Document Title	Timing Analysis
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	645
Document Classification	Auxiliary

Document Status	Final
Part of AUTOSAR Release	4.2.2

Document Change History

Release	**Changed by**	**Description**
4.2.2	AUTOSAR Release Management	• Section 6.3: introduced basic timing tasks like “Collect Timing Requirement” or “Create Timing Model”. Adapted introduction of chapter 6 accordingly. • Clarified relation of the timing properties described in section 6.4 to AUTOSAR TIMEX. • improved glossary and index • New figures for improved overview of use-cases (figures 3.2 and 4.2)
4.2.1	AUTOSAR Release Management	Editorial changes only: improvements, corrections and additions. • New chapter End-to-End Timing Analysis for Distributed Functions; • Chapter Properties and Methods for Timing Analysis: additional information and restructuring; • Added further use-cases; • Added examples, see figures 1.2, 3.1 and 4.1; • Added index at the end of the document;
4.1.3	AUTOSAR Release Management	Initial version

Advice for users

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

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

Table of Contents

Bibliography

[1] Methodology
AUTOSAR_TR_Methodology

[2] Specification of Timing Extensions
AUTOSAR_TPS_TimingExtensions

[3] Software Process Engineering Meta-Model Specification
http://www.omg.org/spec/SPEM/2.0/

[4] Embedded Systems Development, from Functional Models to Implementations

[5] Unified Modeling Language: Superstructure, Version 2.0, OMG Available Specification, ptc/05-07-04
http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf

[6] System Modeling Language (SysML)
http://www.omg.org/spec/SysML/1.3/

[7] UML Profile for Modelling and Analysis of Real-Time and Embedded systems (MARTE)
http://www.omg.org/spec/MARTE/1.1/

[8] Architecture Analysis and Design Language (AADL) AS-5506A
http://standards.sae.org/as5506a/

[9] EAST-ADL - Model Domain Specification
http://www.east-adl.info/Specification.html

[10] TIMMO-2-USE

[11] Tool Support for the Analysis of TADL2 Timing Constraints using TimeSquare
http://hal.inria.fr/docs/00/85/06/73/PDF/paper.pdf

[12] Specification of Operating System
AUTOSAR_SWS_OS

[13] Scheduling algorithms for multiprogramming in a hard real-time environment
http://cn.el.yuntech.edu.tw/course/95/real_time_os/present paper/Scheduling Algorithms for Multiprogramming in a Hard-.pdf

[14] Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised
http://dl.acm.org/citation.cfm?id=1227696

[15] Pushing the limits of CAN-Scheduling frames with offsets provides a major performance
http://www.loria.fr/ nnavet/publi/erts2008_offsets.pdf

[16] Probabilistic response time bound for CAN messages with arbitrary deadlines
http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?arnumber=6176662

1 Introduction

This document represents recommended methods and practices for timing analysis and design within the AUTOSAR development process.

1.1 Objective

During the development of AUTOSAR based systems, a common technical approach for timing analysis is needed. This document describes all major steps of timing analysis needed from the definition and validation of functional timing requirements to the verification of timing requirements on component and system level. Figure 1.1 illustrates the different aspects for timing analysis. Basis for the described methods are AUTOSAR Methodology [1] and AUTOSAR timing extensions [2].

