EnumerationLiteralIndex=9 atp.Status=draft
adaptiveServiceStopSubscriptionStarted	A point in time where a service subscriber starts to stop subscribing to a needed service. **Tags:** atp.EnumerationLiteralIndex=8 atp.Status=draft
adaptiveServiceSubscriptionAcknowledgeCompleted	A point in time where a service provider completes to acknowledge subscription to a needed service. **Tags:** atp.EnumerationLiteralIndex=7 atp.Status=draft

△

Enumeration	**TDEventServiceInstanceDiscoveryTypeEnum**
adaptiveService Subscription Acknowledge Started	A point in time where a service provider starts to acknowledge subscription to a needed service. **Tags:** atp.EnumerationLiteralIndex=6 atp.Status=draft
adaptiveService Subscription Completed	A point in time where a service subscriber completes to subscribe to a needed service. **Tags:** atp.EnumerationLiteralIndex=5 atp.Status=draft
adaptiveService SubscriptionStarted	A point in time where a service subscriber starts to subscribe to a needed service. **Tags:** atp.EnumerationLiteralIndex=4 atp.Status=draft

Table 3.25: TDEventServiceInstanceDiscoveryTypeEnum

3.5.2.3 TDEventComplex

[TPS_TIMEX_00086] Purpose of `TDEventComplex`

Status: DRAFT
Upstream requirements: RS_TIMEX_00001

⌈The element `TDEventComplex` is used to specify relationships between occurrences of events.⌋

Complex timing events can be used during the specification of:

- `VfbTiming` 3.1.1
- `SystemTiming` 3.1.3

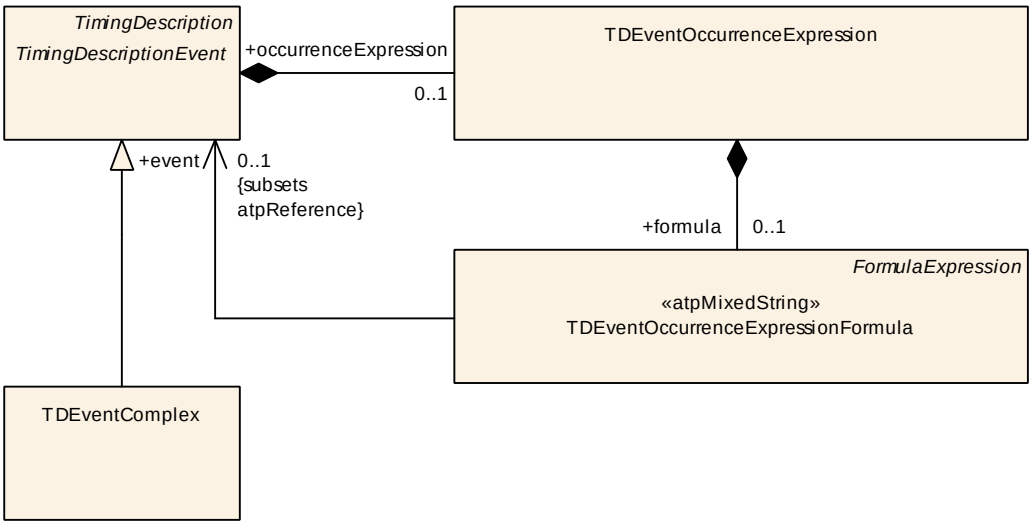


Figure 3.21: Complex timing event

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 3.26: 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 `TimingDescriptionEvent`s as describe in the following section.

3.5.2.4 TDEventSLLET

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

- `VfbTiming` 3.1.1
- `ExecutableTiming` 3.1.2
- `SystemTiming` 3.1.3
- `MachineTiming` 3.1.5

For the remaining aspects, please refer to [6] chapter "System Level Logical Execution Time".

3.5.2.5 Occurrence Expression Language for Timing Events

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

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

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

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

3.5.2.5.1 Specifying an Occurrence Expression

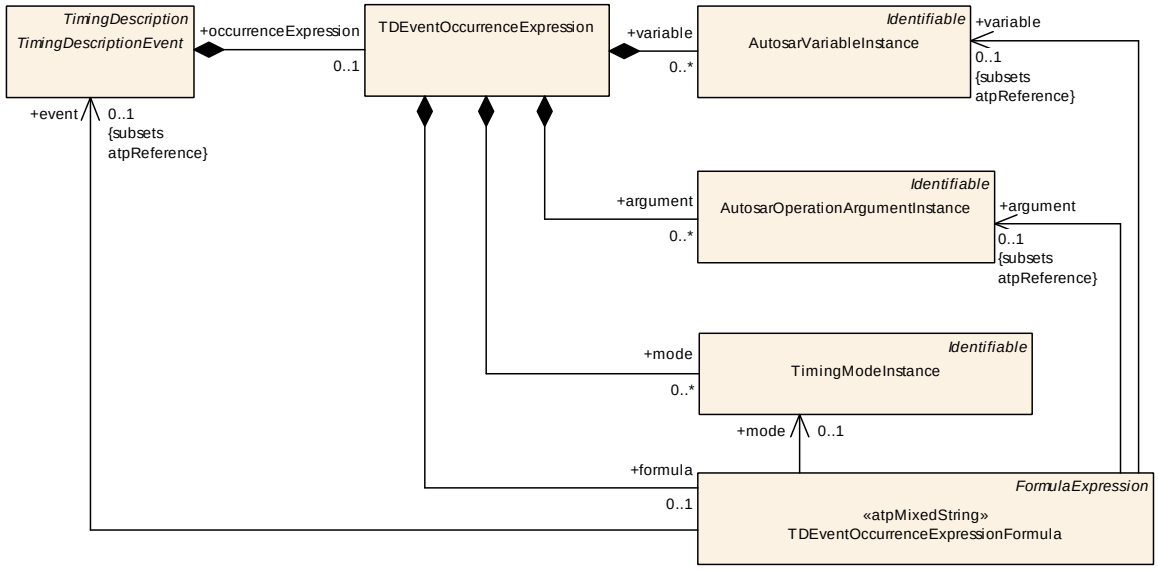


Figure 3.22: The occurrence expression

As shown in Figure 3.22, 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 [5]). The `TDEventOccurrenceExpressionFormula` uses the capabilities of the `FormulaExpression` and adds the following functions to the expression language:

- The function *TIMEX_value*, which requires as operand either:
  - a reference to an `AutosarVariableInstance` or
  - a reference to an `AutosarOperationArgumentInstance` whose value shall be evaluated.

  The return type of this function is `Numerical` (see constraint [constr_4551]).
- The function *TIMEX_occurs*, which requires as operand a reference to the `TimingDescriptionEvent` whose occurrence shall be evaluated. The return type of this function is `Boolean`. It returns TRUE if the referenced timing event occurs at the point in time the expression is evaluated.
- The function *TIMEX_hasOccurred*, which requires as operand a reference to the `TimingDescriptionEvent` whose occurrence shall be evaluated. The return type of this function is `Boolean`. It returns TRUE if the referenced timing event

has occurred *at least once* before or at the same point in time the expression is evaluated.

- The function *TIMEX_timeSinceLastOccurrence*, which requires as operand a reference to the `TimingDescriptionEvent` whose occurrence shall be evaluated. The return type of this function is `Float` and the unit is seconds. It returns the time difference between the point in time of the last occurrence of the referenced event and the point in time the expression is evaluated.
- The function *TIMEX_angleSinceLastOccurrence*, which requires as operand a reference to the `TimingDescriptionEvent` whose occurrence shall be evaluated. The return type of this function is `Float` and the unit is degree. It returns the angle of the crank shaft between the point in time of the last occurrence of the referenced event and the point in time the expression is evaluated.
- The function *TIMEX_modeActive* queries the `TimingModeInstance` specified as argument. The return type of this function is `Boolean`. It returns TRUE if the specified mode declaration is *active* at the point in time the expression is evaluated, otherwise it returns FALSE.

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_4569] Restricted usage of Occurence Expression functions

Status: DRAFT

⌈The functions:

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

shall only be used for an occurrence expression applied to a `TDEventComplex`.⌋

[constr_4570] Application rule for the occurrence expression in `TDEventComplex`

Status: DRAFT

⌈The occurrence expression shall be specified such that it describes an *event* rather than a state. As a consequence the occurrence expression shall ensure that a complex timing event *could* only occur at the occurrence time of one of the referenced `TimingDescriptionEvent`s.⌋

[constr_4571] Use references only as function operands

Status: DRAFT

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

[constr_4591] Use only Numericals in `TDEventOccurrenceExpression`

Status: DRAFT

⌈The target data prototype of the instance references of `variable` and `argument` shall be `Numerical`.⌋

The example given below shows how to combine the functions introduced above in order to specify an occurrence expression for a complex event called *EC*.

Figure 3.23 sketches the AUTOSAR software component model of this example.

A software component named *Swc1* has a required port, called *RequiredPort*, and a provided port, called *ProvidedPort*. Both ports are sender-receiver ports. The sender-receiver port interface of the required port is called *SenderReceiverInterface1*, and consists of three data elements: The first data element is called *DE1*, the second data element is called *DE2*, and the third data element is called *DE3*. Note, that alternatively it would be also possible to define three required sender-receiver ports and the port interface of each of those ports consists of one of the data elements.

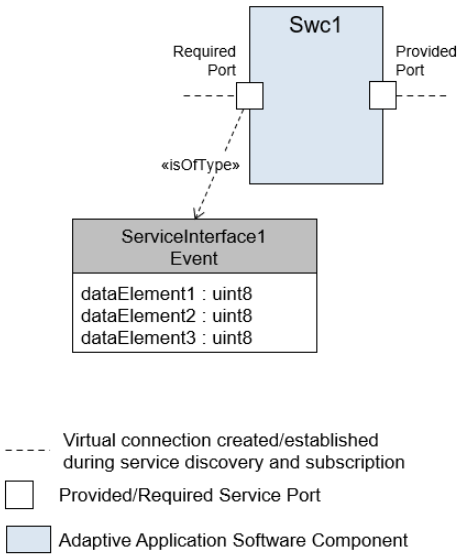


Figure 3.23: The SWC used by the Occurrence Expression Example

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. And this timing model shall only contain timing description events related to the Virtual Functional Bus as described in section 3.5.2.1.

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

```
// Condition 1
( TIMEX_occurs( /example/expression/E1 )
  || TIMEX_occurs( /example/expression/E2 ) )
// Condition 2
&& TIMEX_value( /example/expression/EC/DE3 ) > 3
// Condition 3
&& abs( TIMEX_timeSinceLastOccurrence( /example/expression/E1 ) -
   TIMEX_timeSinceLastOccurrence( /example/expression/E2 ) ) <= 0.0005
```

Listing 3.1: Event Occurrence Filter

Due to the first condition the complex event *EC* can only occur when one of the atomic timing events *E1* or *E2* occurs at the point in time of evaluation. Thus, this expression satisfies the semantics constraint defined in [constr_4570]. Figure 3.26 shows a measurement of the event occurrences.

The corresponding AUTOSAR ARXML file fragment for the complex event *EC* 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 `TimingDescriptionEvent`s 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 3.27: TDEventOccurrenceExpression

Class	«atpMixedString» **TDEventOccurrenceExpressionFormula**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingDescription::TimingDescription Events::TDEventOccurrenceExpression			
Note	This is an extension of the FormulaExpression for the AUTOSAR Timing Extensions. A TDEventOccurrenceExpressionFormula provides the means to express the temporal characteristics of timing event occurrences in correlation with specific variable and argument values. The formal definition of the extended functions (ExtUnaryFunctions) is described in detail in the AUTOSAR Timing Extensions.			
Base	*ARObject*, *FormulaExpression*			
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.

▽

△

Class	«atpMixedString» **TDEventOccurrenceExpressionFormula**			
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 3.28: 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 3.29: AutosarVariableInstance

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 3.30: AutosarOperationArgumentInstance

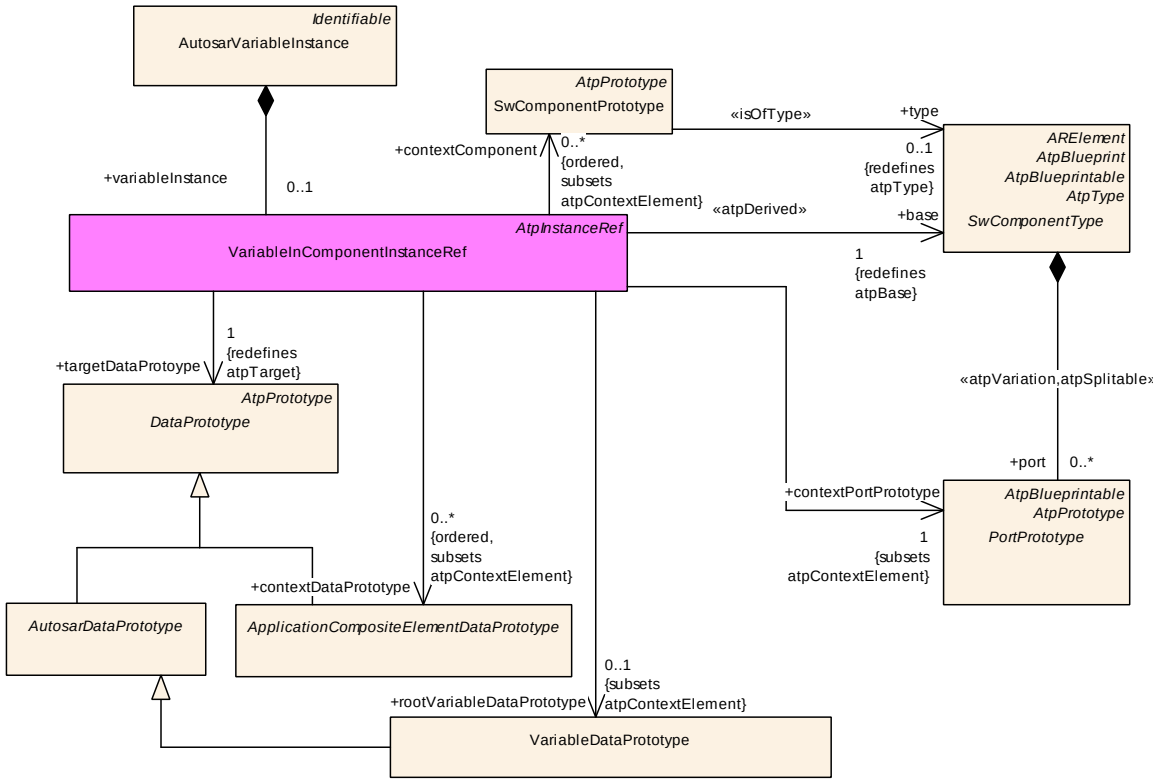


Figure 3.24: The required context information to reference a variable instance within AUTOSAR.

