



Contract number: 2004 - 026976

## Advancing Traffic Efficiency and Safety through Software Technology (ATESST)

### Report title

### Timing Model

**Authors**

Rolf Johansson, Henrik Lönn, Patrick Frey

**Summary**

Timing Model to be applicable for EAST-ADL2 including the implementation level (AUTOSAR)

**Status**

Final

**Dissemination Level**

Public

**Version number**

1.0

**Date of preparation**

2008-02-26

**1 Revision chart and history log**

<b>Version</b>	<b>Date</b>	<b>Reason</b>
0.1	2007-06-12	First Draft
0.2	2007-06-14	Minor updates in all parts of the document
0.3	2007-10-19	Redefinition of Meta Model and added review comments
0.4	2008-01-23	Update according to comments, deletion of HOP definitions on AUTOSAR level (left for TIMMO)
1.0	2008-02-26	Final adjustments, V1.0

**2 Introduction**

---

In the ATESST project modeling timing is mainly defined as an activity in WP 2, where it is a part of modeling of requirements. As the topic of modeling timing is rather large, some of the discussion is made within this document. Modeling timing is question that has got high attention from the AUTOSAR community where there was a timing team that identified a number of delicate questions. Right now there is an ITEA project called TIMMO starting up, addressing a number of timing issues. Putting everything concerning timing into one single document can make it easier for the ATESST project to communicate to the TIMMO project.

---

**3 Contract Based Design**

---

Many timing requirements in an automotive system may be very complex to analyze, needing the involvement of several modules and several companies. A timing model can be the base for a sound understanding and handling of timing issues, capturing timing related information and allowing for decomposition of such complex problems, e.g. by defining the interfaces between model elements which are important to timing issues and especially between actors. In a design-by-contract based approach as assumed in this document, the notion is that one part guarantees as a timing property, and another part may assume this as given in its analysis.

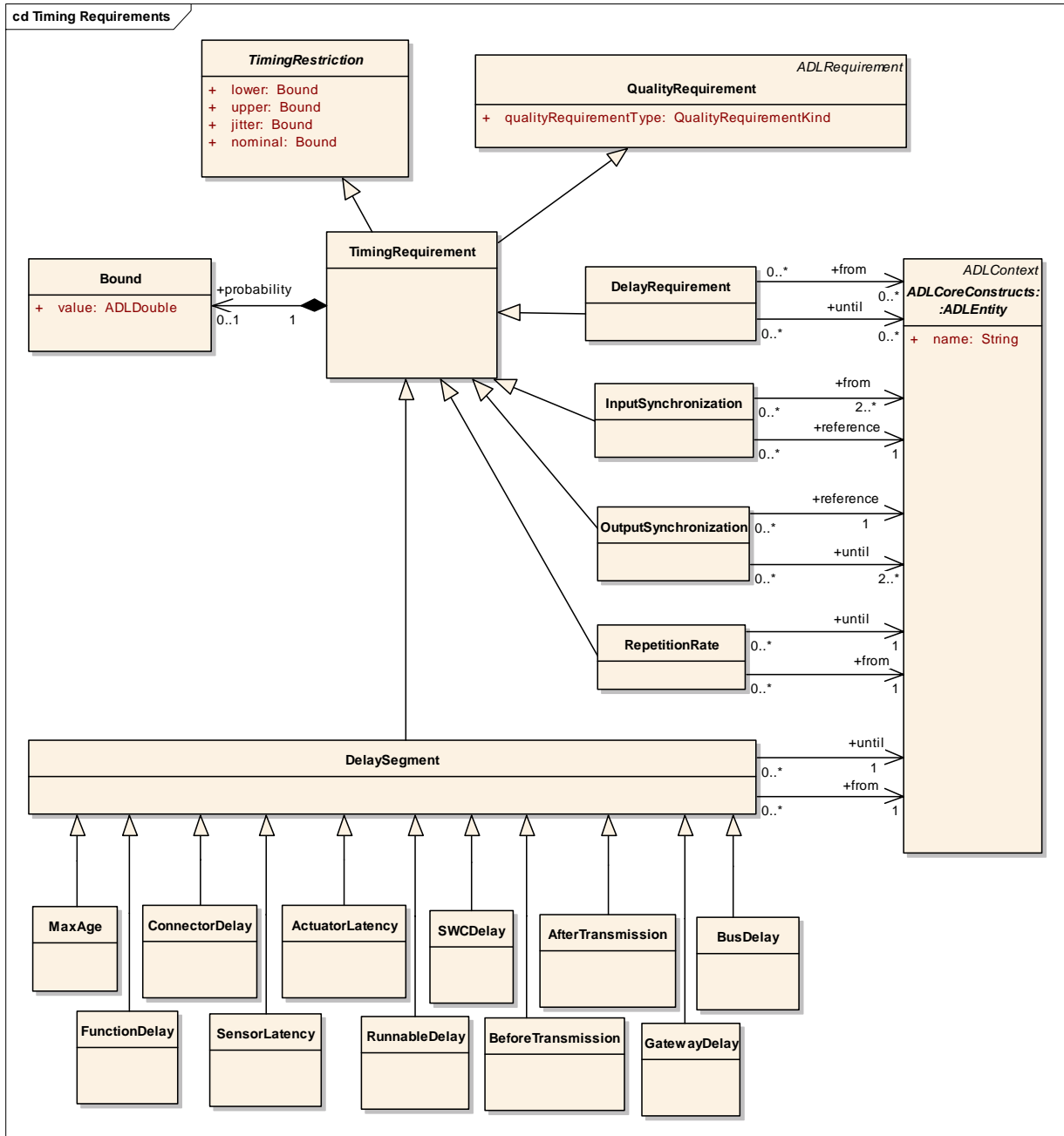
To really be applicable, and to be used properly, the parameters defined in the timing model contract need to be understandable and considered relevant by the contracting actors. This may imply that “similar” timing parameters are communicated differently in e.g. a control domain compared to the domain of body electronics.