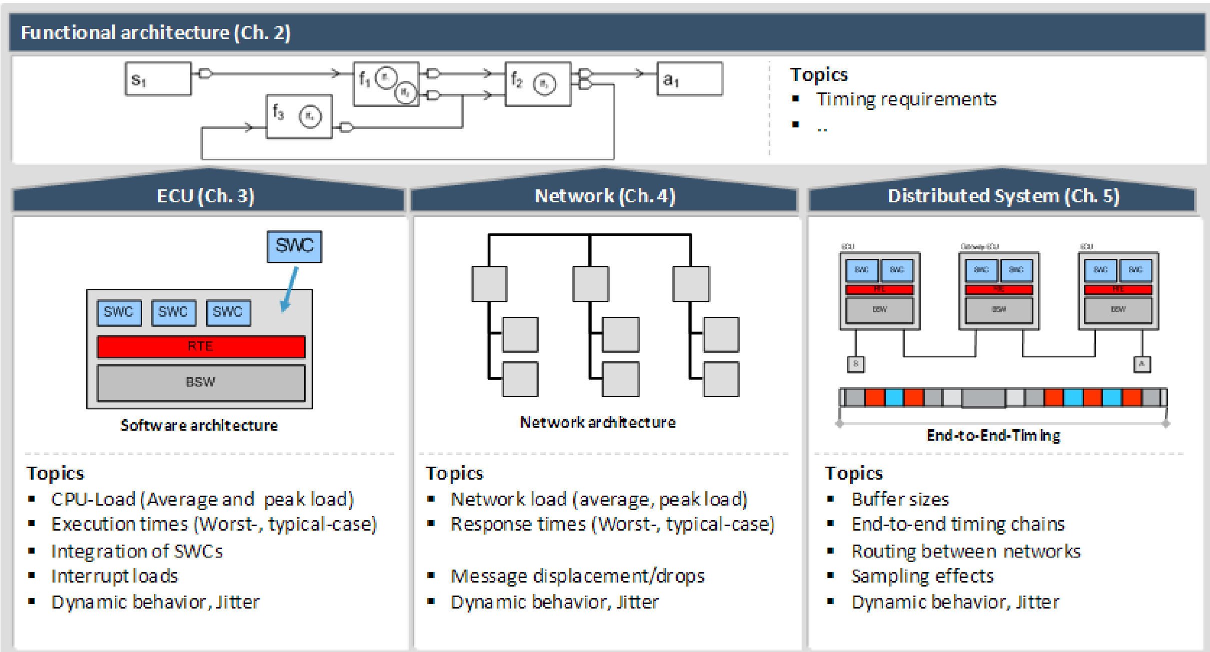

Figure 1.1: Overview of aspects for timing analysis

1.2 Overview

The AUTOSAR timing analysis methodology is divided in following parts:

- Decomposition of timing requirements and levels
- Timing analysis on the ECU level
- Timing analysis on the network level
- End-to-end timing analysis for distributed functions

- Timing properties and methods for timing analysis

For each part, a proposed methodology is presented based on a number of typical real world use-cases. A complete overview of all use-cases is given in section 1.8 on page 13.

1.3 Motivation

The increasing number of functions, complexity in E/E Architectures and the resulting requirements on ECUs and communication networks imply increasing requirements on the development process. A central part of the development process is the design of robust and extendible ECUs and network architectures.

In the development of ECUs complexity is introduced through the integration of multiple SW-Cs (constituting various functions) executed in schedulable tasks. The design and verification of the task schedules becomes difficult due to their dependencies on shared resources such as processing cores and memory.

On the network level heterogeneous network types such as CAN, LIN, FlexRay, MOST and Ethernet are in use. This makes it hard to ensure robustness, especially when routing between protocols over a gateway takes place. The design of an efficient and robust network architecture and configuration is increasingly difficult. This creates the need for a systematic approach.

These aspects must be addressed in the E/E development process together with additional requirements regarding quality, testability, ability to perform diagnostic services and so on. The overall goal is to achieve sufficient reliability and performance at a cost optimum under the requirement of scalability over several vehicle classes. In order to enable integration of additional functions over the life-cycle of a vehicle, the extensibility of an E/E architecture is of high importance.

To make optimal technical decisions during the development of E/E architectures and their components it is necessary to have suitable criteria to decide how to implement a function.

One of the most important criteria in the development of current E/E architectures is timing. Many functions are time critical due to their safety requirements. Other functions have certain timing requirements in order to guarantee a high quality (customer) function. These functions often have certain latency and jitter constrains. For distributed functions these constraints are constituted of several segments where ECU and network are the two main parts. In order to specify and analyze these timing requirements functional timing chains are important. These are described in more detail in Chapter 2.

1.4 Example

The active steering shown in the figure 1.2 visualizes an end-to-end timing constraint with a real example. The system consists of sensors, ECUs, buses and an actuator. With the vehicle dynamics model of the car and the active steering function on his mind, the functional developer defined a minimum reaction time for the outlined chain: 30ms. This becomes a top level end-to-end timing requirement for the system.

This timing requirement then gets decomposed, i.e. it gets sliced into smaller portions T1...T5, one portion for each component of the system. Obviously, ECUs and buses handle many more features with many more timing requirements, all competing for network and computation resources. On an ECU with tasks/interrupts and their runnables, the top level timing requirements are broken down into more fine grained timing requirements and the competition for resources is continued on a lower level.