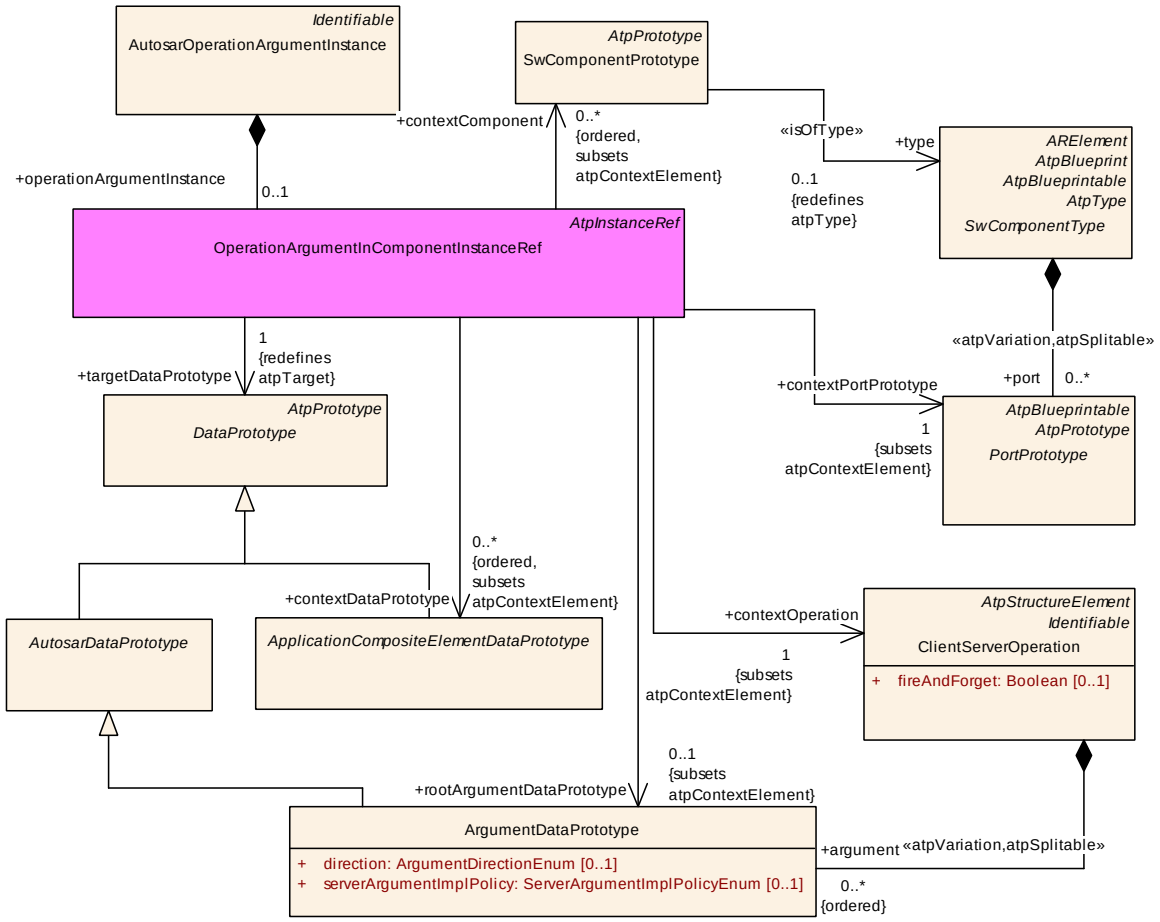


Figure 3.25: The required context information to reference an operation argument instance within AUTOSAR.

3.5.2.6 Occurrence Expression Language Syntax

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

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

3.5.2.6.1 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.

3.5.2.6.1.1 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_4592] Restricted usage of AutosarVariableInstance for Content Filter

Status: DRAFT

⌈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_4572] Restricted usage of AutosarOperationArgumentInstance for Content Filter

Status: DRAFT

⌈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).⌋

3.5.2.6.1.2 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 3.5.2.5.1, the interpreter "implements" a function *EC(t)* which returns TRUE, if the complex event EC occurs at time *t*:

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

Since the expression satisfies [constr_4570], it shall only be evaluated at occurrence times of *E1* or *E2*, because only then the complex event *EC* can occur and the expression can return TRUE.

As shown in the sketched trace in Figure 3.26 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.

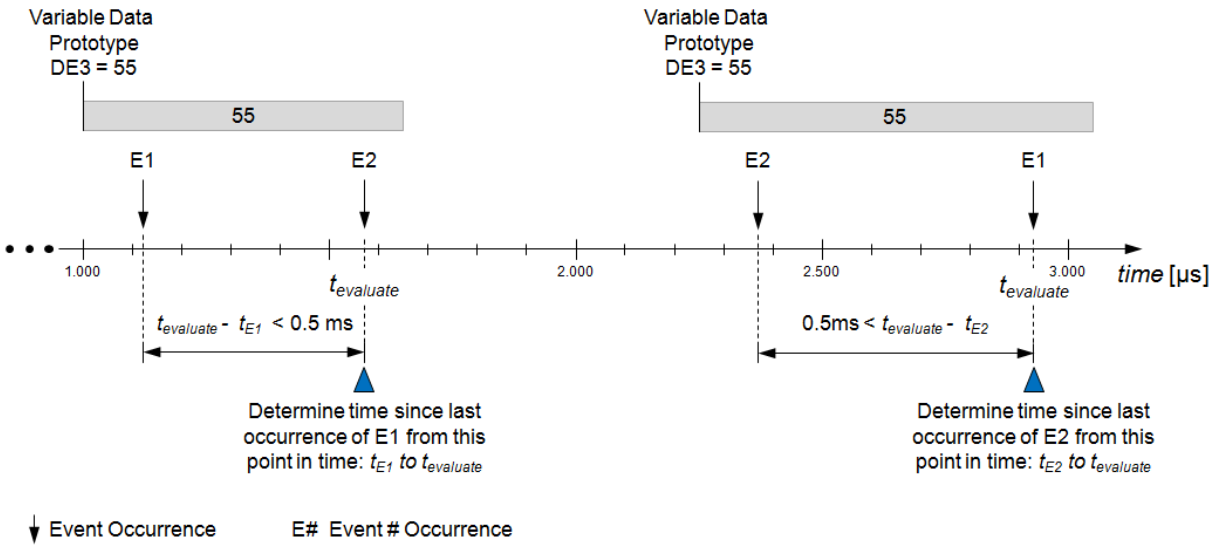


Figure 3.26: Trace showing various occurrences of the timing description events *E1* and *E2*, as well as the value of the variable *DE3*.

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

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

There are different ways to gather the required information:

- Model analysis and simulation: In a deterministic system environment, occurrences of `TimingDescriptionEvent`s can be determined offline, for example the point in time a frame will be transmitted in the static segment of a FlexRay network.
- Target trace: The required information can be gathered from a running system by recording the points in time a `TimingDescriptionEvent` has occurred.

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).

3.5.2.7 Time Base Referencing for Timing Description Events

Please refer to [6] chapter "Time Base Referencing for Timing Description Events".

3.6 TimingConstraint

Timing constraints can be applied either on:

- TimingDescriptionEvent: classifies a single event or a group of events with a temporal restriction, for example a period, a latency or a time interval considered as synchronous. Also the direction has to be considered, which means in the semantics of the constraint it matters whether an event source (forward semantics) or an event sink (backward semantics) is considered.
- TimingDescriptionEventChain: a condition or property for this event chain is set. As the event chain has a semantic of a directed acyclic graph, the direction is obvious, but it matters whether a single event chain or a group of event chains are constrained.

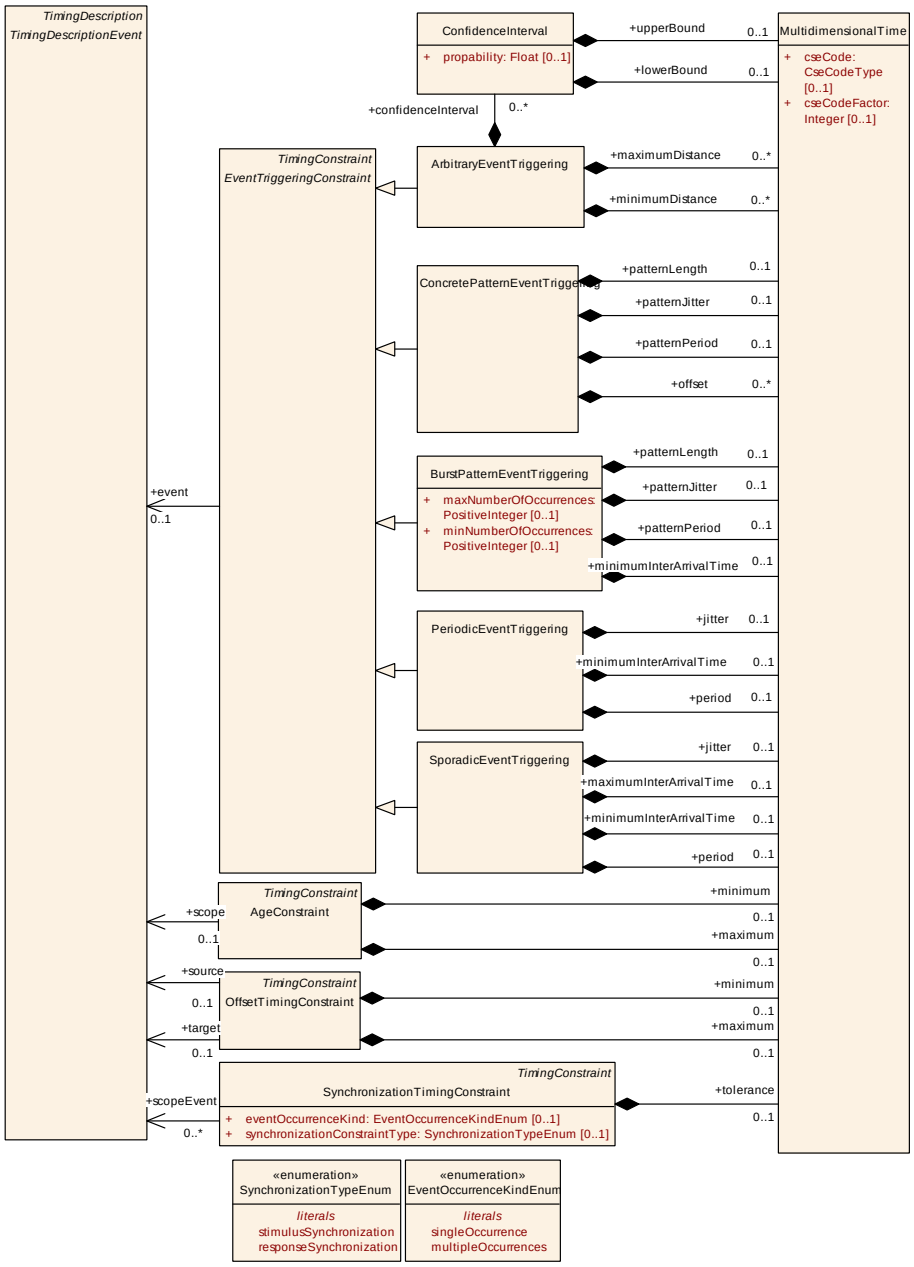


Figure 3.27: `TimingConstraint` vs `TimingDescriptionEvent`

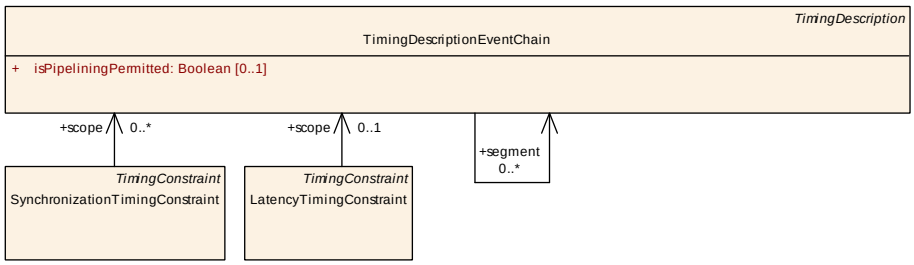


Figure 3.28: `TimingConstraint` vs `TimingDescriptionEventChain`

Mentioned in context of a requirement specification, Timing Constraints can be used as functional requirements and therefore can be tested. For usage in context of a

performance specification, Timing Constraints can be used as system properties or timing guarantees.

Table "Constraints" in [6] gives an overview over scope and usage of the different types of Timing Constraints described in the following chapters:

3.6.1 EventTriggeringConstraint

[TPS_TIMEX_00071] `EventTriggeringConstraint` specifies occurrence behavior respectively model

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008

⌈The element `EventTriggeringConstraint` is used to specify the particular occurrences of a given timing description event.⌋

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 3.31: EventTriggeringConstraint

3.6.1.1 PeriodicEventTriggering

[TPS_TIMEX_00076] `PeriodicEventTriggering` specifies periodic occurrences of events

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008

⌈The element `PeriodicEventTriggering` is used to specify the characteristics of a timing description event which occurs periodically.⌋

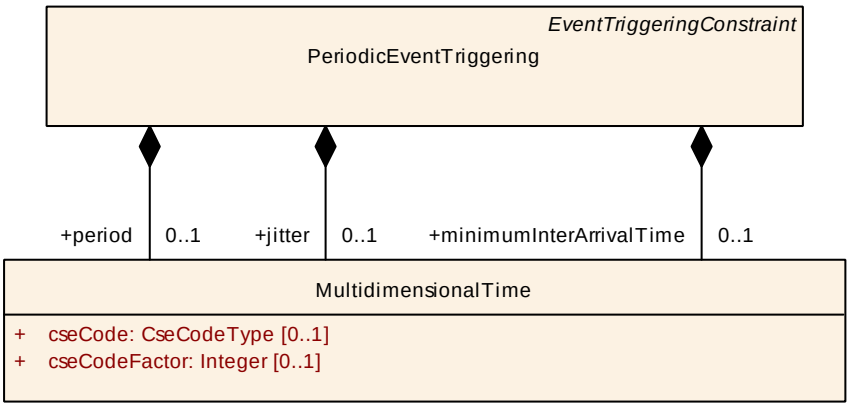


Figure 3.29: PeriodicEventTriggering

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 3.32: PeriodicEventTriggering

The Periodic Event Triggering is characterized by the following parameters:

- Period
- Jitter
- Minimum Inter-Arrival Time

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

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

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

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

[constr_4589] Maximum value of the parameter `minimumInterArrivalTime`

Status: DRAFT

⌈The value of the parameter `minimumInterArrivalTime` shall be less than or equal the value of the parameter `period`.⌋

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

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

Figure 3.30 illustrates the parameters of the `PeriodicEventTriggering`. The upper part of this figure shows the case that the value of `jitter` is less than the value of the parameter `period`; whereas the lower part of this figure shows the case that the value of `jitter` is greater than or equal the value of the parameter `period`.