In this document, the timing model is described based on a few examples, where the application and the contracting parties are identified, as is also the information needed to model in the contract.

Two general examples are used throughout the document. The first example is ABS functionality and the second is central locking functionality. Both of these are distributed by nature, and they may illustrate important issues as communication and synchronization. Some timing requirements of the ABS emanate from the domain of control engineering and they are periodic in nature. The user requirements (which also include timing requirements) of the central locking are related to sporadic events rather than to something periodically, and they are also outside the domain of control engineering.

**4 Timing Requirement Meta Model**

The objective of this document is to specify relevant meta-model entities in order to capture and express all relevant requirements towards timing. The class diagram shown below is an extract of what is currently defined in the meta-model of the EAST-ADL2 developed in the ATESSST project. The meta-model is kept simple, but is still able to express a number of timing requirements.



**Figure 1. EAST-ADL2 metamodel for timing constructs**

This meta model may be sufficient for modelling timing requirements (and hence timing parameters) identified throughout this document.

All the timing requirements can be associated to either a port or the start of the execution of a function; on the implementation level this means an AUTOSAR runnable. This implies that every timing requirement can be expressed as a number of required timing properties on a certain set of functions (runnables) and ports.

Furthermore, there is the notion of “firm real time” requirements, i.e. real time requirements that are not as strict as “hard real time” requirements but they are anyhow of such importance that there is a requirement to meet these requirements in “most cases” at least under “normal” conditions. The way this is modelled, is to use two different timing requirements for the same parameter, but where the conditions when they are applicable are disjoint. In the Meta model this is seen as an attribute on the ADLRequirement (i.e. a possibility not only for requirements on timing) called “Applicability”.

One way to express this could be to define extraordinary conditions as when more than  $m$  out of  $n$  events occur within a certain time interval. If more than  $m$  events happen the situation is considered as extraordinary and only the hard real time requirements are applicable. If not more than  $m$  of the  $n$  events occurs the situation is considered as normal and the (tougher) firm real time requirements are considered applicable. An example of this could be that when a driver presses more than five buttons during the interval of 500 ms, the required response time for the window lift to start is 800 ms instead of 150 ms. The normal condition is thus the assumption that never more than five buttons are pressed within 500 ms. And for these normal conditions the firm real time requirement for the window lift to start is 150 ms. The hard real time requirement that always should hold is that the window lift should start 800 ms after the button is pressed.

---

**5 Timing Requirements**

---

A timing model has the purpose of making it possible to

- Formulate all timing requirements
- Analyze timing requirements
- Break down timing requirements to modules and actors (companies), thus supporting contract based design
- Assess fulfilment of timing requirements

Timing requirements may arise on any level of abstraction. Timing requirements may be as well external as internal. External timing requirements are those formulated on a vehicle feature level. They are external in the sense that they arise directly from the external requirements either as explicitly formulated on the vehicle feature level, or as identified in the design-independent modelling on the analysis level. This means that these requirements are not a consequence of a choice of neither design nor implementation.

On the vehicle feature level the timing requirements may be either expressed as explicit values or implicit. In case a requirement is expressed implicitly on the feature level it should be set to an explicit value on the analysis level. An example of an implicit requirement on the vehicle feature level is that a response should be “almost immediate”. On the analysis level this ergonomic requirement is to be defined in terms of milliseconds.

All these external timing requirements can be traced on lower levels of abstractions, but there will also appear new timing requirements that appear as a consequence of the design and implementation decisions made.

In this document the focus of a unified timing model is on the analysis and design levels and the implementation level. The reason for not having a special focus on the vehicle feature level is that either timing requirements on this level are the same as on the analysis level, or they are not explicitly expressed as either of a quantified delay requirement, repetition rate requirement or jitter synchronization requirement. This means that what is modelled on the analysis level is considered sufficient to cover formalised timing requirements on the vehicle feature level also. So the focus is on the analysis level, which serves as the basis for decisions on architecture, and on the implementation level where all AUTOSAR components are described.

---

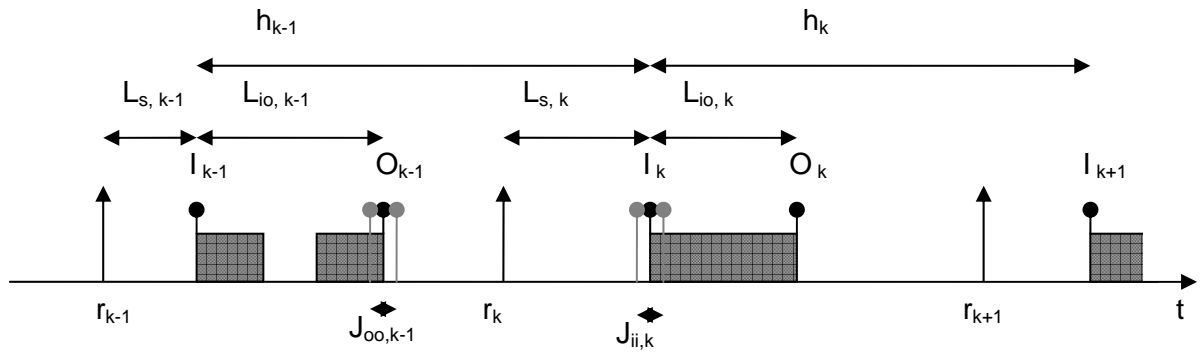
**5.1 Analysis Level**

---

The purpose of the analysis level is to define an abstract functional architecture to analyse the vehicle’s functional content prior to design. The focus is to define what the vehicle functions should do as a basis for design. This abstract definition of the functions is also the context where preliminary timing properties are defined and analysed.