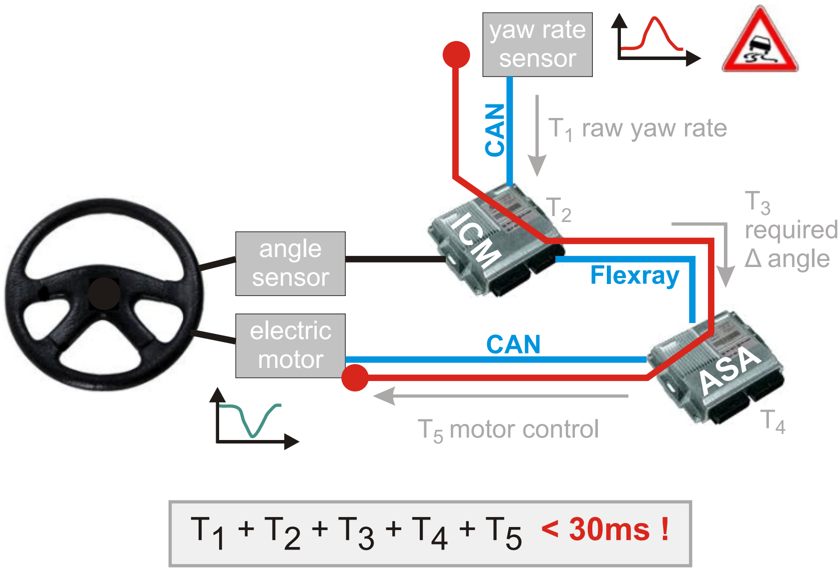

Figure 1.2: Set-up and End-to-end-timing requirement from an active steering project

1.5 Scope

This document describes how to implement timing analysis during the development of E/E systems. Similar to [1], this does not include a complete process description but rather a set of practical methods to define timing requirements and how to ensure that these requirements are met. As stated in [1], the methodology is designed to cover the needs of various AUTOSAR stakeholders:

- Organizations: Methodology is modeled in a modular format to allow organizations to tailor it and combine the methodology within their own internal processes, while identifying points where they interact with other organizations.
- Engineers: Methodology is scoped to allow engineers of various roles quickly find AUTOSAR information that is relevant to their specific needs.

- Tool Vendors: Methodology provides a common language to share among all AUTOSAR members and a common expectation of what capabilities tools should support.

The following topics are addressed:

- Definition of appropriate timing analysis methods including related timing properties for all stages of an AUTOSAR development process without the exchange of company confidential information.
- Definition of requirements for timing analysis methods enabling implementation of appropriate tools.
- Documentation of relevant experience in the area of timing analysis (Network and ECU/software) with relevant use-cases.
- Structuring of timing tasks, timing properties and related methods with regard to use-cases.
- Timing as enabler for exchange on a functional level between OEM and tier1.

Delimitation:

- Contents of this document is complementary, and not overlapping, to the contents of the AUTOSAR timing extensions [2]
- Definition of meta models to document timing attributes (e.g. AUTOSAR TIMEX)
- Definition of timing behavior for specific SW-Cs or functions in AUTOSAR.

1.6 Acronyms and Abbreviations

Abbreviation	*Meaning*
AUTOSAR	AUTomotive Open System ARchitecture
BSW	Basic Software
CAN	Controller Area Network
COM	Communication module
CPU	Central Processing Unit
DES	Discrete Event Simulation
E2E	End to end
ECU	Electrical Control Unit
ID	Identifier
I/O	Input/Output
LIN	Local Interconnect Network
NW	Network
PIL	Processor-In-The-Loop
PDU	Protocol Data Unit
RE	Runnable Entities
RTE	Runtime Environment
SW-C	Software Component
SPEM	Software Process Engineering Meta-Model
TD	Timing Description

TIMEX	AUTOSAR Timing Extensions [2]
UML	Unified Modeling Language
WCET	Worst case execution time
WCRT	Worst case response time
VFB	Virtual Functional Bus

Table 1.1: Acronyms and Abbreviations

1.7 Glossary of Terms

Term	*Synonym*	*Definition*
Event-triggered Frame	Sporadic Frame	A frame that is sent on an event triggered by the application independent from a communication schedule. The event-triggered sending is limited by a debounce time which specifies the shortest allowed temporal distance between two send events.
Execution Time		The execution time is the total time that the function needs to be assigned the resource in order to complete.
Frame		A frame is a data package sent over a communication medium. This element describes the pure Layout of a frame sent on a channel. For example, a frame on CAN and FlexRay. A commonly used synonym is “message”.
Information Packages		Smallest send able information unit on a resource (e.g. frame).
Software Task		A software task executed on a computational unit, e.g. on a CPU of an ECU.
Interrupt Load		The load of the CPU for servicing interrupts.
Load	Utilization	The load is the total share of time that a resource is used.
Period		The time period between two time-triggered send events of the same frame (e.g. 100 ms).
Response Time	Latency	The response time is the time between the activation of a function and its termination as defined in TIMEX.
Stuff bit		In CAN frames, a bit of opposite polarity is inserted after five consecutive bits of the same polarity.
System Parameter		A quantity influencing the timing behavior of the system.
Timing Task		A number of steps to accomplish a specific goal (see 6 “Description of Timing Tasks”).
Timing Constraint		A timing constraint may have two different interpretation alternatives. On the one hand, it may define a restriction for the timing behavior of the system (e.g. minimum (maximum) latency bound for a certain event sequence). In this case, a timing constraint is a requirement which the system must fulfill. On the other hand, a timing constraint may define a guarantee for the timing behavior of the system. In this case, the system developer guarantees that the system has a certain behavior with respect to timing (e.g. a timing event is guaranteed to occur periodically with a certain maximum variation). Compare AUTOSAR Timing Extension [2]
Timing Method	Technique	Defines an ordered number of steps to derive particular timing related work products (e.g. timing property, timing model)