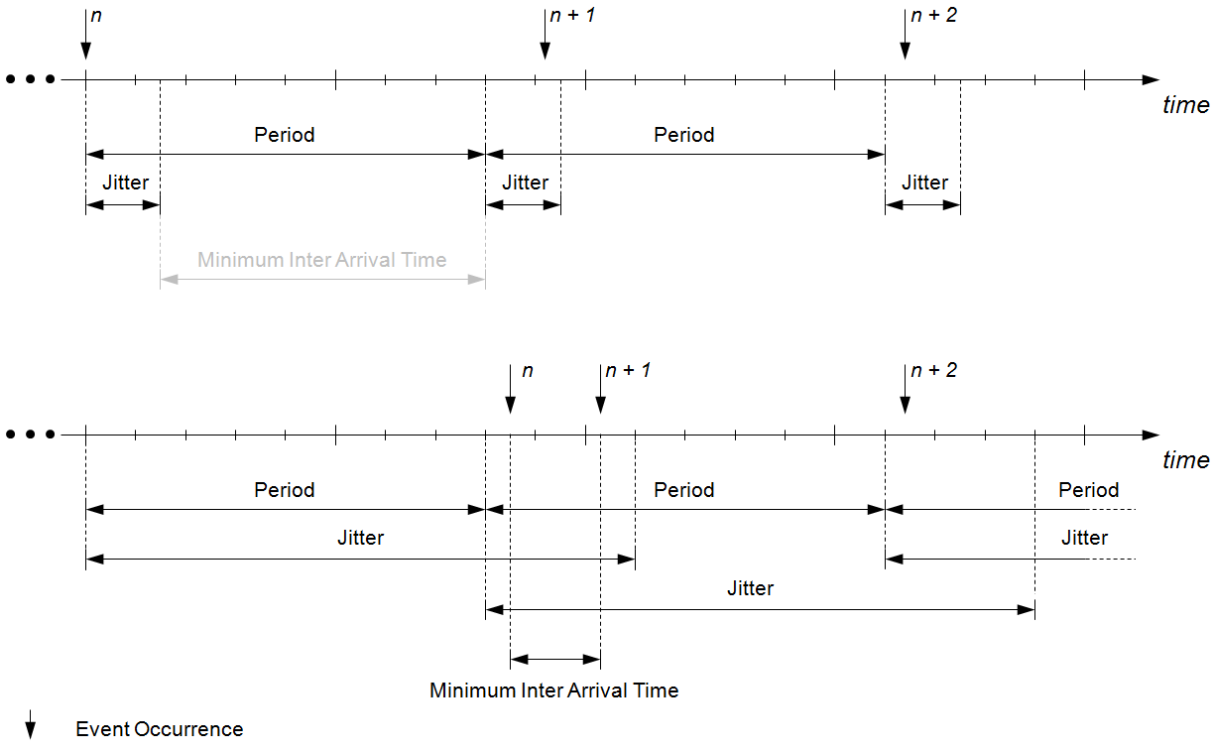


Figure 3.30: Parameters characterizing the Periodic Event Triggering

3.6.1.1.1 Examples

A Periodic Event Triggering Constraint is specified with the following parameters: `period` is six milliseconds (6ms) and `jitter` is two milliseconds (2ms). In other words, one imposes a timing constraint on an event to occur every six milliseconds and specifies that a deviation of two milliseconds is tolerable. In addition, it is assumed that the `minimumInterArrivalTime` is one millisecond (1ms) and therefore has no impact on the timing of the event's occurrences. This timing constraint is shown in Figure 3.31. The repeating gray-colored rectangles in this figure indicate the time intervals during which the event may occur; in other words it marks the subsequent time intervals the event is expected to occur.

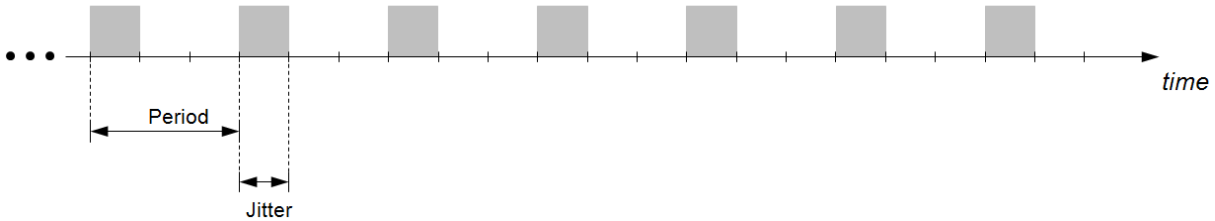


Figure 3.31: Example of a Periodic Event Triggering Constraint

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

The subsequent event occurrences shown in Figure 3.32 satisfy the given periodic event triggering constraint, because all occurrences of the event observed during the observation time interval happen in their corresponding time interval given by `period` and `jitter`.

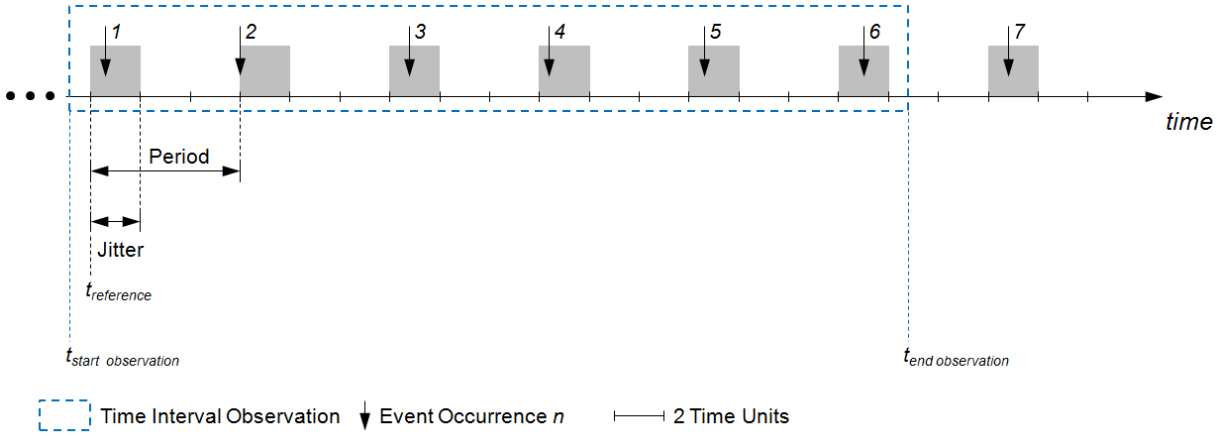


Figure 3.32: Event occurrences satisfying the given Period Event Triggering Constraint shown in the example at the beginning of this subsection.

The subsequent event occurrences shown in Figure 3.33 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 3.32 the reference point-in-time is another one.

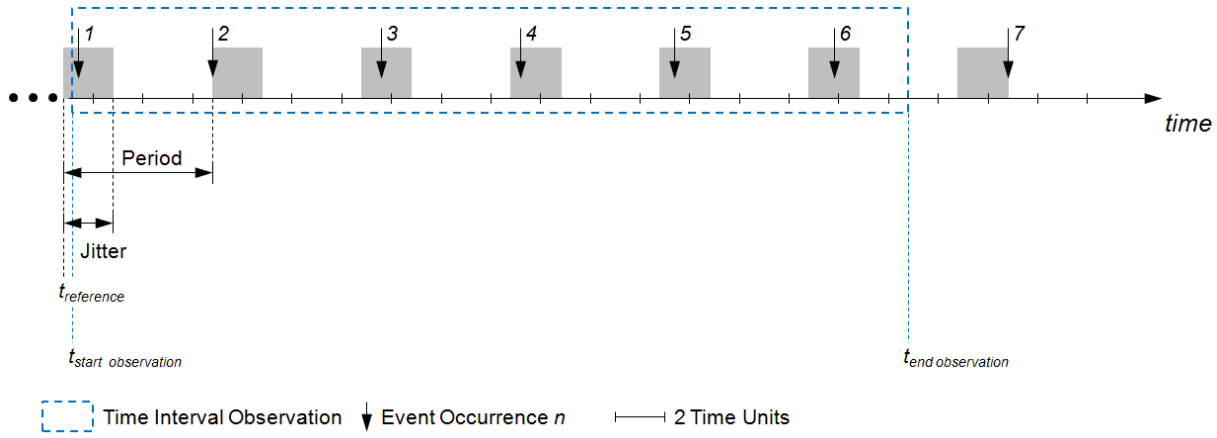


Figure 3.33: Event occurrences satisfying the given Period Event Triggering Constraint shown in the example at the beginning of this subsection, but with another reference point-in-time $t_{reference}$.

The subsequent event occurrences shown in Figure 3.34 violate the given periodic event triggering constraint, because the fifth occurrence of the event does not happen in its corresponding time interval given by `period` and `jitter`. In other words, there does not exist a reference point-in-time that ensures that all occurrences of the event observed during the observation time interval happen in their corresponding time interval given by `period` and `jitter`. And this results in a violation of the parameters `period` and `jitter`.

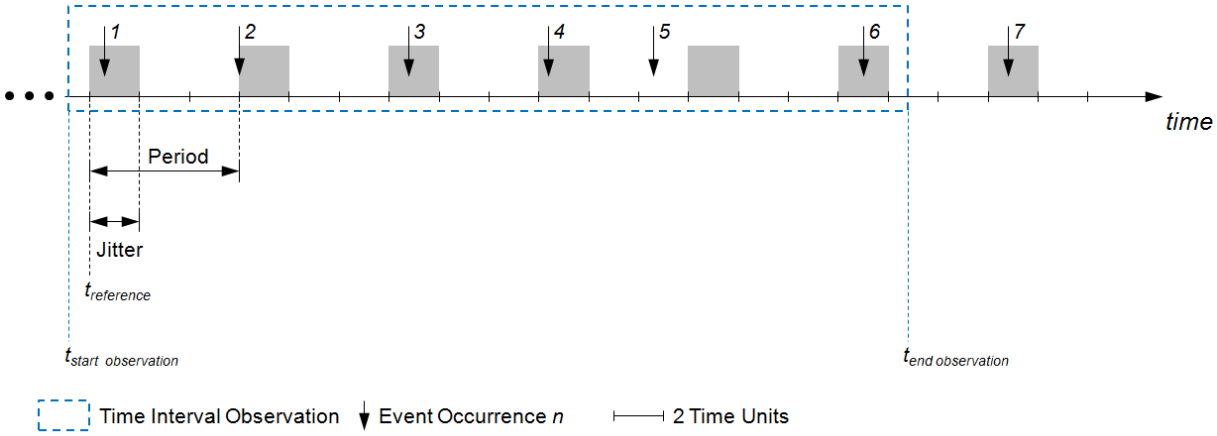


Figure 3.34: Event occurrences violating the given Period Event Triggering Constraint shown in the example at the beginning of this subsection.

The subsequent event occurrences shown in Figure 3.35 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`.

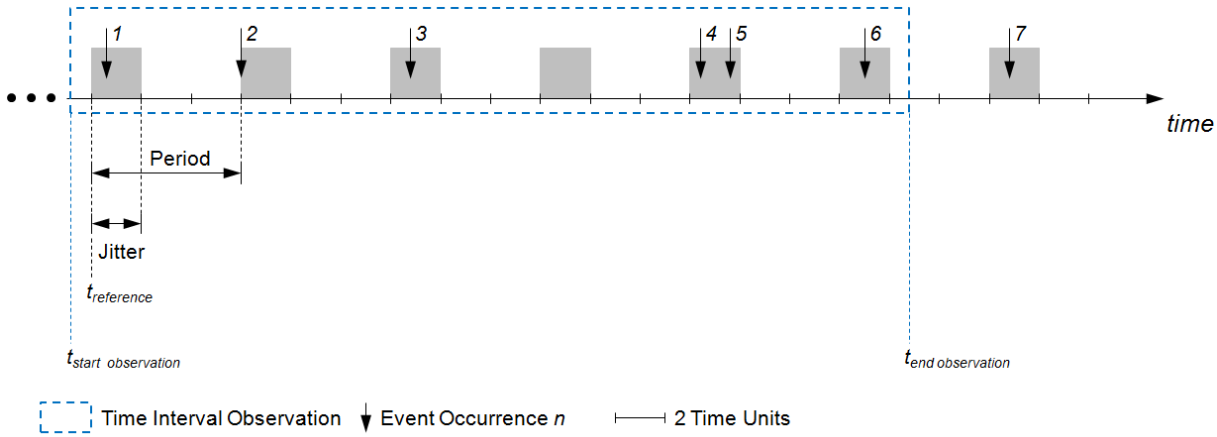


Figure 3.35: Event occurrences satisfying the given Period Event Triggering Constraint shown in the example at the beginning of this subsection.

3.6.1.2 SporadicEventTriggering

[TPS_TIMEX_00077] `SporadicEventTriggering` **specifies sporadic occurrences of events**

Status: DRAFT

Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008

⌈The element `SporadicEventTriggering` is used to specify the characteristics of a timing description event which occurs sporadically.⌋

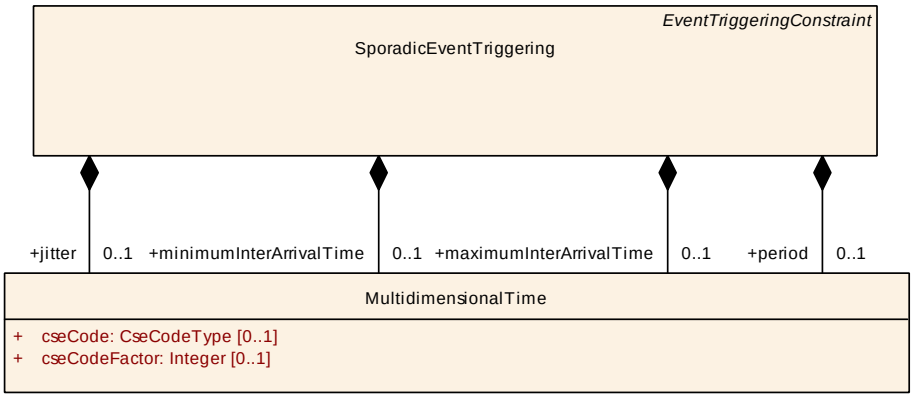


Figure 3.36: SporadicEventTriggering

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 3.33: SporadicEventTriggering

This is a generalization of the periodic event triggering described in subsection 3.6.1.1. The difference is that the event can, but not necessarily shall occur. For this rea-

son, there is one additional parameter required for the specification of the SporadicEventTriggering, namely the maximumInterArrivalTime, which specifies the largest possible time distance between two event occurrences.