5.1.1 Control Loop Timing

It is on the analysis level a control engineer defines the control algorithms and defines the corresponding timing requirements. A typical timing model can in this use case can be expressed as following:



At the time  $r_k$  the task  $k$  is released. The input is sampled at  $I_k$  and the corresponding out occurs at  $O_k$ . Latency and jitter parameters are defined as follows (within parentheses is what kind of specialisation of TimingRequirement):

- $h_k$  : Sampling Interval (RepetitionRate)
- $L_s$  : Sampling Latency (DelayRequirement)
- $L_{io}$  : Input-Output Latency (DelayRequirement)
- $J_{ii}$  : Input Synchronization Jitter (InputSynchronization)
- $J_{oo}$  : Output Synchronization Jitter (OutputSynchronization)
- $J_h = \text{maximum}(h) - \text{minimum}(h)$  : Sampling Interval Jitter (RepetitionRate)
- $J_s = \text{maximum}(L_s) - \text{minimum}(L_s)$  : Sampling Jitter (DelayRequirement)
- $J_{io} = \text{maximum}(L_{io}) - \text{minimum}(L_{io})$  : Input-Output Latency Jitter (DelayRequirement)

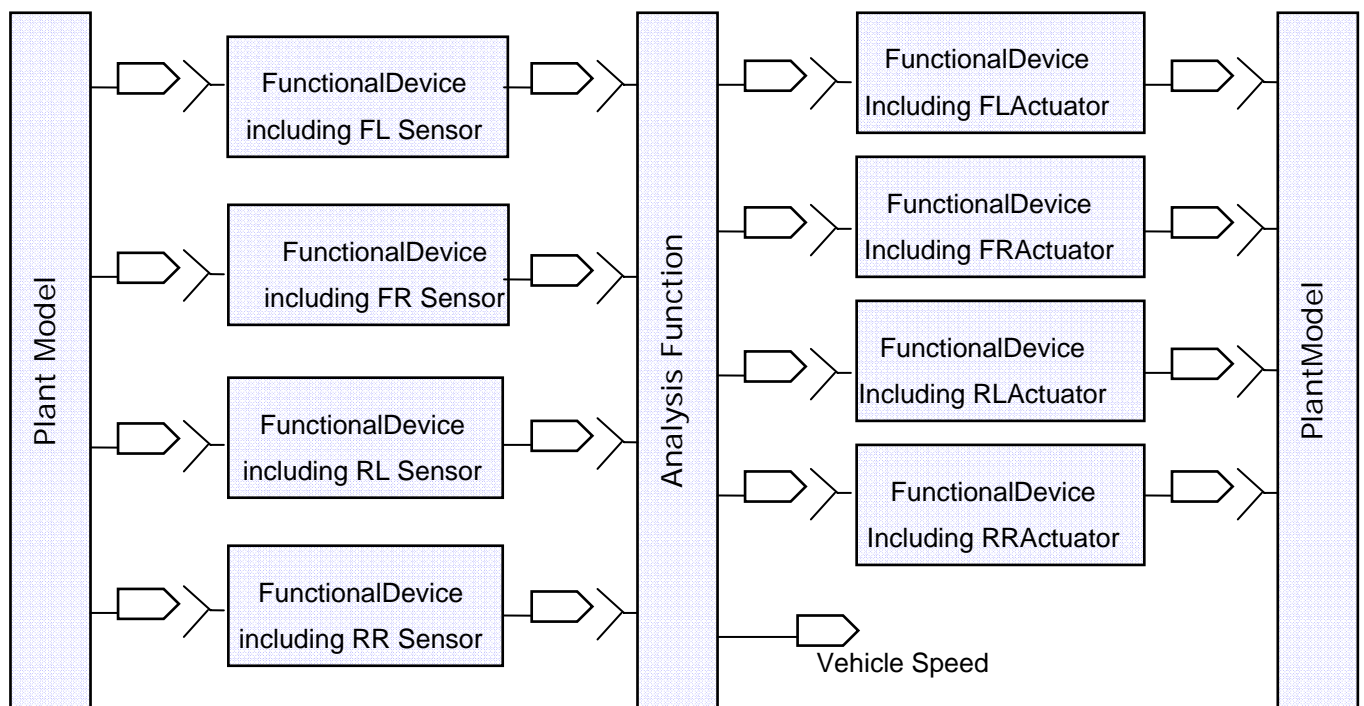
Note that the events indicated as  $I$  and  $O$  respectively above can refer to more than one event each, i.e. there may be a number of sensors and actuators relevant for the same control loop. In the figure this is depicted as some grey events around a black one. The control loop may require a number of inputs (sensors) and generate a number of corresponding outputs (actuators). This means that when identifying min or max of a certain timing parameter, this should cover all possible candidates with respect to applicable timing chains that may show up on lower levels of abstraction. Having more than one input makes it possible to introduce the parameter Input Jitter, i.e. the difference between the first and the last related input. Similarly having more than and one output introduces the parameters Output jitter.

Example: ABS

There are four wheels, and for each wheel there is one sensor and one actuator. Besides controlling each wheel, the ABS should also generate the value of the “vehicle speed” to other functions.

As mentioned above, the points in time I and O, may each refer to a number of events as a consequence of multiple instances of sensors and/or actuators for a certain control loop. Modelling on the analysis level should make this visible and serve as the basis for decisions on choosing a certain control algorithm, and what architectures this may imply.

Describing the functionality of the ABS on the analysis level could be done as follows:



On this analysis level, the choice of control function is done. As a consequence of this, the timing parameters on the analysis level are set explicit.