Timing Model		A timing model collects all relevant timing information in one single place, typically tool-based. The model can be used to describe the timing behavior or it can be used to generate timing related configuration files.
Timing Property		A timing property defines the state or value of a timing relevant aspect within the system (e.g. the execution time bounds for a `RunnableEntity` or the priority of a task). Thus, a property does not represent a constraint for the system, but a somehow gathered (e.g. measured, estimated or determined) or defined attribute of the system.
Use-case	Scenario	Typical problem, broken down into tasks
Worst case		The term "worst case" denotes an upper bound on any value a certain property can take during run-time. This is usually different from and may never be smaller than the maximum value observed in the actual system. Typically worst-case values are derived using static analyses based on models of the system.
Work Product		See SPEM [3].

Table 1.2: Glossary of Terms

1.8 Use-cases

In order to show the proposed usage of timing analysis methodology a number of real-world use-cases are included in the document.

The use-cases are divided into categories using the same structure as the chapters:

- Timing analysis on the ECU level (chapter 3)
- Timing analysis on the network level (chapter 4)
- End-to-end timing analysis for distributed functions (interface between ECU and network level) (chapter 5)

Section	Use-case	Page
3.4	ECU use-case "Collect Timing Information of a SW-C"	37
3.5	ECU use-case "Select an ECU Supplier"	39
3.6	ECU use-case "Validate Timing after SW-C integration"	40
3.3	ECU use-case "Create Timing Model of the entire ECU"	35
3.7	ECU use-case "Validation of Timing"	41
3.8	ECU use-case "Debug Timing"	43
3.9	ECU use-case "Optimize Timing of an ECU"	45
3.10	ECU use-case "Optimize Scheduling"	47
3.11	ECU use-case "Optimize Code"	49
3.12	ECU use-case "Verify Timing Model(s)"	50
3.13	ECU use-case "Compare Timing Properties"	52
4.3	NW use-case "Integration of new communication"	57
4.4	NW use-case "Design and configuration of a new network"	60
4.5	NW use-case "Remapping of an existing communication link"	63
5.3	E2E use-case "Derive per-hop time budgets from End-to-End timing requirements"	67
5.4	E2E use-case "Deriving timing requirements from the timing assessment of an existing implementation"	68

Section	Use-case	Page
5.5	E2E use-case "Specify Timing Requirements for Signals/Parameters"	70
5.6	E2E use-case "Assert timing requirements against guarantees"	72
5.7	E2E use-case "Trace-based timing assessment of a distributed implementation"	74

Table 1.3: List of all use-cases in this document

1.9 Document Structure and Chapter Overview

This section contains an overview of the document and the chapter contents. Figure 1.1 on page 8 illustrates the different aspects for timing analysis and indicates the chapters in which these will be addressed.

Chapter 1 “Introduction” contains the objective, motivation, scope of the document abbreviations and glossary of terms. Additionally, a list of the use-cases is contained in section 1.8.

Chapter 2 “Decomposition of Timing Requirements” contains a short introduction of the challenge of breaking down functional timing requirements from an abstract user’s view to the implementation view of AUTOSAR timing extensions. The problem definition and different approaches and concepts for methodological solutions are introduced.

Chapter 3 “Timing Analysis for SW-Integration on ECU Level” contains use-cases for applying timing analysis at ECU level. The chapter covers several use-cases with different levels of abstraction covering the complete development workflow of an ECU ranging from a supplier nomination up to timing optimization. For every use-case the corresponding methods and timing properties are linked. This chapter is addressed mainly to ECU architects and integrators for software components (SW-C).

Chapter 4 “Timing Analysis for Networks” contains use-cases for applying timing analysis at network level, covering scenarios such as extension of an existing network, design of a new network or redesign/reconfiguration of existing network architectures. These use-cases are split into smaller tasks. For each of these tasks the necessary timing properties and the corresponding timing methods are presented. These are used to validate the timing and performance constraints typical for the corresponding use-case. This chapter is addressed mainly to system architects and integrators.