The Sporadic Event Triggering is characterized by the following parameters:

- Minimum Inter-Arrival Time
- Maximum Inter-Arrival Time
- Period
- Jitter

The first two parameters are required ones and the last two parameters are optional. These parameters are described in the following and Figure 3.37 illustrates the parameters of the SporadicEventTriggering.

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

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

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

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

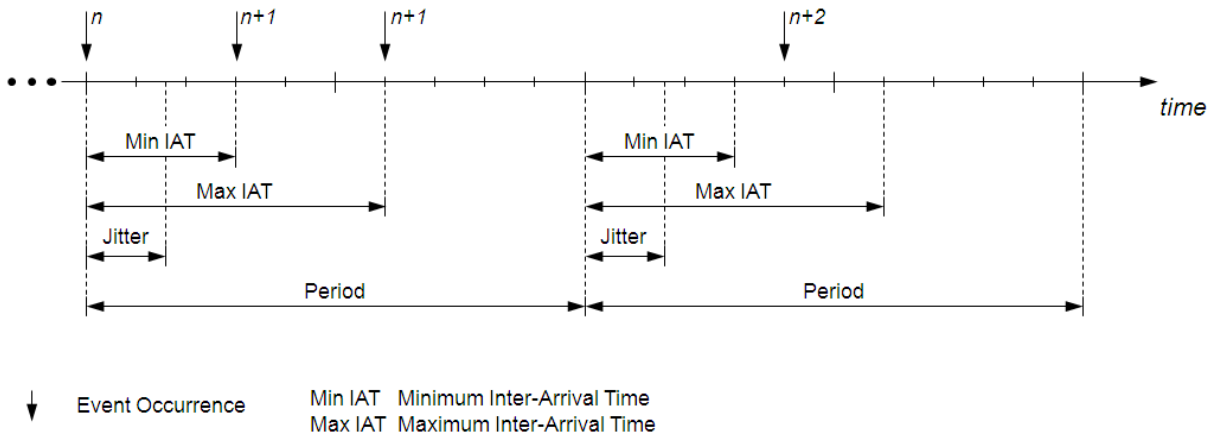


Figure 3.37: Parameters characterizing the Sporadic Event Triggering

3.6.1.3 ConcretePatternEventTriggering

[TPS_TIMEX_00078] `ConcretePatternEventTriggering` specifies concrete pattern of occurrences of events

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008

⌈The element `ConcretePatternEventTriggering` is used to specify the characteristics of a timing description event which occurs as a concrete pattern.⌋

This describes events which occur following a known pattern.

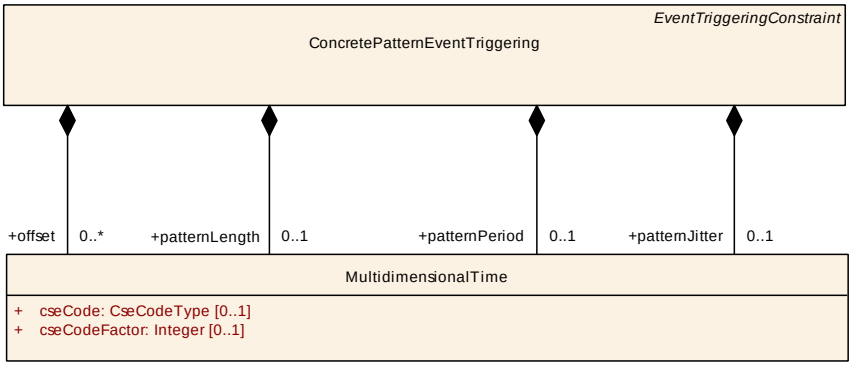


Figure 3.38: ConcretePatternEventTriggering

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***
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`.

▽

△

Class	**ConcretePatternEventTriggering**			
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 3.34: ConcretePatternEventTriggering

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

- Pattern Length
- Offset
- Pattern Period
- Pattern Jitter

The first two parameters are required ones, whereas the two last parameters are optional. The parameters are described in the following and are illustrated in Figure 3.39 and Figure 3.40.

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

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

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

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

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

[constr_4585] Specifying `patternLength`

Status: DRAFT

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

[constr_4590] Specifying `patternLength`, `patternJitter` and `patternPeriod`

Status: DRAFT

⌈The pattern length, pattern jitter and pattern period shall be specified such that the following holds: `patternLength` + `patternJitter` < `patternPeriod`.⌋

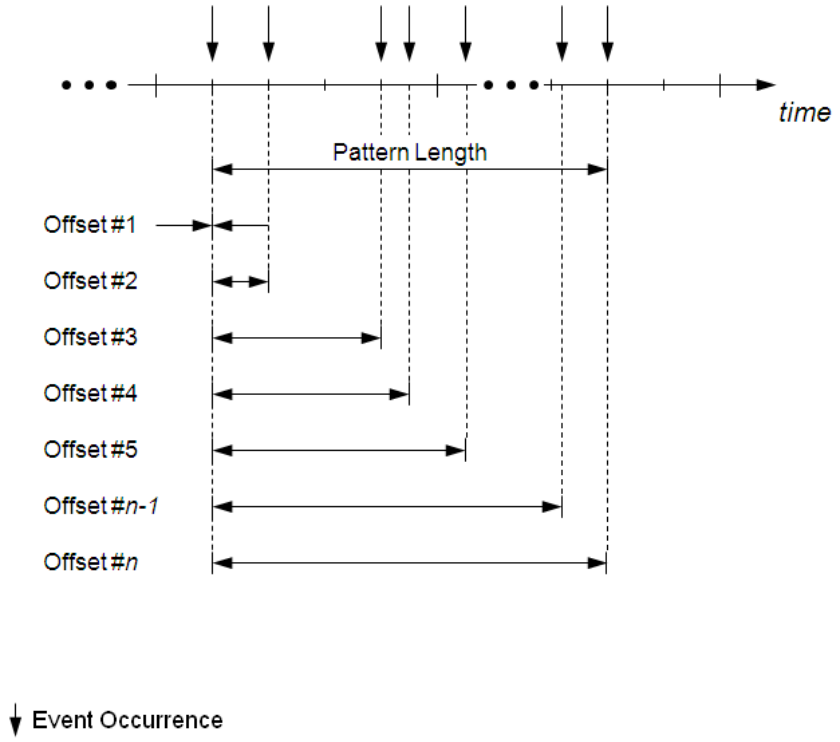


Figure 3.39: Parameters characterizing the Concrete Pattern Event Triggering

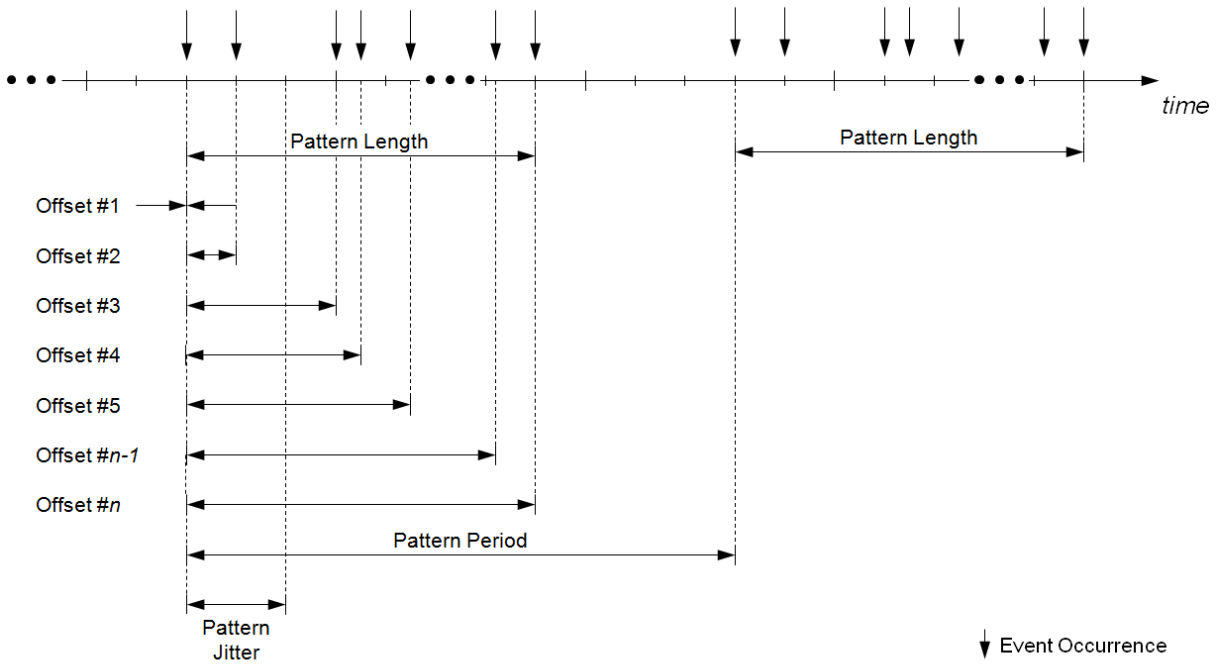


Figure 3.40: Parameters characterizing the Concrete Pattern Event Triggering when periodically being repeated

3.6.1.4 BurstPatternEventTriggering

[TPS_TIMEX_00079] `BurstPatternEventTriggering` specifies burst of occurrences of events

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008

⌈The element `BurstPatternEventTriggering` is used to specify the characteristics of a timing description event which occurs as a burst.⌋

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

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

- Pattern Jitter

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

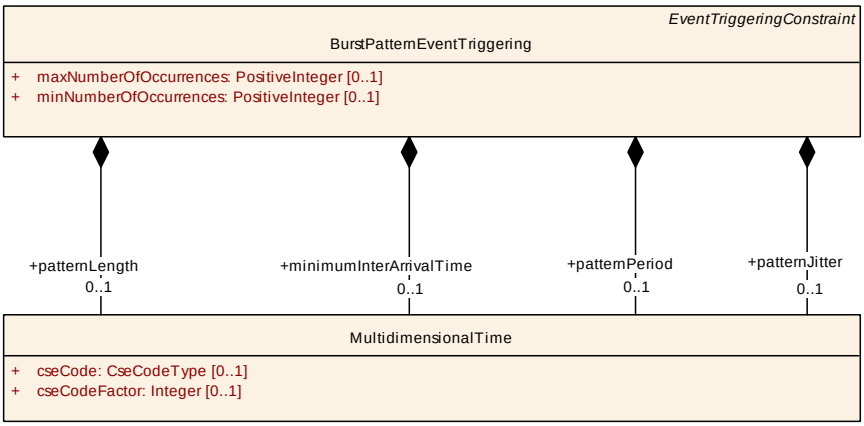


Figure 3.41: BurstPatternEventTriggering

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*, *Timing Constraint*, *Traceable*			
Aggregated by	*TimingExtension*.timingGuarantee, *TimingExtension*.timingRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maxNumberOf Occurrences	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`.
minimumInter ArrivalTime	MultidimensionalTime	0..1	aggr	Specifies the minimum distance between subsequent occurrences of the event within the given time interval.
minNumberOf Occurrences	PositiveInteger	0..1	attr	The minimum number of event occurrences within the given time interval. **Tags:** xml.sequenceOffset=10
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 3.35: BurstPatternEventTriggering

The parameters are described in the following and are illustrated in Figure 3.42 and Figure 3.43.

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

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

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

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

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

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

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

[constr_4574] Specifying minimum and maximum number of occurrences

Status: DRAFT

⌈The minimum and maximum number of occurrences shall be specified such that the following holds: $0 \leq$ minNumberOfOccurrences $\leq$ maxNumberOfOccurrences.⌋

[constr_4575] Specifying minimum inter-arrival time and pattern length

Status: DRAFT

⌈The minimum inter-arrival time and pattern length shall be specified such that the following holds: $0 <$ minimumInterArrivalTime $\leq$ patternLength.⌋

[constr_4576] Specifying pattern length, pattern jitter and patter period

Status: DRAFT

⌈The pattern length, pattern jitter and pattern period shall be specified such that the following holds: `patternLength` + `patternJitter` < `patternPeriod`.⌋