Let’s assume the following derived/internal timing requirements s a consequence of the analysis of the functionality on the analysis level:

- Sampling interval (RepetitionRate: nominal): 5 ms
- Sampling latency (DelayRequirement: upper): 3 ms
- Input-Output latency, sensors to actuators (DelayRequirement: upper): 5 ms

From the first sensor to the last actuator

- Input-Output latency, sensors to vehicle speed (DelayRequirement: upper): 5 ms

From the first sensor to signal valid

- Input-Output latency jitter, sensors to actuators (DelayRequirement: jitter): 2 ms

Jitter of delay from the first sensor to the last actuator

- Input-Output latency, sensors to vehicle speed (DelayRequirement: jitter): 3 ms

Jitter of delay from the first sensor to signal valid

- Sampling interval jitter (RepetitionRate: jitter): 1 ms

- Sampling jitter (DelayRequirement: jitter): 2 ms

- Input synchronization jitter (InputSynchronization: upper): 0,5 ms

From the first sensor to the last sensor

- Output synchronization jitter (OutputSynchronization: upper): 0,5 ms

From the first actuator to last actuator

In the analysis model these identified requirements can be expressed as is shown for some of the requirements below:



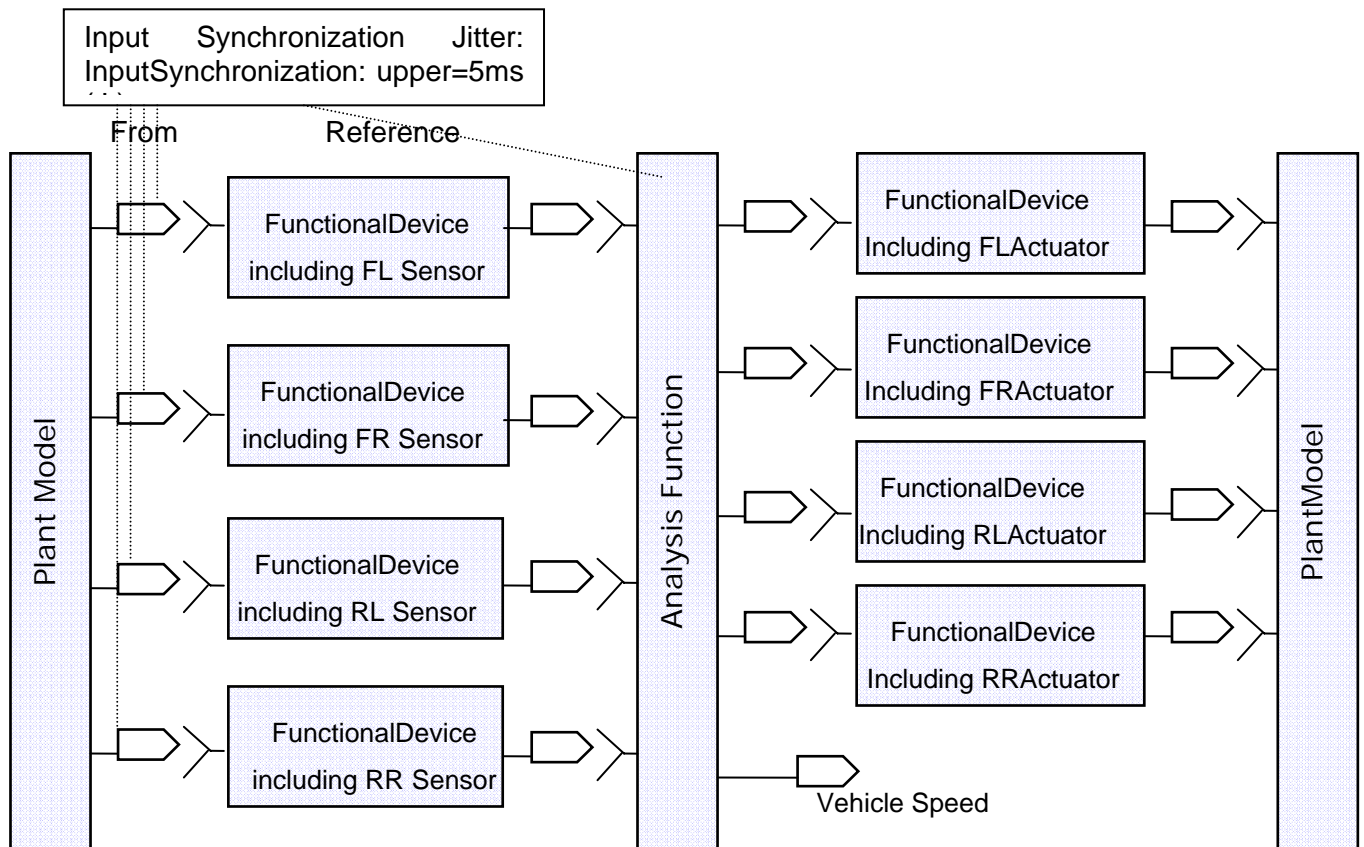
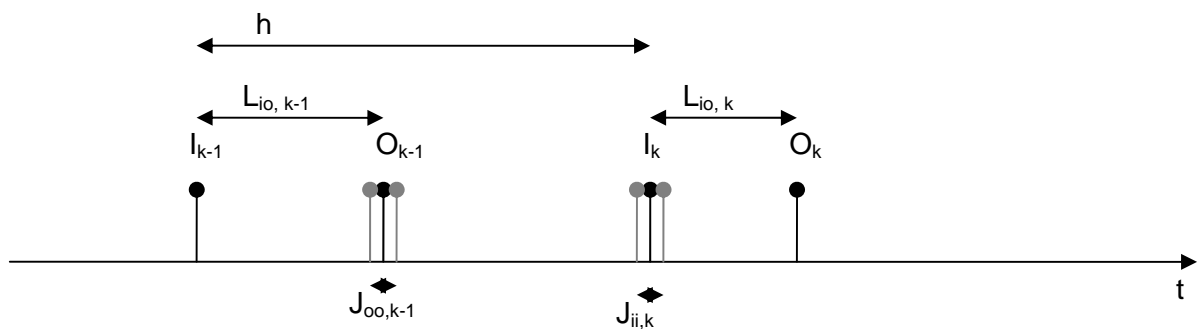