Chapter 5 “End-to-End Timing Analysis for Distributed Functions” This chapter introduces the techniques and methodology to reason about the end-to-end timing of distributed functions. As a distributed function, we consider the following two alternatives: A function that executes locally but requires data from from sensors or functions communicated over the network. In this case there exists at least an assumption about the maximum age of the data. A function that consists of several computation steps that are performed on different ECUs, connected via dedicated or shared buses. In this case event chains often exist with overall latency or periodicity constraints.

Chapter 6 “Properties and Methods for Timing Analysis” covers the timing tasks, timing properties and the methods derived from the use-cases specified in chapter 3 “Timing Analysis for SW-Integration on ECU Level” , 4 “Timing Analysis for Networks” and 5 “End-to-End Timing Analysis for Distributed Functions”. The timing methods describe how to solve the tasks derived from the use-cases of the ECU, network or both domains (i.e. End-to-End). Every single method is presented in detail including its classification, description, relation to use-cases, requirements, timing properties, inputs, boundary conditions and its implementation. Some of the methods deliver timing properties as output which can be evaluated by means of timing constraints to check the fulfillment of the timing requirement. Every single timing property is characterized by its classification, description, relation to use-cases, requirements, timing methods, format, (valid) range and implementation. The methods can be grouped in three main groups: simulation, analytical calculation and measurement; whereas the properties can be separated in two main groups: latency-like and bandwidth-like. An overview of the relation between the single methods and the single timing properties respectively is given, but also the interaction between the two is outlined.

2 Decomposition of Timing Requirements

The decomposition of timing requirements is a primary concern for the design and analysis of a real-time system. Actually, at the beginning of the system design process, timing requirements are expressed at the level of the customer functionality identified in the specification. The development of the customer functionality requires its decomposition into small and manageable components. This decomposition activity called architecting implies also a decomposition of timing requirements attached to the decomposed functionality. First section of this chapter introduces basic concepts of real-time architectures and their properties. Then, after giving an overview of the proposed approach for the decomposition of timing requirements, a focus on dedicated methodologies and their associated languages is done.

2.1 Basic Concepts of Real Time Architectures

2.1.1 Real Time Architecture Definition

An E/E architecture is the result of early design decisions that are necessary before a group of stakeholders can collaboratively build a system. An architecture defines the constituents (such as components, subsystems, ECUs, functions, runnables, compilation units ...) and the relevant relations (such as "calls", "sends data to", "synchronizes with", "uses", "depends on" ...) among them. In addition to the above-mentioned structural aspects, a real-time architecture shall provide means to fulfill timing requirements. Like for the system's constituents, real-time architecting consists in decomposing timing requirements and identifying relationships (such as refinement and traceability) among them. In fact, the timing requirements decomposition is a consequence of the structural decomposition where timing requirements are segmented upon the decomposed units. However, while structural decomposition could be driven by functional concerns, input/output data flows, and/or provided/required services, timing decomposition is a more complex task to achieve. Indeed, correct timing requirement decompositions shall be locally and globally feasible. Locally the subcomponent timing properties shall fulfill its assigned timing requirement segment. The design of a real-time software architecture consists in finding a functional decomposition and a platform configuration which timing properties allow fulfilling local and global timing requirements.

Timing properties are highly dependent on the underlying software and hardware platform resources. Moreover, access to shared platform resources by the decomposed units introduces some overhead (like blocking times or interferences ...). Timing properties will depend on:

- The chosen *placement* (e.g. allocation of functions/components on ECUs);
- The chosen *partitioning* (e.g. grouping of runnables on tasks);
- The chosen *scheduling* (e.g. tasks priority assignment, shared resources access protocol);

In order to assess these architectural choices with regard to timing requirements, timing analysis is necessary. Analysis methods and associated timing properties used for such an assessment can depend on the kind of real-time architecture under consideration (e.g. time-triggered or event-triggered architecture). Chapter 6 details this aspect. Timing analysis can be introduced at the system level as a prediction instrument for the refinement of system functions toward their implementation [4]. Although, timing analysis in early development phases requires to make assumptions about the resources of the implementation platform, it constitutes a sound guide for the decomposition and refinement of timing requirements

From the application point of view the following two timing properties are particularly important:

- *Execution and transmission times*;
- *Response times*.

First introduction of these terms is given bellow. A more detailed description and classification of these notions is provided in Chapter 6.

2.1.2 Execution and Transmission Times

The execution time of a schedulable entity (function, runnable, software task) on a computing resource (e.g. ECU) is the duration taken by the schedulable entity to complete its execution in a continuous way without any consideration of other schedulable entities that are sharing the same computing resource (no suspension/preemption).