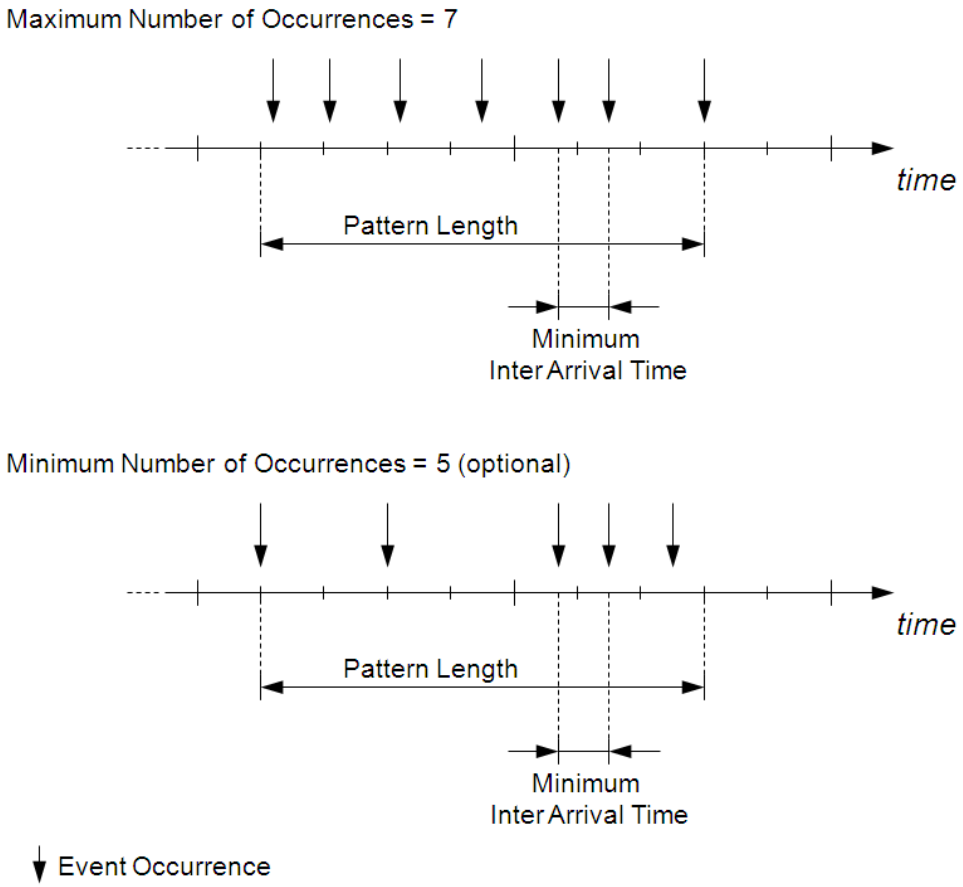


Figure 3.42: Parameters characterizing the Burst Pattern Event Triggering

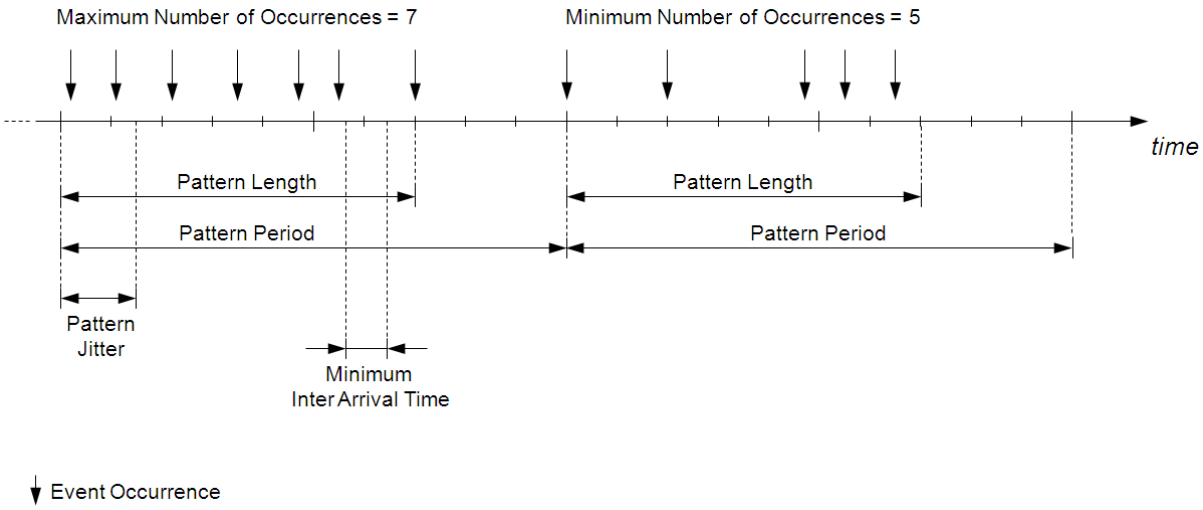


Figure 3.43: Parameters characterizing the Burst Pattern Event Triggering when periodically being repeated

3.6.1.5 ArbitraryEventTriggering

[TPS_TIMEX_00080] `ArbitraryEventTriggering` specifies arbitrary occurrences of an event

Status: DRAFT

Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00006, RS_TIMEX_00008

⌈The element `ArbitraryEventTriggering` is used to specify the characteristics of a timing description event which occurs arbitrarily.⌋

This describes the occasional occurrence of a timing event.

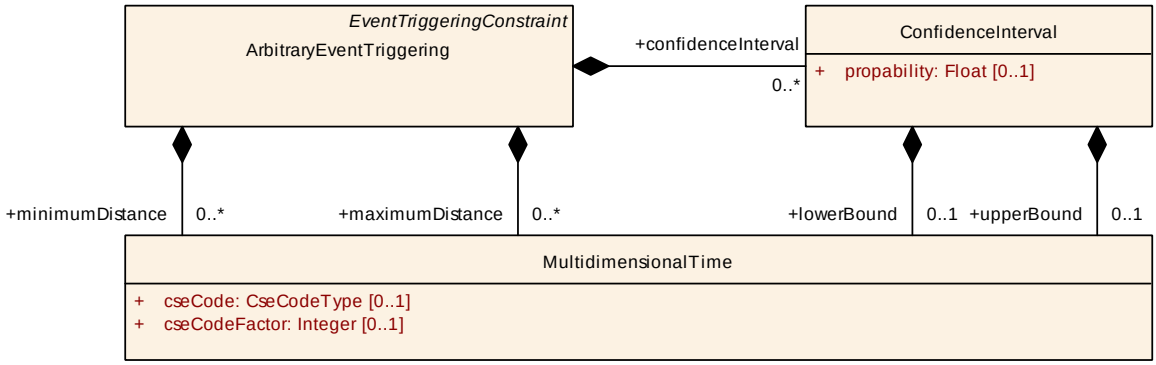


Figure 3.44: ArbitraryEventTriggering

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
maximumDistance	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
minimumDistance	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 3.36: 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 3.37: ConfidenceInterval

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

The Arbitrary Event Triggering is characterized by the following parameters:

- Minimum Distance

- Maximum Distance

These parameters are required ones and are described in the following. Figure 3.45 illustrates the parameters of the `ArbitraryEventTriggering`.

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

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

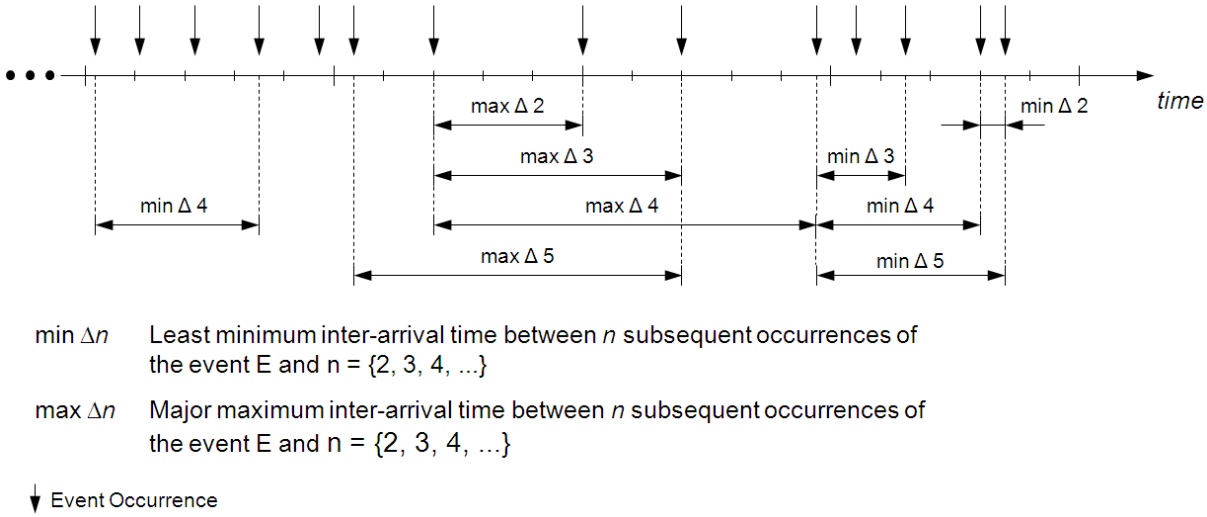


Figure 3.45: Parameters characterizing the Arbitrary Event Triggering

3.6.2 LatencyTimingConstraint

[TPS_TIMEX_00072] `LatencyTimingConstraint` specifies latency constraints

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00012

⌈The element `LatencyTimingConstraint`[1] is used to specify the amount of time that elapses between the occurrence of any two timing description events.⌋

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

In the timing specification a `LatencyTimingConstraint` is associated with one `TimingDescriptionEventChain`, and specifies the minimum and/or maximum time duration between the occurrence of the stimulus and the occurrence of the corresponding response of that chain. However, in multi-rate networks, data can get lost

[1] A synonym for delay

or get duplicated because of potential different producer and consumer periods. Data loss occurs, if the consumer's period is greater than the producer's period (undersampling). Accordingly, data duplication occurs, if the consumer's period is smaller than the producer's period (oversampling). This is depicted in figure 3.46.

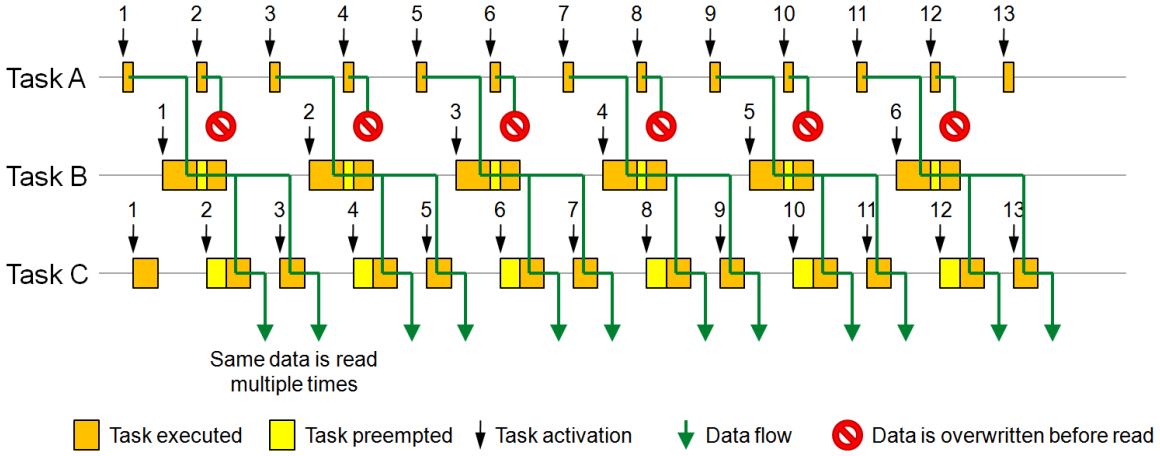


Figure 3.46: Loss and duplication of data due to under- and oversampling.

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

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

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

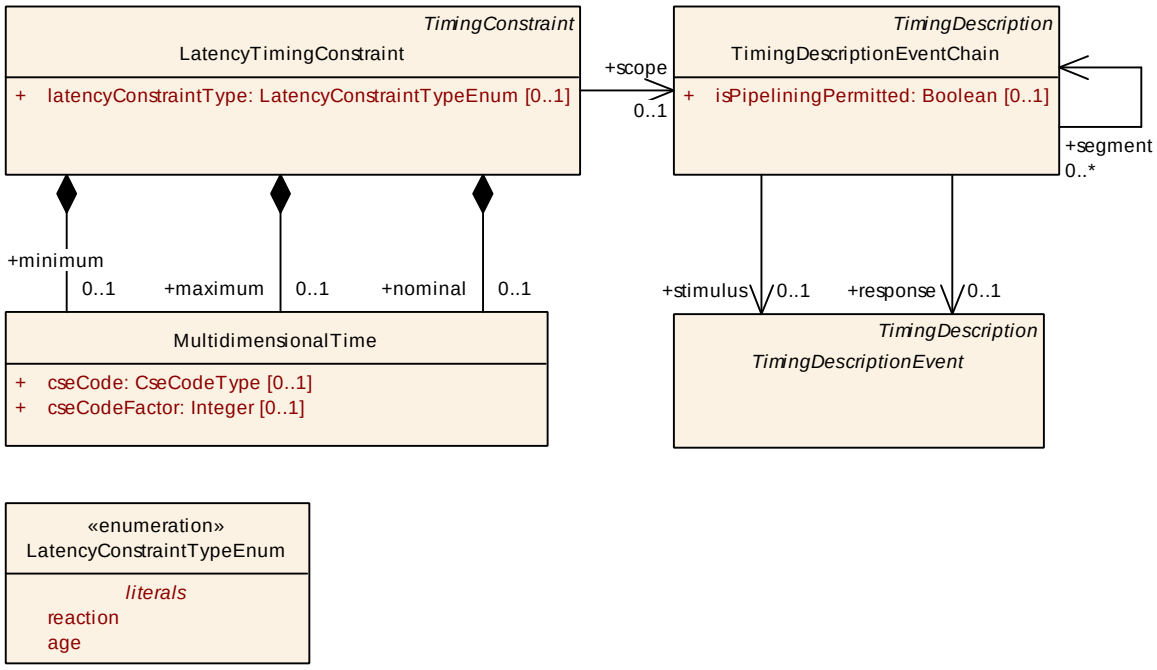


Figure 3.47: Latency constraint

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 3.38: LatencyTimingConstraint

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 3.39: LatencyConstraintTypeEnum

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

The application of latency constraints leads to some interesting observations:

- In systems without over- and 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.

3.6.3 AgeConstraint

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

[TPS_TIMEX_00073] `AgeConstraint` to specify age constraints

Status: DRAFT
Upstream requirements: RS_TIMEX_00001

⌈The element `AgeConstraint` is used to specify a minimum and maximum age that is tolerated when a variable data prototypes is received.⌋

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

At a later stage during the development, when the refined software architecture exposes the relation between the actual sender of the data and the receiver, an event chain between the sending and receiving point in time shall be defined and associated with a `LatencyTimingConstraint` (see 3.6.2) in order to refine the previous defined age constraint.

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

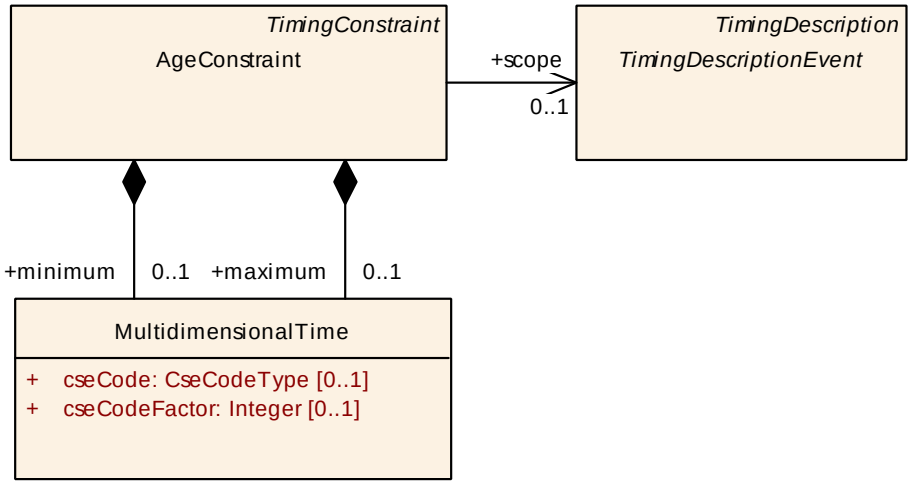


Figure 3.48: Age constraint

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

[constr_4573] Restricted usage of `AgeConstraint`

Status: DRAFT

⌈An `AgeConstraint` shall only be defined for events of type `TimingDescriptionEvent` associated with the receipt and reading of data.⌋

Class	**AgeConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::AgeConstraint			
Note	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 3.40: AgeConstraint

3.6.4 SynchronizationTimingConstraint

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

[TPS_TIMEX_00074] `SynchronizationTimingConstraint` specifies synchronicity constraints

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00007, RS_TIMEX_00008, RS_TIMEX_00017

⌈The element `SynchronizationTimingConstraint` is used to specify a synchronization constraint among the occurrences of two or more timing description events.⌋

A `SynchronizationTimingConstraint` is imposed either on events (3.6.4.2) or on event chains (3.6.4.1).