Figure 3. Illustration of timing requirement for input synchronization

5.1.2 Event Triggered Body Electronics Timing

Typically timing requirements in event triggered body electronics may look like the following on the analysis level:



$I_k$  is the point in time where there is a stimulus input event, and the corresponding output activation occurs at  $O_k$ . Latency and jitter requirements are defined as follows:

- h : (Minimum) Repetition Interval (RepetitionRate: lower)
- $L_{io}$  : Input-Output Latency (DelayRequirement)
- $J_{ii}$  : Input Synchronization Jitter (InputSynchronization)
- $J_{oo}$  : Output Synchronization Jitter (OutputSynchronization)
- $J_{io} = \text{maximum}(L_{io}) - \text{minimum}(L_{io})$  : Input-Output Latency Jitter (DelayRequirement: jitter)

Example: Central Locking

There are three different possible ways (door key, remote control, and telematics) to trigger a locking or an unlocking operation. There are four actuators, one for each door (Front Left, Front Right, Rear Left, and Rear Right).

On the vehicle feature level we can assume that there is an output synchronization requirement that the door modules should lock or unlock “almost synchronously” (no machine gun sound). Furthermore we could assume that the user of the remote control cannot expect to subsequently lock and unlock (or unlock and lock) more often than what it takes to do two distinct button presses after each other.

On the analysis level these ergonomically defined requirements can be refined to explicit number and modelled as:

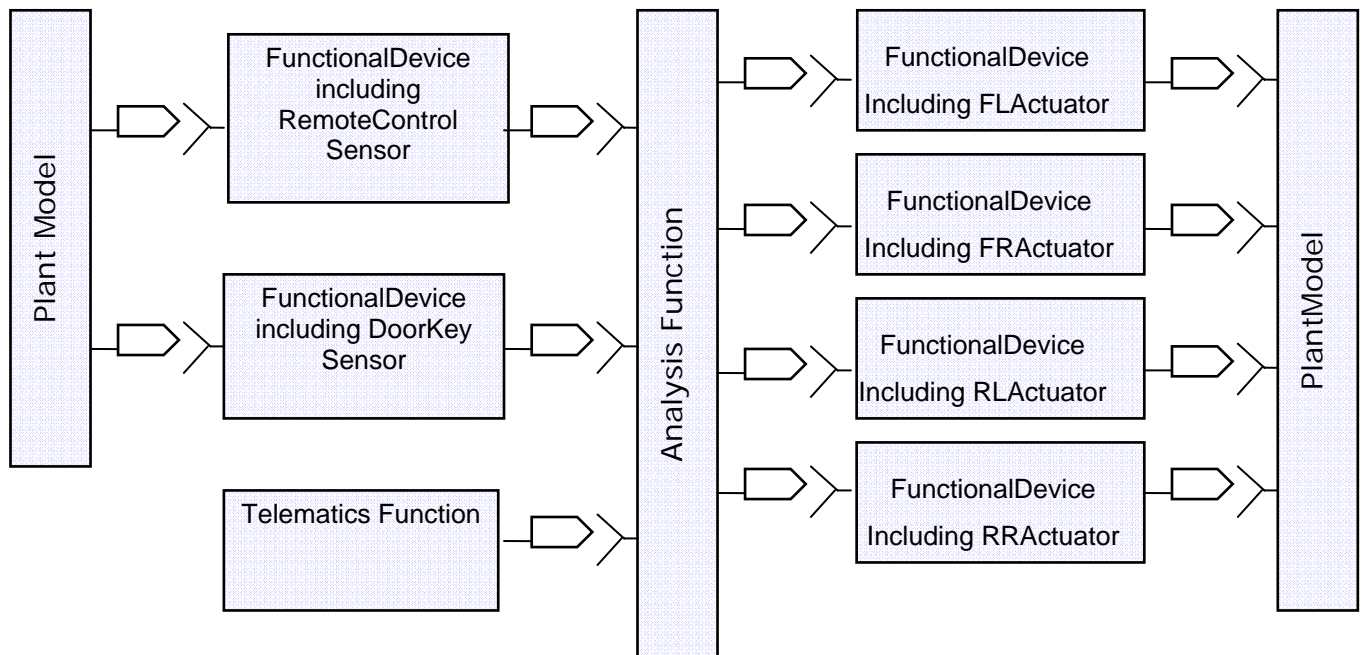
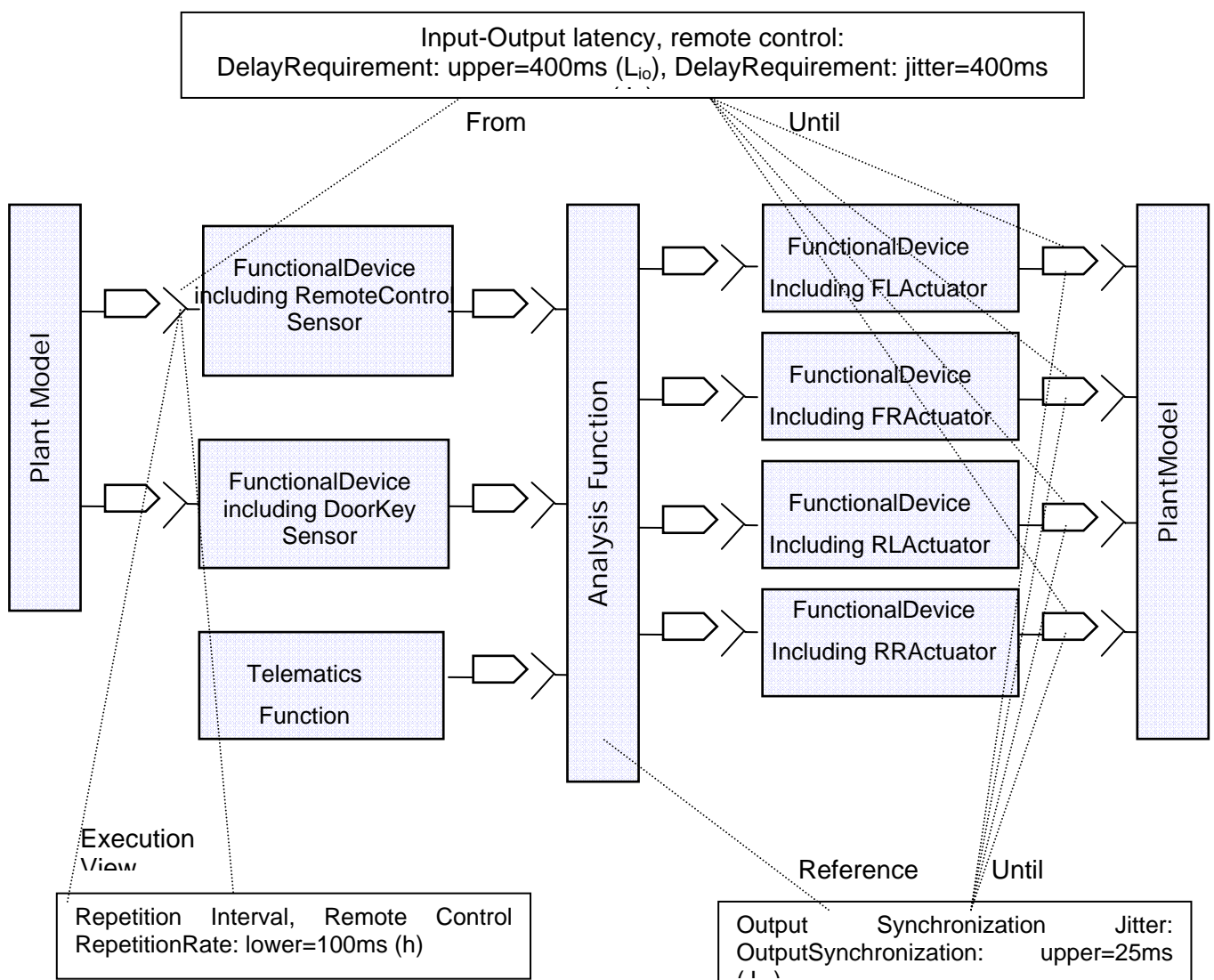