Similarly, the transmission time of a signal/message/frame on a communication resource (e.g. bus, network) is the duration taken by the signal/message/frame to transit from its source to its destination without any consideration of other signals/messages/frames transiting on the same communication resource.

An execution/transmission time is a quantitative property that can be described with the following characteristics:

- A *statistical qualifier* (worst, best, mean/average) representing the bounds of execution/transmission time. This bound could be the upper bound which corresponds to the worst-case execution/transmission time (WCET/WCTT), the lower bound corresponding to the best-case execution/transmission time (BCET/BCTT), or the average-case execution/transmission time (ACET/ACTT) which could be useful for performance analysis. Among these three qualifiers, the WCET is the most commonly used for timing properties verification/validation of real-time systems.
- A *method* (estimation, measurement, calculation (static analysis)) denoting the way an execution/transmission time is obtained. The precision of an execution time is highly dependent on its source. For instance, input data used for measurements triggers specific branches of the function/program which impacts the measured execution time value. For that reason, measurements can only provide

average execution time or a distribution of execution times. To obtain execution time upper bound, static analysis techniques are employed (abstract interpretation, model checking ...).

- An *accuracy factor*. The accuracy of the evaluated WCET/WCTT depends on many factors among which the level of details of the software (instruction level) as well as the level of details of the execution/communication resource (like cache mechanisms). This latter could provide elements of unpredictability like branch prediction mechanisms that could affect the WCET analysis by making it more complex to achieve and too pessimistic. In order to avoid over dimensioning of execution platforms, and in order to allow accurate response time analysis (see the following subsection) WCET/WCTT analysis shall provide safe but accurate WCETs/WCTTs.

Sometimes, a WCET/WCTT can be a requirement to satisfy, especially at the very low levels of abstraction once the ECUs, network and deployment are fixed. However, in the very upper levels of abstraction, timing requirements usually refer to end-to-end response time bounds defined in the following subsection.

2.1.3 Response Time

The response time of a schedulable entity (function, runnable, task, ...) is the time duration taken by the schedulable entity to complete its execution. Unlike for execution time, the response time takes into account other schedulable entities that are sharing the same execution/communication resource. Hence, the response time of a schedulable entity comprises its execution time and additional terms induced by the concurrent access to shared resources (blocking times, jitters...). See Chapter 6 for more details.

An end-to-end response time is a response time in which several schedulable entities are involved. These schedulable entities form a chain. First schedulable entity of the chain is called the *source* schedulable entity and the last one is called the *sink* schedulable entity. The end-to-end response time is the elapsed time until the sink schedulable entity of the chain terminates its execution.

Like an execution time, a response time is a quantitative property that can be described with the following characteristics:

- A *statistical qualifier* (worst, best and mean/average). The worst-case response time (WCRT) is the upper bound usually computed by timing analyses to assess timing requirements fulfillment. A more detailed definition of statistical qualifier is given in Chapter 6.
- A *method* (estimation, measurement, calculation (static analysis)) denoting the way a response time is obtained. Methods for response time determination are given in Chapter 6.

- An *accuracy factor*. The accuracy of a WCRT is highly dependent on the accuracy of the Worst Case Executions Times of the executable entities that are involved in the chain.

2.2 Timing Requirements Decomposition Problem

Mastering timing requirements is one key success factor for the development and integration of state of the art automotive E/E-systems. Timing requirements shall be carried out continuously during the complex development process of a vehicle, and further shall be reused and exchanged for the re-use of functions or components to other vehicle projects: timing requirements have to be described systematically and carefully. The required level of detail can vary from timing constraints for high level customer related features at the vehicle level, over timing requirements for the control of a power amplifier for a particular actuator, to ECU-internal timing for data synchronicity of software functions on a multi-core microcontroller at the operational level.

As illustrated on Figure 2.1, the development process follows the well-known V-model, which describes a systematic and staggered top-down approach from system specifications to system integration. On the left branch process steps of specification are described, implementing decomposition from an entire E/E-system to single components. The base of the V describes implementation and associated test procedures. Following the right branch of the V testing and integration procedures up to vehicle system integration can be read in bottom up order.