Class	**SynchronizationTimingConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::SynchronizationTiming Constraint			
Note	This constraint is used to restrict the timing behavior of different, but correlated events or event chains, with regard to synchronization. Two scenarios are supported: • If (`synchronizationConstraintType`==`responseSynchronization`) – `TimingDescriptionEvent`s: An arbitrary number of correlated events which play the role of responses shall occur synchronously with respect to a predefined tolerance. – `TimingDescriptionEventChain`s: 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`) – `TimingDescriptionEvent`s:An arbitrary number of correlated events which play the role of stimuli shall occur synchronously with respect to a predefined tolerance. – `TimingDescriptionEventChain`s: 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***

▽

△

Class	**SynchronizationTimingConstraint**			
event OccurrenceKind	EventOccurrenceKind Enum	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	TimingDescriptionEvent Chain	*	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])
synchronization ConstraintType	SynchronizationType Enum	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 3.41: SynchronizationTimingConstraint

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 3.42: EventOccurrenceKindEnum

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

▽

△

Enumeration	SynchronizationTypeEnum
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 3.43: SynchronizationTypeEnum

[constr_4588] `SynchronizationTimingConstraint` shall either reference events or event chains

Status: DRAFT

⌈The `SynchronizationTimingConstraint` shall either reference timing description events or timing description event chains, but not both at the same time.⌋

3.6.4.1 SynchronizationTimingConstraint on Event Chains

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

The `SynchronizationTimingConstraint` is characterized by the following parameters:

- Tolerance
- Event Occurrence Kind
- Synchronization Constraint Type

The parameters are described in the following and are illustrated in Figure 3.49 and Figure 3.50.

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

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

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

[constr_4580] `SynchronizationTimingConstraint` shall reference at least two event chains

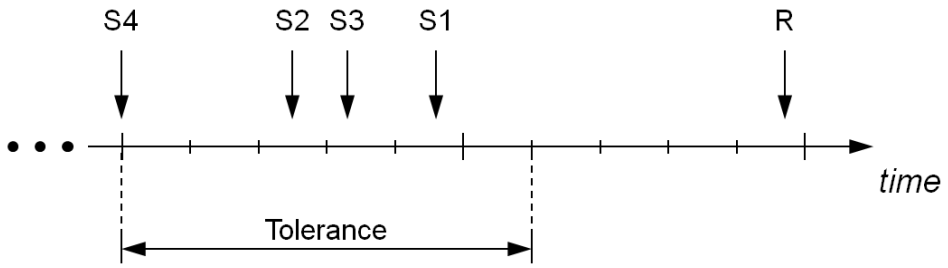
Status: DRAFT

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

[constr_4587] Specifying attribute `synchronizationConstraintType`

Status: DRAFT

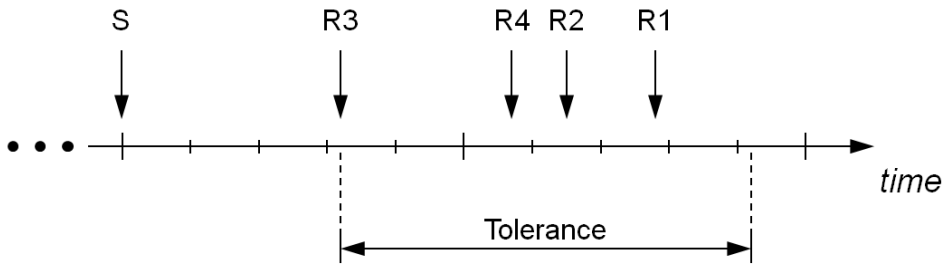
⌈The attribute `synchronizationConstraintType` shall be specified if the `SynchronizationTimingConstraint` is imposed on event chains.⌋



S# Stimulus Event
R Response Event

Event Occurrence

Figure 3.49: Parameters characterizing the Synchronization Timing Constraint imposed on the stimulus events of event chains.



S Stimulus Event
R# Response Event

Event Occurrence

Figure 3.50: Parameters characterizing the Synchronization Timing Constraint imposed on the response events of event chains.

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

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

3.6.4.2 SynchronizationTimingConstraint on Events

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

At a later stage during the development, when the refined software architecture exposes the complete event chains (e.g. because the common stimulus gets known), the respective event chains shall be specified and associated with a `SynchronizationTimingConstraint` on event chains (see 3.6.4.1) in order to refine the previously defined `SynchronizationTimingConstraint` on events.

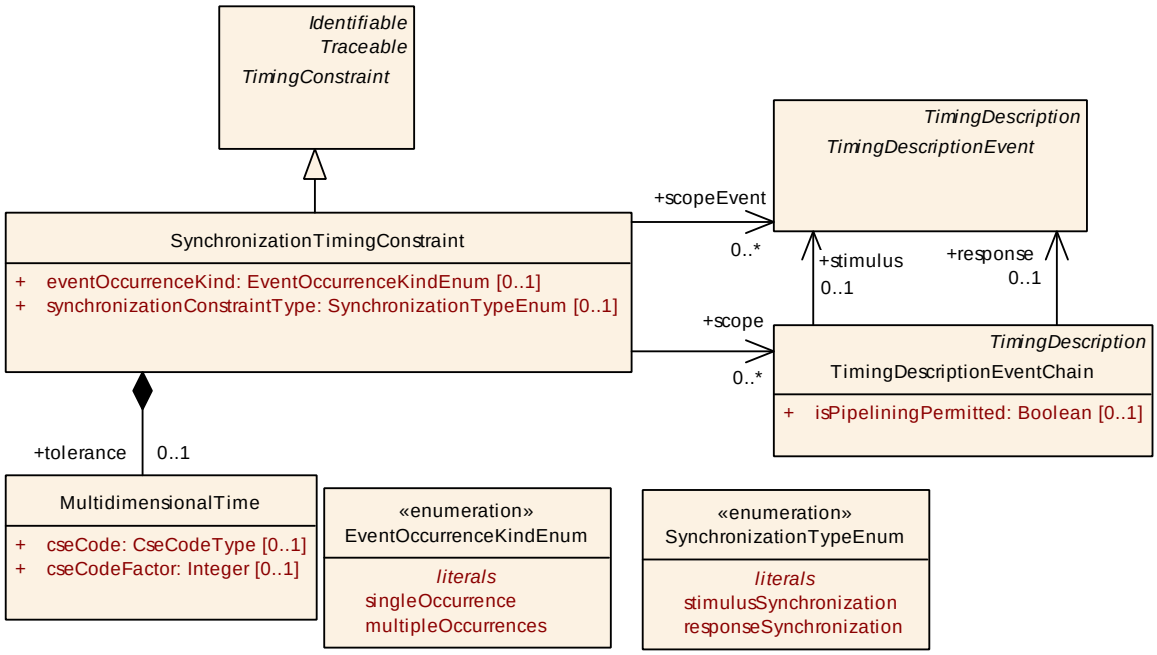


Figure 3.51: Synchronization Timing Constraint on Events

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

- Tolerance
- Event Occurrence Kind
- Synchronization Constraint Type

The parameters are described in the following and are illustrated in Figure 3.52.

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

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

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

[constr_4579] SynchronizationTimingConstraint shall reference at least two events

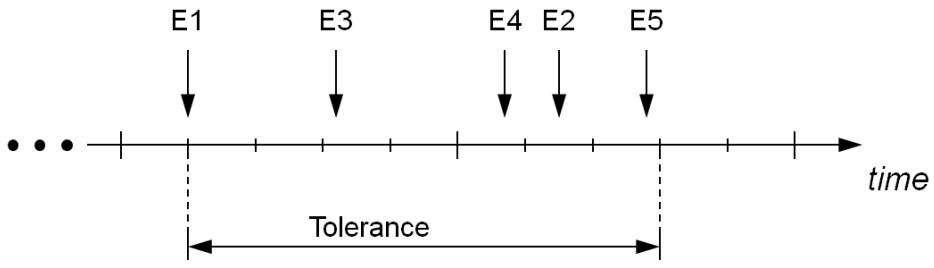
Status: DRAFT

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

[constr_4586] Specifying attribute synchronizationConstraintType

Status: DRAFT

⌈The attribute synchronizationConstraintType shall be specified if the SynchronizationTimingConstraint is imposed on events.⌋



E# Event #

Event Occurrence

Figure 3.52: Parameter characterizing the Synchronization Constraint

3.6.5 OffsetTimingConstraint

[TPS_TIMEX_00081] `OffsetTimingConstraint` specifies offset between occurrences of events

Status: DRAFT
Upstream requirements: RS_TIMEX_00001, RS_TIMEX_00002, RS_TIMEX_00008

⌈The element `OffsetTimingConstraint` is used to specify an offset between the occurrences of two timing description events.⌋

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.

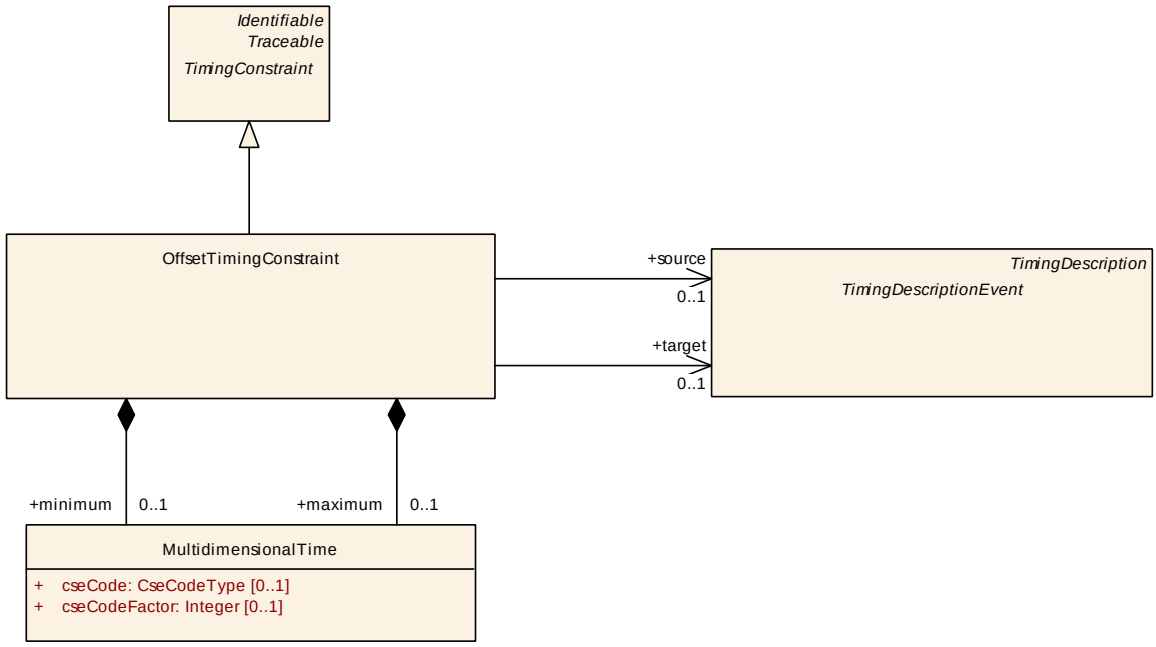


Figure 3.53: Offset Timing Constraint

Class	**OffsetTimingConstraint**			
Package	M2::AUTOSARTemplates::CommonStructure::Timing::TimingConstraint::OffsetConstraint			
Note	Bounds the time offset between the occurrence of two timing events, without requiring a direct functional dependency between the `source` and the `target`. If the `target` event occurs, it is expected to occur earliest with the `minimum`, and latest with the `maximum` offset relatively after the occurrence of the `source` event. Note: not every `source` event occurrence 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 3.44: OffsetTimingConstraint

3.6.6 Traceability of Constraints

[TPS_TIMEX_00089] `TimingConstraint` is a `Traceable`

Status: DRAFT
Upstream requirements: RS_TIMEX_00010

⌈The element `TimingConstraint` and all of its specializations, commonly called timing constraints, are traceable.⌋

The support for traceability [3] enables one to specify relationships between timing constraints and corresponding AUTOSAR elements that satisfy those timing requirements.

3.7 Logical Execution Time

`Logical Execution Time` (`LET`) is currently restricted to `CP`.

3.8 System Level Logical Execution Time

Please refer to [6] chapter "System Level Logical Execution Time".

3.9 Blueprinting

Please refer to [6] chapter "Blueprinting".

3.10 Methodology

The AUTOSAR methodology (see [7] for a general introduction) provides several well-defined process steps, and furthermore artifacts that are provided or needed by these steps.

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

[TPS_TIMEX_00075] Optional use of timing extensions

Status: DRAFT
Upstream requirements: RS_TIMEX_00003

⌈The elements `TimingExtension`, `TimingDescription`, and `TimingConstraint` of the timing extensions are derived from the element `ARElement`. This enables one to deliver timing extensions in a separate document. In addition, there are no external references from any template that point to timing extensions elements.⌋

A Reference Material

A.1 Terms and Abbreviations

The main list of terms and abbreviations are defined in [8] and [6].

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.1.

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

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.1.

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 Executable Timing Description is complete	This imposition time is aimed at the time when a Executable 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 Machine Timing Description is complete	This imposition time is aimed at the time when a Machine Timing is complete.
at the time when the Service Timing Description is complete	This imposition time is aimed at the time when a Service Timing is complete.

▽

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

△

at anyTime in the workflow	This means that the constraint is invariant of the imposition time and therefore universally applicable. Some model configurations *never* make sense and therefore need to be restricted as early as possible in order to avoid the situation where obviously non-sensical model content is unjustifiably tolerated until some step in the workflow. And then (considerable) effort has to be spent for cleaning up the model.

Table A.1: 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 [9] and denotes how each of them are satisfied by the meta-model.