Figure 4. Abstract entities forming the Central Locking system on Analysis Level

Let's assume the following required timing parameter values as a consequence of the analysis of the functionality on the analysis level:

- (Minimum) Repetition Interval (RepetitionRate: lower): 100 ms
- Input-Output Latency (for all three ways of activation) (DelayRequirement: upper): 400 ms
- Output Synchronization Jitter (OutputSynchronization: upper): 25 ms
- Input-Output Latency Jitter (DelayRequirement: jitter): 400 ms

In the analysis model these identified requirements can be expressed as is shown for some of the requirements below:



**Figure 5. Input-Output latency and repetition interval and synchronization requirements for a remote control system**

## 5.2 Design Level

---

On the design level the focus is to describe the timing requirements and behaviour with respect to both the algorithmic behaviour **and** its realization. In addition to what is considered on the analysis level this may imply such things as:

- Fault tolerance
- Diagnosis
- Algorithm partitioning to a distributed algorithm (i.e. modularization and segmentation)

However, there are no SW components for deployment identified on this level, and for example application software components and compositions as identified by AUTOSAR do not appear above the implementation level.

For timing requirements, the same concepts apply on analysis and design level, and the same meta-model entities are used to express timing requirements on analysis and design level. . The difference lies in the level of detail and the complexity of the functional structure where the timing requirements apply. Also, the correspondence/traceability between timing requirements on design and implementation level should always be possible to derive and capture in the model. This is less often the case between analysis level and implementation level. As models on analysis level may be considerably refined and detailed before design and implementation, the link between implementation level requirements and analysis level timing requirements may be less obvious to define.

Further illustration of the examples on the Design level is TBD.

---

## 5.3 Implementation Level

---

On the implementation level the focus is to analyse the timing requirements that are relevant in the context of AUTOSAR. This means that the different application software components are visible as atomic SW-Cs and/or as Compositions as well as RunnableEntities. Furthermore, for the sensors and actuators the sensor software components and the actuator software components respectively are modelled. For the middleware this is according to the AUTOSAR basic software modules (BSW) and the runtime environment (RTE).

Timing requirements from the design level are valid also on implementation level. The requirements are reformulated to comply with the corresponding entities on implementation and internal/derived requirements are added. What timing requirements are added on this level is not a consequence of the nature of the application (as is the case on the analysis and design levels), but a consequence of the choices of the implementation details such as related to the type of bus protocol and how a timing requirement is broken down to parts in a timing budget process. In the following paragraph there is a more general analysis of the hand-over-points (HOPs) and timing chain segments in a AUTOSAR context, that are considered as a standard in this timing model.

## 6 General Delay Chain Segments

In this section is described how an (end-to-end) Delay can be further divided into delay segments, and how these are defined. For the timing requirements not being **Delay** i.e. **Output synchronization jitter**, **Input synchronization jitter** and **Repetition rate**, they cannot be further divided into smaller parts.

A Delay can be regarded on different levels of abstractions as well as on different levels of granularity. Please note that going down in level of abstraction doesn't imply that the level of granularity must increase. Furthermore, a single Delay requirement can have a cardinality higher than 1 for both the associations From and Until. This implies that the topology can be more complex than just one single chain. When considering the properties upper (max delay) and lower (min delay) of a certain Delay, these can be constrained by different chains, as pointed out in the previous chapters. In the following is assumed that for each timing Delay requirement, the relevant timing chain is already identified, and the task is about dividing this certain timing chain into well defined segments in order to make the contract based design methodology easier.