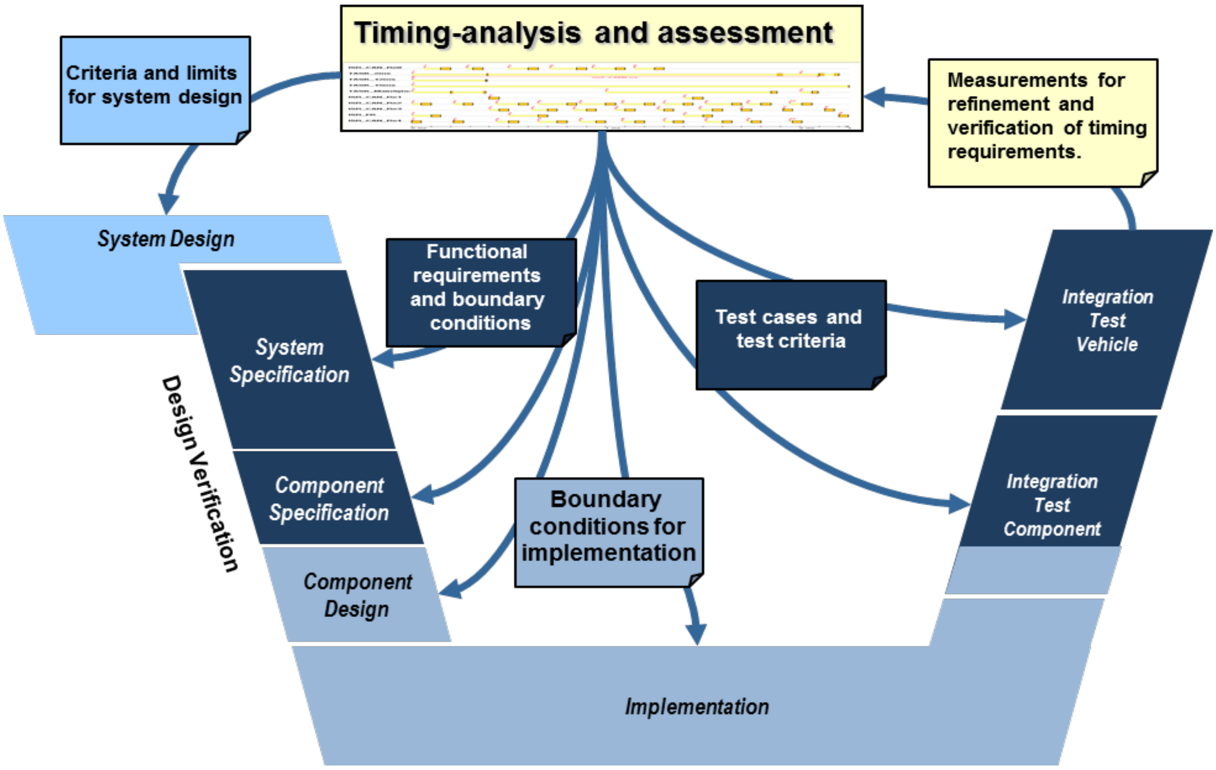

Figure 2.1: Application of timing analysis in a development process according to the V-model

According to these basic steps of an automotive OEM development process, requirements shall be traceable in any process step. This means that timing requirements shall be identifiable and traceable from a requirements specification via a supplier's performance specification to a test and integration documentation (protocols). As far as E/E-processes are concerned this means that timing requirements shall resist the process transformation between two companies like OEM an tier1-supplier and further down to tier2 and 3 suppliers. This can only be achieved by using a standardized system of description and methodology, referencing the model artifacts that are generally exchanged between development partners.

The AUTOSAR Timing Extensions (TIMEX) [2] based on the AUTOSAR System Template, represents the standardized format for exchange of a system description within an AUTOSAR compliant software development process. In addition TIMEX is an optional component which does not imply changes in the AUTOSAR System Template. The concept of the observable event, which occurs or can be observed in a referenced modeling artifact e.g. a RTE-port, allows specifying observation points and sequences of events in causal order (event chains) with additional timing constraints on them. The TIMEX concept is assumed to meet all use-cases of describing temporal behavior in an AUTOSAR system by means of timing requirements.

Unfortunately the OEM development process does not start with AUTOSAR. AUTOSAR only represents an implementation view for some software components, but not a view on higher level functional concepts that can comprise non software functions. Actually at the very beginning of the process, requirements are described in natural language. These requirements have to be "formalized" in a non-natural language in order to assess them and allow their decomposition. The assessment of timing requirements should be done as early as possible in the development process. To enable this at system/functional level, a system/functional modeling language is needed. This language must provide concepts for functions design modeling and must also provide a formal way to capture and decompose timing requirements during the functional design. Several approaches based on Architecture Description Languages (ADLs) could be used to fill the gap between requirements specification in natural language and the implementation phase modeled in AUTOSAR. We can cite UML-based [5] Architecture Description Languages: SysML [6] (UML specialization for System Modeling) and MARTE [7] (UML specialization for Modeling and Analysis of Real-Time end Embedded systems). Other approaches that are more domain specific like AADL [8] for aerospace or EAST-ADL [9] for automotive also exist. The choice of the appropriate system/functional level modeling language depends on the internal OEMs' processes. However, there are some general timing related criteria that are important to consider:

- A support for hierarchical timing requirements process;
- The ease of mapping the decomposed timing requirements to AUTOSAR TIMEX model artifacts that constitutes today the exchange format between the OEM and its suppliers.