Requirement	Description	Satisfied by
[RS_TIMEX_00001]	Timing properties	[TPS_TIMEX_00058] [TPS_TIMEX_00059] [TPS_TIMEX_00060] [TPS_TIMEX_00061] [TPS_TIMEX_00062] [TPS_TIMEX_00063] [TPS_TIMEX_00064] [TPS_TIMEX_00065] [TPS_TIMEX_00069] [TPS_TIMEX_00070] [TPS_TIMEX_00071] [TPS_TIMEX_00072] [TPS_TIMEX_00073] [TPS_TIMEX_00074] [TPS_TIMEX_00076] [TPS_TIMEX_00077] [TPS_TIMEX_00078] [TPS_TIMEX_00079] [TPS_TIMEX_00080] [TPS_TIMEX_00081] [TPS_TIMEX_00082] [TPS_TIMEX_00083] [TPS_TIMEX_00084] [TPS_TIMEX_00085] [TPS_TIMEX_00086] [TPS_TIMEX_00087] [TPS_TIMEX_00088] [TPS_TIMEX_00090] [TPS_TIMEX_00092] [TPS_TIMEX_00093]
[RS_TIMEX_00002]	Timing constraints	[TPS_TIMEX_00071] [TPS_TIMEX_00072] [TPS_TIMEX_00074] [TPS_TIMEX_00076] [TPS_TIMEX_00077] [TPS_TIMEX_00078] [TPS_TIMEX_00079] [TPS_TIMEX_00080] [TPS_TIMEX_00081]
[RS_TIMEX_00003]	Optionality of timing constraints	[TPS_TIMEX_00075]
[RS_TIMEX_00004]	Event chains	[TPS_TIMEX_00070]
[RS_TIMEX_00005]	Structure of event chains	[TPS_TIMEX_00070]
[RS_TIMEX_00006]	Triggering behavior of event chains	[TPS_TIMEX_00071] [TPS_TIMEX_00076] [TPS_TIMEX_00077] [TPS_TIMEX_00078] [TPS_TIMEX_00079] [TPS_TIMEX_00080]
[RS_TIMEX_00007]	Synchronization of event chains	[TPS_TIMEX_00074]
[RS_TIMEX_00008]	Multiple asynchronous time bases	[TPS_TIMEX_00071] [TPS_TIMEX_00074] [TPS_TIMEX_00076] [TPS_TIMEX_00077] [TPS_TIMEX_00078] [TPS_TIMEX_00079] [TPS_TIMEX_00080] [TPS_TIMEX_00081]
[RS_TIMEX_00010]	Validity of timing properties and constraints	[TPS_TIMEX_00089]
[RS_TIMEX_00012]	Sensor/actuator delay	[TPS_TIMEX_00072]
[RS_TIMEX_00017]	Synchronization constraint on events	[TPS_TIMEX_00074]
[RS_TIMEX_00019]	AUTOSAR Methodology support	[TPS_TIMEX_00092] [TPS_TIMEX_00093]

▽

△

Requirement	Description	Satisfied by
[RS_TIMEX_00024]	Support for Service Oriented Communication	[TPS_TIMEX_00058] [TPS_TIMEX_00059] [TPS_TIMEX_00060] [TPS_TIMEX_00061] [TPS_TIMEX_00062] [TPS_TIMEX_00063] [TPS_TIMEX_00064] [TPS_TIMEX_00065]

Table A.2: Requirements Tracing

B Mentioned Class Tables

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

Class	***ARElement*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, Allocator, ApApplicationError, ApApplicationErrorDomain, ApApplicationErrorSet, ApplicabilityInfoSet, *AutosarDataType*, *BaseType*, BlueprintMappingSet, BuildActionManifest, CalibrationParameterValueSet, CanXlProps, ClientIdDefinitionSet, Collection, *CompositionPortToExecutablePortMapping*, CompuMethod, ConsistencyNeedsBlueprintSet, ConstantSpecification, ConstantSpecificationMappingSet, CryptoEllipticCurveProps, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, CryptoServiceQueue, CryptoSignatureScheme, DataConstr, DataExchangePoint, DataTransformationSet, DataTypeMappingSet, DdsCpConfig, *DiagnosticCommonElement*, DiagnosticConnection, DiagnosticContributionSet, DltContext, DltEcu, Documentation, E2EProfileCompatibilityProps, E2EProfileConfigurationSet, EndToEndProtectionSet, EthIpProps, EthTcpIpIcmpProps, EthTcpIpProps, EvaluatedVariantSet, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FirewallRule, GeneralPurposeConnection, *GrantDesign*, HwCategory, HwElement, HwType, *IEEE1722TpConnection*, IPSecConfigProps, *IdsCommonElement*, IdsDesign, ImpositionTimeDefinitionGroup, InterfaceMapping, InterpolationRoutineMappingSet, KeywordSet, LTMessageCollectionToPortPrototypeMapping, LifeCycleInfoSet, LifeCycleStateDefinitionGroup, LogAndTraceMessageCollectionSet, MacSecGlobalKayProps, MacSecParticipantSet, McFunction, McGroup, ModeDeclarationGroup, ModeDeclarationMappingSet, PhysicalDimension, PhysicalDimensionMappingSet, *PlatformModuleEndpointConfiguration*, *PortInterface*, PortInterfaceMappingSet, PortInterfaceToDataTypeMapping, PortPrototypeBlueprint, PostBuildVariantCriterion, PostBuildVariantCriterionValueSet, PredefinedVariant, ProcessDesign, ProcessDesignToMachineDesignMapping, RapidPrototypingScenario, SdgDef, *SecureComProps*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping*, *ServiceInterfaceElementMapping*, ServiceInterfacePedigree, SignalServiceTranslationPropsSet, SoftwareClusterDesign, SomeipDataPrototypeTransformationProps, SomeipRemoteUnicastConfig, SomeipSdClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroupTimingConfig, SomeipSdServerServiceInstanceConfig, SwAddrMethod, SwAxisType, *SwComponentType*, SwRecordLayout, SwSystemconst, SwSystemconstantValueSet, System, SystemSignal, SystemSignalGroup, *TimingExtension*, TlsConnectionGroup, TlvDataIdDefinitionSet, TransformationPropsSet, TransformationPropsToServiceInterfaceElementMapping, Unit, UnitGroup, *UploadablePackageElement*, ViewMapSet			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.1: ARElement

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 B.2: AUTOSAR

Class	**AdaptiveApplicationSwComponentType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents the ability to support the formal modeling of application software on the AUTOSAR adaptive platform. Consequently, it shall only be used on the AUTOSAR adaptive platform. **Tags:** atp.recommendedPackage=AdaptiveApplicationSwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
internalBehavior	AdaptiveSwcInternalBehavior	0..1	aggr	This aggregation represents the internal behavior of the AdaptiveApplicationSwComponentType for the AUTOSAR adaptive platform. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalBehavior.shortName, internalBehavior.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.3: AdaptiveApplicationSwComponentType

Class	***AdaptivePlatformServiceInstance*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	This meta-class represents the ability to describe the existence and configuration of a service instance in an abstract way.
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*

▽

△

Class	***AdaptivePlatformServiceInstance*** (abstract)			
Subclasses	*ProvidedApServiceInstance*, *RequiredApServiceInstance*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
e2eEvent ProtectionProps	End2EndEvent ProtectionProps	*	aggr	This aggregation allows to protect an event or a field notifier that is defined inside of the ServiceInterface that is referenced by the ServiceInstance in the role service Interface.
e2eMethod ProtectionProps	End2EndMethod ProtectionProps	*	aggr	This aggregation allows to protect a method or a field getter or a field setter that is defined inside of the Service Interface that is referenced by the ServiceInstance in the role serviceInterface
secureCom Config	ServiceInterface ElementSecureCom Config	*	aggr	Configuration settings to secure the communication of ServiceInterface elements.
serviceInterface Deployment	ServiceInterface Deployment	0..1	ref	Reference to a ServiceInterfaceDeployment that identifies the ServiceInterface that is represented by the Service Instance.

Table B.4: AdaptivePlatformServiceInstance

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

Table B.5: Boolean

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
fireAndForget	Boolean	0..1	attr	This attribute defines whether this method is a fire&forget method (true) or not (false).
possibleApError	ApApplicationError	*	ref	This reference identifies AdaptivePlatformApplicationErrors as a possible error raised by the enclosing ClientServerOperation.

▽

△

Class	**ClientServerOperation**			
possibleApError Set	ApApplicationErrorSet	*	ref	This reference represents the ability to refer to an entire group of ApApplicationErrors as one model element instead of having to refer to all the represented Ap ApplicationErrors separately.

Table B.6: ClientServerOperation

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	SwComponent Prototype	*	aggr	The instantiated components that are part of this composition. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=component.shortName, component.variation Point.shortLabel vh.latestBindingTime=postBuild
connector	SwConnector	*	aggr	`SwConnector`s have the principal ability to establish a connection among `PortPrototype`s. They can have many roles in the context of a `CompositionSwComponentType`. Details are refined by subclasses. The aggregation of `SwConnector`s 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 `AssemblySwConnector`s between `ApplicationSwComponentType`s and `ServiceSwComponentType`s during the ECU integration. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=connector.shortName, connector.variation Point.shortLabel vh.latestBindingTime=postBuild
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the `ConstantSpecificationMapping` to be applied for initValues of `PPortComSpec`s and `RPortComSpec`. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the `DataTypeMapping` to be applied for the used `ApplicationDataType`s in `ServiceInterface`s. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dataTypeMapping

▽

△

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 B.7: CompositionSwComponentType

Class	**Executable**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents an executable program. **Tags:** atp.recommendedPackage=Executables			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
buildType	BuildTypeEnum	0..1	attr	This attribute describes the buildType of a module and/or platform implementation.
implementation Props	Executable ImplementationProps	*	aggr	This aggregation contains the collection of implementation-specific properties necessary to properly build the enclosing Executable.
minimumTimer Granularity	TimeValue	0..1	attr	This attribute describes the minimum timer resolution (TimeValue of one tick) that is required by the Executable.
reporting Behavior	ExecutionState ReportingBehavior Enum	0..1	attr	this attribute controls the execution state reporting behavior of the enclosing Executable.
rootSw Component Prototype	RootSwComponent Prototype	0..1	aggr	This represents the root SwCompositionPrototype of the Executable. This aggregation is required (in contrast to a direct reference of a SwComponentType) in order to support the definition of instanceRefs in Executable context.
traceSwitch Configuration	TraceSwitch Configuration	*	aggr	Configuration of the MsgId based trace switch **Tags:** atp.Status=draft
version	StrongRevisionLabel String	0..1	attr	Version of the executable.

Table B.8: Executable

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 B.9: 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, *FMFormulaByFeaturesAndAttributes*, *SwSystemconstDependentFormula*, TDEventOccurrenceExpressionFormula, TimingConditionFormula			
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 B.10: FormulaExpression

Class	***Identifiable*** (abstract)
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractFunctionalClusterDesign*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstance Filter*, *AbstractServiceInstance*, *AbstractSignalBasedToISignalTriggeringMapping*, AdaptiveSwcInternal Behavior, ApApplicationEndpoint, *ApmcAbstractDefinition*, *ApmcConfigurationElementDef*, *Apmc ContainerElementValue*, ApmcContainerValue, ApmcEnumerationLiteralDef, ApplicationEndpoint, ApplicationError, AppliedStandard, ArtifactChecksum, ArtifactLocator, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BuildAction Entity*, BuildActionEnvironment, Chapter, CheckpointTransition, ClassContentConditional, ClientId Definition, ClientServerOperation, Code, *CollectableElement*, ComManagementMapping, *Comm ConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, *CouplingPortAbstractShaper*, *CouplingPortStructuralElement*, CryptoCertificate, CryptoKeySlot, CryptoKeySlotDesign, CryptoKeySlotUsageDesign, CryptoProvider, *CryptoServiceMapping*, DataPrototypeGroup, DataPrototypeTransformationPropsIdent, Data Transformation, DdsCpDomain, DdsCpPartition, DdsCpQosProfile, DdsCpTopic, DdsDomainRange, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, Diagnostic FunctionInhibitSource, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DiagnosticSovd MethodPrimitive, DltApplication, DltArgument, DltMessage, DoIpInterface, DoIpLogicAddress, DoIp LogicalAddress, DoIpNetworkConfigurationDesign, DoIpRoutingActivation, E2EProfileConfiguration, End2EndEventProtectionProps, End2EndMethodProtectionProps, EndToEndProtection, Ethernet WakeupSleepOnDatalineConfig, EventHandler, EventMapping, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMap Element, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FieldMapping, FireAndForget MethodMapping, FlexrayArTpNode, FlexrayTpPduPool, *FrameTriggering*, GeneralParameter, Global Supervision, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HealthChannel*, *HeapUsage*, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, *IEEE1722TpAcfBus*, *IEEE1722TpAcfBus Part*, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignalTriggering, *IdentCaption*, ImpositionTime, InternalTriggeringPoint, Keyword, LifeCycleState, Linker, MacAddressVlanMembership, MacMulticastGroup, MacSecKayParticipant, McDataInstance, MemorySection, MemoryUsage, Method Mapping, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, NetworkEndpoint, *NmCluster*, *NmNode*, *PackageableElement*, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, PduTriggering, PerInstanceMemory, *PersistencyDeploymentElement*, *PersistencyInterfaceElement*, *Phm Supervision*, *PhysicalChannel*, PortGroup, *PortInterfaceMapping*, ProcessToMachineMapping, Processor, ProcessorCore, PskIdentityToKeySlotMapping, ResourceConsumption, ResourceGroup, RootSwClusterDesignComponentPrototype, RootSwComponentPrototype, RootSwComposition ▽ ▽

△

Class	***Identifiable*** (abstract)			
	△ Prototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecution Context, RptProfile, RptServicePoint, RunnableEntityGroup, *SdgAttribute*, SdgClass, SecOcJobMapping, SecOcJobRequirement, SecureCommunicationAuthenticationProps, *SecureCommunicationDeployment*, SecureCommunicationFreshnessProps, SecurityEventContextDataElement, SecurityEventContextProps, *ServiceEventDeployment*, *ServiceFieldDeployment*, ServiceInterfaceElementSecureComConfig, *Service MethodDeployment*, *ServiceNeeds*, SignalServiceTranslationEventProps, SignalServiceTranslation Props, SocketAddress, SoftwarePackageStep, SomeipEventGroup, SomeipProvidedEventGroup, SomeipTpChannel, *SpecElementReference*, *StackUsage*, *StateManagementActionItem*, State ManagementActionList, StateManagementStateNotification, *StateManagementStateRequest*, Static SocketConnection, StructuredReq, SupervisionCheckpoint, SupervisionMode, SupervisionMode Condition, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SwitchAsynchronous TrafficShaperGroupEntry, SystemMapping, *TimeBaseResource*, *TimingClock*, TimingClockSync Accuracy, TimingCondition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, Timing ModeInstance, TlsCryptoCipherSuite, TlsCryptoCipherSuiteProps, TlsJobMapping, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, TransformationISignalPropsIdent, *TransformationProps*, TransformationTechnology, Trigger, UcmDescription, UcmRetryStrategy, UcmStep, VariableAccess, VariationPointProxy, VehicleRolloutStep, 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
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

▽

△

Class	***Identifiable*** (abstract)			
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:** xml.attribute=true

Table B.11: Identifiable

Class	**Machine**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::MachineManifest			
Note	Machine that represents an Adaptive Autosar Software Stack. **Tags:** atp.recommendedPackage=Machines			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
default Application Timeout	EnterExitTimeout	0..1	aggr	This aggregation defines a default timeout in the context of a given Machine with respect to the launching and termination of applications.
environment Variable	TagWithOptionalValue	*	aggr	This aggregation represents the collection of environment variables that shall be added to the environment defined on the level of the enclosing Machine. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=environmentVariable
machineDesign	MachineDesign	0..1	ref	Reference to the MachineDesign this Machine is implementing.
module Instantiation	AdaptiveModule Instantiation	*	aggr	Configuration of Adaptive Autosar module instances that are running on the machine. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=moduleInstantiation.shortName
processor	Processor	*	aggr	This represents the collection of processors owned by the enclosing machine.
secure Communication Deployment	SecureCommunication Deployment	*	aggr	Target-configuration of secure communication protocol configuration settings to crypto module entities. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=secureCommunication Deployment.shortName

▽

△

Class	Machine			
trustedPlatform Executable LaunchBehavior	TrustedPlatform ExecutableLaunch BehaviorEnum	0..1	attr	This attribute controls the behavior of how authentication affects the ability to launch for each Executable.