On the vehicle feature level we consider here that a formulated timing requirement concerns the entire functionality including sensors and actuators. Such a delay covering the feature requirement from input to output is called an **End-to-end Delay**. Please note that on the vehicle feature level, this may or may not be formulated with a numerical value in seconds. There is an alternative that the formulation is implicit here (e.g. "fast enough not to annoy the driver").

On the analysis level functional devices (representing sensors and actuators, respectively) and ADLFunctions are identified. These ADLFunctions may or may not be elementary. The delay segments through a functional device including a sensor or an actuator are called **Sensor latency** and **Actuator latency**, respectively. The delay through an ADLFunction is called **Function delay**, and the delay through a connector is called **Connector delay**. The borders, i.e. the handover points (HOPs) between these segments are the ports. The analysis level may be used for different levels of details, where an ADLFunction may be divided into several, thus representing a distributed algorithm.

On the design level all ADLFunctions are expressed as elementary ADLFunctionPrototypes These are the smallest entities w.r.t. to algorithmic behaviour, and one or more elementary ADLFunctions are mapped to one AUTOSAR RunnableEntity on the implementation level. The set of possible segments are the same as on the analysis level (Sensor latency, actuator latency, function delay, and connector delay). Please note that one Function delay segment on the Analysis level can be represented by a chain of several Function delay segments and Connector segments, on the design level (or on the analysis level when considering a hierarchical ADLFunction), i.e. modularization and decomposition is supported.

On the implementation level the AUTOSAR entities can be described on different levels of granularity. On a very detailed level, all the runnables are visible. The delay segment through a runnable is called **Runnable delay**. One or more runnables are located in one atomic SWC. Note that Runnable delay is the response time of the runnable and not only its worst case execution time. Please note the different categories of runnables. RunnableEntity Cat1: run-to-completion, non-blocking (i.e. no waiting on external events); Runnable Entity Cat2: can block due to waiting on events. Furthermore note the difference between worst case execution time (WCET) and worst case response time (WCRT). WCET is the accumulated worst-case execution time for a specific task (excluding the time when a task is e.g. preempted) WCRT includes OS overheads.

On a detailed level, all bus segments and the correspondent BSW are also identified. The segment between RTE and the bus is called **Before Transmission delay (BT)**. The segment from the bus through the COM stack to the RTE is called the **After Transmission delay (AT)**. The segment on the bus between these two is called **Transmission delay (T)**. In case of a gateway between bus segments, the delay through the gateway is represented by the segment **Gateway delay (GW)**. These segments and their hand-over points (HOPs) are dependent on the bus protocol, and they are further defined in the coming chapter.

For each atomic software component (SWC) it is possible to put a requirement on the propagation delay represented by the segment **SWC delay**. The HOPs of such segments are more precisely defined by the segments it hands over to, as is considered as internal AUTOSAR and hence outside the scope of this document. Anyhow, it includes the RTE, which implies that two SWCs collocated in the same ECU are represented as two SWC delay segments following directly after one another.

On a less detailed level of granularity, the communication between SWC located on different ECUs is just described as a bus including BSW, without further description of how these are implemented. Such a segment representing the entire delay between two SWC ports on different ECUs is called a **Bus delay**. A Bus delay can as well represent one bus segment ending with a gateway, as a more abstract bus including two or several bus segments connected by gateways.

On a less detailed level of granularity, the connectors between the SWCs are not identified as either implemented completely by RTE or through bus communication, but just as connectors. As for Analysis and Design levels such a segment is called Connector delay.

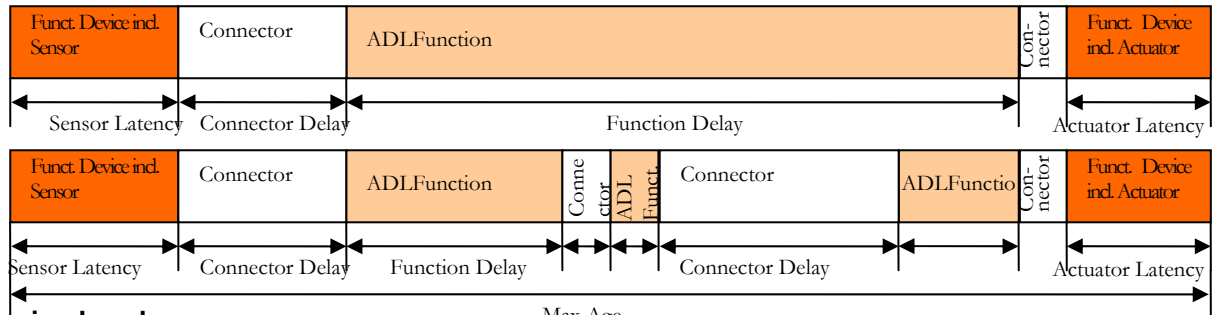
There is a possibility to leave out even more details using the AUTOSAR description entities, and this is done by Compositions. In this timing model compositions are treated as atomic SWCs, i.e. they are described by the delay segment SWC delay.

Below is sketched one delay sequence through one single timing chain. This is done for different levels of abstraction and levels of granularity, respectively. The delay segments as described above are identified in the figure. All these delay segments correspond to the delay segments defined in the meta model as described in a previous chapter.