Table B.12: Machine

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

Table B.13: 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 B.14: PPortPrototype

Class	*PortPrototype* (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
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.
portPrototype Props	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype.
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table B.15: PortPrototype

Class	**RPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType

Table B.16: RPortPrototype

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***
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 B.17: RootSwCompositionPrototype

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 B.18: SenderReceiverInterface

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 B.19: 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	AdaptiveApplicationSwComponentType, *AtomicSwComponentType*, CompositionSwComponentType, ParameterSwComponentType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
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

▽

△

Class	***SwComponentType*** (abstract)			
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
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

Table B.20: SwComponentType

Class	**System**			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The top level element of the System Description. **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***
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.
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects relevant in the System Description. **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
systemVersion	RevisionLabelString	0..1	attr	Version number of the System Description.

Table B.21: System

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*, LatencyTimingConstraint, OffsetTimingConstraint, 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 B.22: 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 B.23: 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.

▽

△

Class	***TimingDescriptionEvent*** (abstract)			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimingDescription*			
Subclasses	*TDEventCom*, TDEventComplex, *TDEventServiceInstance*, *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 B.24: TimingDescriptionEvent

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***
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 B.25: TimingDescriptionEventChain

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	ExecutableTiming, MachineTiming, ServiceTiming, SystemTiming, VfbTiming
Aggregated by	ARPackage.element

▽

△

Class	*TimingExtension* (abstract)			
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
timingDescription	TimingDescription	*	aggr	The timing descriptions that belong to a specific timing specification. In order to support different timing description variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingDescription.shortName, timingDescription.variationPoint.shortLabel vh.latestBindingTime=postBuild
timingGuarantee	TimingConstraint	*	aggr	The timing constraints that belong to a specific timing specification in the role of a timing guarantee. In order to support different timing constraint variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingGuarantee.shortName, timingGuarantee.variationPoint.shortLabel vh.latestBindingTime=postBuild
timingRequirement	TimingConstraint	*	aggr	The timing constraints that belong to a specific timing specification in the role of a timing requirement. In order to support different timing constraint variants within a timing specification, the aggregation is marked with the stereotype "atpVariation". **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timingRequirement.shortName, timingRequirement.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	***TimingExtension*** (abstract)			
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 B.26: TimingExtension

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 B.27: TimingModeInstance

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

Table B.28: Traceable

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*

▽

△

Class	**VariableDataPrototype**			
Aggregated by	ApplicationInterface.indication, *AtpClassifier*.atpFeature, BswInternalBehavior.arTypedPerInstance Memory, BswModuleDescription.providedData, BswModuleDescription.requiredData, BulkNvData Descriptor.bulkNvBlock, *InternalBehavior*.staticMemory, NvBlockDescriptor.ramBlock, NvDataInterface. nvData, SenderReceiverInterface.dataElement, ServiceInterface.event, SwcInternalBehavior.arTypedPer InstanceMemory, SwcInternalBehavior.explicitInterRunnableVariable, SwcInternalBehavior.implicitInter RunnableVariable			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

Table B.29: VariableDataPrototype

C Splitable Elements in the Scope of this Document

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

Each entry in 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 [5].

Name of splitable element	*Splitkey*
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 C.1: Usage of splitable elements

D Variation Points in the Scope of this Document

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

Each entry in 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 [5].

Variation Point	*Latest Binding Time*
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 D.1: Usage of variation points

E Change History

E.1 Change History of this document according to AUTOSAR Release R20-11

E.1.1 Added Specification Items in R20-11

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
[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_00027]	Purpose of TDEventComplex
[TPS_TIMEX_00032]	Purpose of VfbTiming
[TPS_TIMEX_00034]	Purpose of SystemTiming
[TPS_TIMEX_00037]	TimingConstraint is a Traceable
[TPS_TIMEX_00039]	TDEventTrigger specifies events observable at trigger ports
[TPS_TIMEX_00040]	Blueprinting VfbTiming
[TPS_TIMEX_00042]	Purpose of TDEventVfbPort
[TPS_TIMEX_00043]	Purpose of TDEventVfbReference
[TPS_TIMEX_00058]	Purpose of TDEventServiceInstance
[TPS_TIMEX_00059]	Purpose of TDEventServiceInstanceEvent
[TPS_TIMEX_00060]	Purpose of TDEventServiceInstanceField
[TPS_TIMEX_00061]	Purpose of TDEventServiceInstanceMethod
[TPS_TIMEX_00062]	Purpose of TDEventServiceInstanceDiscovery

△

Number	Heading
[TPS_TIMEX_00063]	Purpose of `MachineTiming`
[TPS_TIMEX_00064]	Purpose of `ExecutableTiming`
[TPS_TIMEX_00065]	Purpose of `ServiceTiming`

Table E.1: Added Specification Items in R20-11

E.1.2 Changed Specification Items in R20-11

none

E.1.3 Deleted Specification Items in R20-11

none

E.1.4 Added Constraints in R20-11

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_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`

▽

△

Number	Heading
[constr_4521]	Specifying attribute `synchronizationConstraintType`
[constr_4522]	`SynchronizationTimingConstraint` shall either reference events or event chains
[constr_4543]	Maximum value of the parameter `minimumInterArrivalTime`
[constr_4544]	Specifying `patternLength`, `patternJitter` and `patternPeriod`
[constr_4551]	Use only Numericals in `TDEventOccurrenceExpression`
[constr_4552]	Restricted usage of `AutosarVariableInstance` for Content Filter

Table E.2: Added Constraints in R20-11

E.1.5 Changed Constraints in R20-11

none

E.1.6 Deleted Constraints in R20-11

none

E.2 Change History of this document according to AUTOSAR Release R21-11

E.2.1 Added Specification Items in R21-11

Number	Heading
[TPS_TIMEX_00069]	Purpose of `TimingDescriptionEvent`
[TPS_TIMEX_00070]	Purpose of `TimingDescriptionEventChain`
[TPS_TIMEX_00071]	`EventTriggeringConstraint` specifies occurrence behavior respectively model
[TPS_TIMEX_00072]	`LatencyTimingConstraint` specifies latency constraints
[TPS_TIMEX_00073]	`AgeConstraint` to specify age constraints
[TPS_TIMEX_00074]	`SynchronizationTimingConstraint` specifies synchronicity constraints
[TPS_TIMEX_00075]	Optional use of timing extensions
[TPS_TIMEX_00076]	`PeriodicEventTriggering` specifies periodic occurrences of events
[TPS_TIMEX_00077]	`SporadicEventTriggering` specifies sporadic occurrences of events
[TPS_TIMEX_00078]	`ConcretePatternEventTriggering` specifies concrete pattern of occurrences of events

▽

△

Number	Heading
[TPS_TIMEX_00079]	`BurstPatternEventTriggering` specifies burst of occurrences of events
[TPS_TIMEX_00080]	`ArbitraryEventTriggering` specifies arbitrary occurrences of an event
[TPS_TIMEX_00081]	`OffsetTimingConstraint` specifies offset between occurrences of events
[TPS_TIMEX_00082]	Purpose of `TDEventVfb`
[TPS_TIMEX_00083]	`TDEventVariableDataPrototype` specifies events observable at sender/receiver ports
[TPS_TIMEX_00084]	`TDEventOperation` specifies events observable at client/server ports.
[TPS_TIMEX_00085]	`TDEventModeDeclaration` specifies events observable at mode ports.
[TPS_TIMEX_00086]	Purpose of `TDEventComplex`
[TPS_TIMEX_00087]	Purpose of `VfbTiming`
[TPS_TIMEX_00088]	Purpose of `SystemTiming`
[TPS_TIMEX_00089]	`TimingConstraint` is a `Traceable`
[TPS_TIMEX_00090]	`TDEventTrigger` specifies events observable at trigger ports
[TPS_TIMEX_00091]	Blueprinting `VfbTiming`
[TPS_TIMEX_00092]	Purpose of `TDEventVfbPort`
[TPS_TIMEX_00093]	Purpose of `TDEventVfbReference`

Table E.3: Added Specification Items in R21-11

E.2.2 Changed Specification Items in R21-11

none

E.2.3 Deleted 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
[TPS_TIMEX_00009]	Optional use of timing extensions
[TPS_TIMEX_00010]	`PeriodicEventTriggering` specifies periodic occurrences of events

▽

△

Number	Heading
[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_00027]	Purpose of TDEventComplex
[TPS_TIMEX_00032]	Purpose of VfbTiming
[TPS_TIMEX_00034]	Purpose of SystemTiming
[TPS_TIMEX_00037]	TimingConstraint is a Traceable
[TPS_TIMEX_00039]	TDEventTrigger specifies events observable at trigger ports
[TPS_TIMEX_00040]	Blueprinting VfbTiming
[TPS_TIMEX_00042]	Purpose of TDEventVfbPort
[TPS_TIMEX_00043]	Purpose of TDEventVfbReference

Table E.4: Deleted Specification Items in R21-11

E.2.4 Added Constraints in R21-11

Number	Heading
[constr_4569]	Restricted usage of functions
[constr_4570]	Application rule for the occurrence expression in TDEventComplex
[constr_4571]	Use references only as function operands
[constr_4572]	Restricted usage of AutosarOperationArgumentInstance for Content Filter
[constr_4573]	Restricted usage of AgeConstraint
[constr_4574]	Specifying minimum and maximum number of occurrences
[constr_4575]	Specifying minimum inter-arrival time and pattern length
[constr_4576]	Specifying pattern length, pattern jitter and patter period
[constr_4577]	TDEventVfb shall reference PortPrototypeBlueprint only in Blueprints
[constr_4578]	Only VfbTiming shall be a Blueprint
[constr_4579]	SynchronizationTimingConstraint shall reference at least two events
[constr_4580]	SynchronizationTimingConstraint shall reference at least two event chains

▽

△

Number	Heading
[constr_4581]	Specifying stimulus and response in `TimingDescriptionEventChain`
[constr_4582]	Specifying event chain segments
[constr_4583]	Referencing no further event chain segments
[constr_4584]	Specifying `stimulus` event and `response` event of first and last event chain segment
[constr_4585]	Specifying `patternLength`
[constr_4586]	Specifying attribute `synchronizationConstraintType`
[constr_4587]	Specifying attribute `synchronizationConstraintType`
[constr_4588]	`SynchronizationTimingConstraint` shall either reference events or event chains
[constr_4589]	Maximum value of the parameter `minimumInterArrivalTime`
[constr_4590]	Specifying `patternLength`, `patternJitter` and `patternPeriod`
[constr_4591]	Use only Numericals in `TDEventOccurrenceExpression`
[constr_4592]	Restricted usage of `AutosarVariableInstance` for Content Filter

Table E.5: Added Constraints in R21-11

E.2.5 Changed Constraints in R21-11

none

E.2.6 Deleted 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_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`

▽

△

Number	Heading
[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_4543]	Maximum value of the parameter `minimumInterArrivalTime`
[constr_4544]	Specifying `patternLength`, `patternJitter` and `patternPeriod`
[constr_4551]	Use only Numericals in `TDEventOccurrenceExpression`
[constr_4552]	Restricted usage of `AutosarVariableInstance` for Content Filter

Table E.6: Deleted Constraints in R21-11

E.3 Change History of this document according to AUTOSAR Release R22-11

E.3.1 Added Specification Items in R22-11

Number	Heading
[TPS_TIMEX_00094]	Standardized `category`s of `TimingDescriptionEvent` in Adaptive Platform
[TPS_TIMEX_00095]	Standardized `category`s of `TimingDescriptionEventChain` in Adaptive Platform

Table E.7: Added Specification Items in R22-11

E.3.2 Changed Specification Items in R22-11

none

E.3.3 Deleted Specification Items in R22-11

none

E.3.4 Added Constraints in R22-11

none

E.3.5 Changed Constraints in R22-11

none

E.3.6 Deleted Constraints in R22-11

none

E.4 Change History of this document according to AUTOSAR Release R23-11

E.4.1 Added Specification Items in R23-11

none

E.4.2 Changed Specification Items in R23-11

none

E.4.3 Deleted Specification Items in R23-11

Number	Heading
[TPS_TIMEX_00091]	Blueprinting VfbTiming

Table E.8: Deleted Specification Items in R23-11

E.4.4 Added Constraints in R23-11

Number	Heading
[constr_6902]	Existence of ExecutableTiming.executable
[constr_6903]	Existence of ServiceTiming.serviceInstance

▽

△

Number	Heading
[constr_6904]	Existence of `MachineTiming.machine`

Table E.9: Added Constraints in R23-11

E.4.5 Changed Constraints in R23-11

none

E.4.6 Deleted Constraints in R23-11

Number	Heading
[constr_4577]	`TDEventVfb` shall reference `PortPrototypeBlueprint` only in Blueprints
[constr_4578]	Only `VfbTiming` shall be a Blueprint

Table E.10: Deleted Constraints in R23-11

E.5 Change History of this document according to AUTOSAR Release R24-11

E.5.1 Added Specification Items in R24-11

none

E.5.2 Changed Specification Items in R24-11

none

E.5.3 Deleted Specification Items in R24-11

none

E.5.4 Added Constraints in R24-11

none

E.5.5 Changed Constraints in R24-11

none

E.5.6 Deleted Constraints in R24-11

none