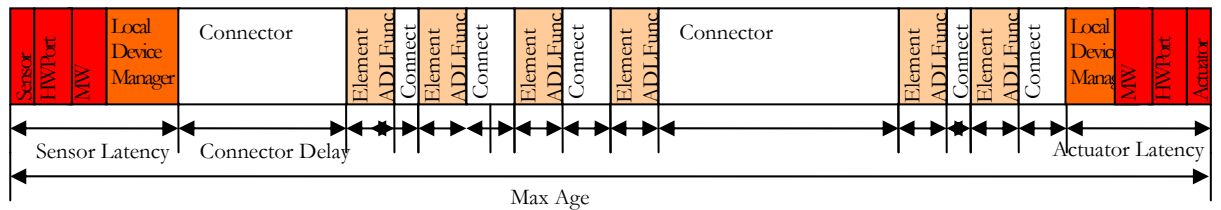
Vehicle Feature Level



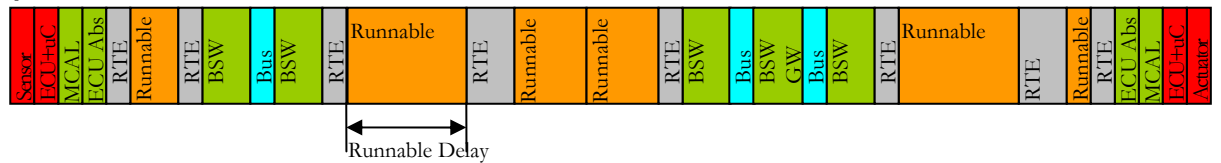
Analysis Level



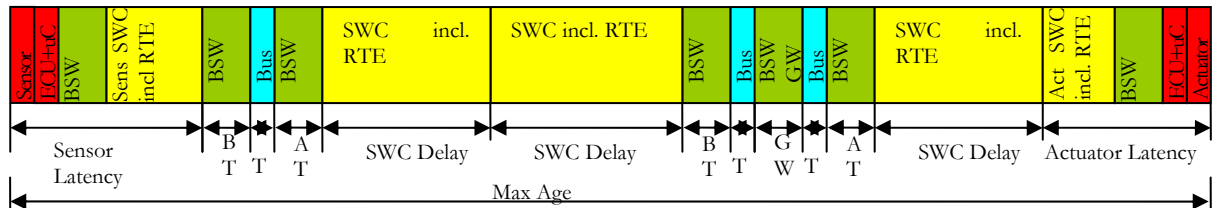
Design Level



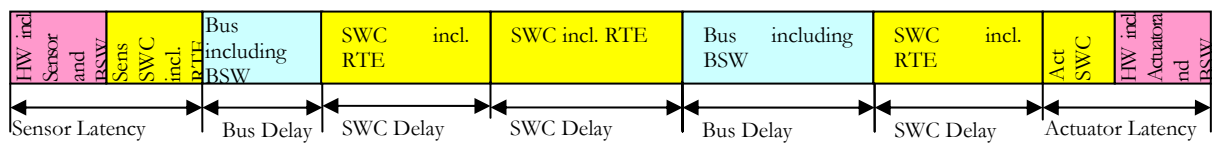
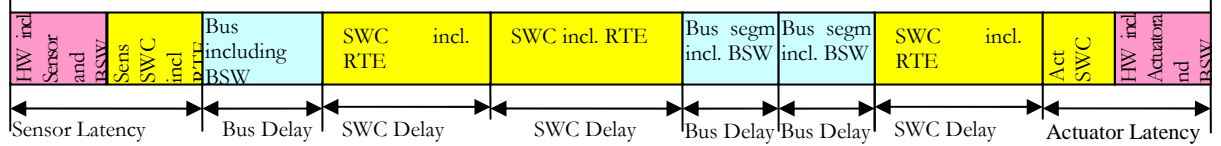
Implementation Level



ECU Internal View



System Topology View



VFB View

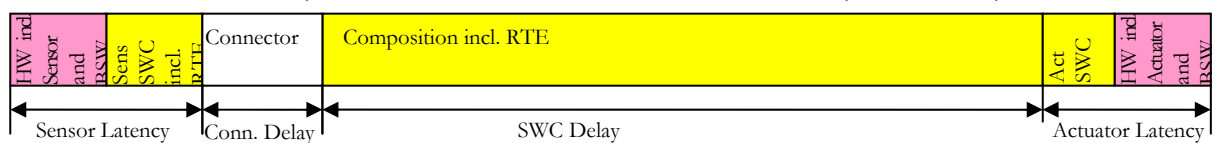
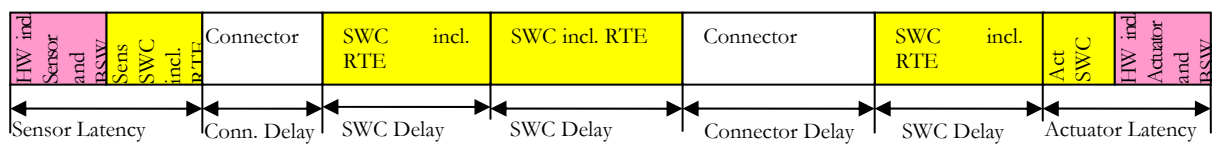


Figure 6. Informal definition of timing segments on different abstraction levels

---

**7 AUTOSAR Timing Chain Segments and Hand-Over-Points on the Implementation Level**

---

On the implementation level, the AUTOSAR basic software is the coming standard to which this timing model should be conformant. EAST-ADL2 provides timing requirements constructs that allows timing behaviour of the AUTOSAR software architecture to be defined. Further, the role of the EAST-ADL2 framework is to provide traceability of timing requirements on the AUTOSAR level to the more abstract timing requirements stemming from ergonomics, safety, performance, etc. of the functions on design level and above.

Following the concept of contract based design, it is essential to identify the hand-over points (HOPs) being on the boarder between two actors. If the actor is an ECU integrator, one HOP of interest is the one close to the bus to which the ECU is connected, and another is the HOP close to RTE where the application software developer has the responsibility. However, further definitions of the HOPs on AUTOSAR level is outside the scope of the ATESSST project.

**8 Conclusion**

---

This document contains an description the timing concepts discussed in the ATESST project. Most of these concepts are included in the EAST-ADL2 language, although some concepts on implementation level are considered AUTOSAR-specific and left out for the time being. Future work will settle whether the EAST-ADL2 timing concepts that apply to implementation level will be part of EAST-ADL2 or AUTOSAR.