In the following section an approach based on all these ideas and concepts is drawn which shall give orientation to implement a hierarchical timing requirements process in the own organization and also, in the end, enables the exchange of AUTOSAR TIMEX compliant model artifacts.

2.3 Hierarchical Timing Description

During the early design phase of an automotive development process the architecture discussion is about high level customer related functions. These functions can be detailed in functional "cause and effect" or "activity" chains, which from a temporal view can be budgeted - justified by customer's experience. The functional quality and thus technical effort dedicated to the customer's experience is a business decision of a company.

One example is the reaction time from pressing a button to a reaction, which varies between simply switching (rear window heating) and controlling a motion (e.g. seat or mirror adjustment). The other example is a powertrain or chassis control function which can cause inconveniences like bucking during shifting or braking, and which would not contribute to positive press reviews of a premium vehicle.

From methodological and technical view timing analysis is a tool to assure the desired temporal behavior during the mapping of a functional network to a component network as depicted on Figure 2.2.

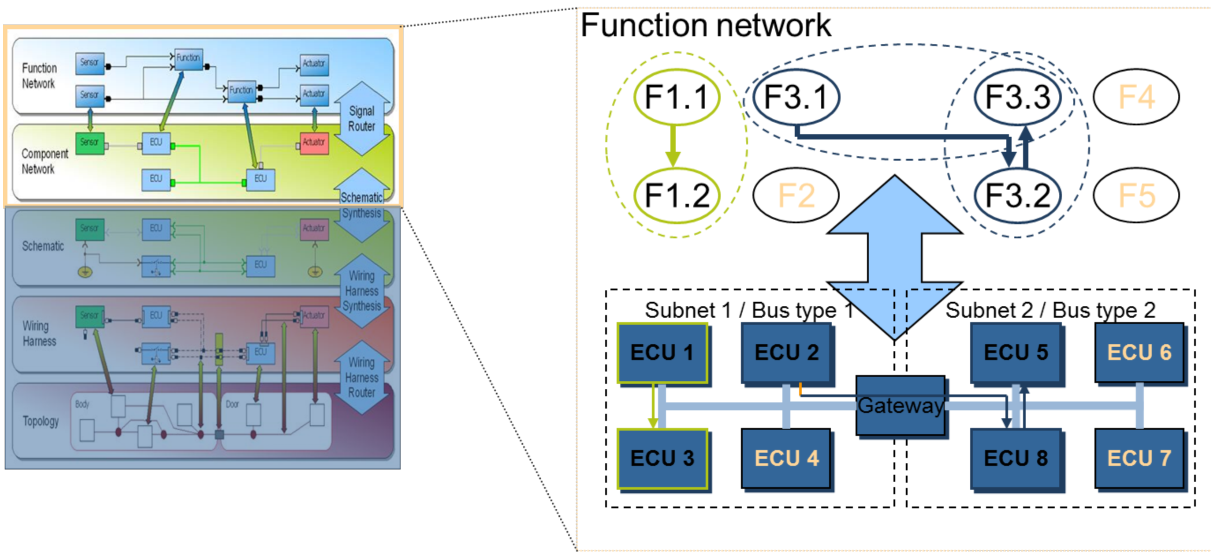

Figure 2.2: Mapping of a function network to a component network

Once the major timing budgets for customer related functions are defined and a distribution of functional parts to hardware components is done [1], a more detailed temporal

[1] In an AUTOSAR development process a software component (SW-C) is defined with a scope local to the hardware component it is mapped on. It contains a functional contribution to the vehicle function with a system wide scope.

view of a networking architecture can be made. This allows a first assessment of the feasibility of the function distribution in terms of performance and timing. This process can iteratively be refined during further process steps to have more precise analysis results.

For further understanding, it can be assumed that each function in Figure 2.3 is contained in the compositional scope of an AUTOSAR SW-C, where it is represented as an AUTOSAR runnable entity, shortly often named “Runnable”. Other mapping strategies can also be considered. Regardless of the chosen strategy, the mapping is usually constrained by the functional design choices made at the functional level for timing requirements assessment. For instance, a feasibility test founded on the computation of the load (utilization) of each hardware resource (ECUs, buses), is based on a given allocation of functions on hardware resources. This allocation must be taken into account for the mapping of functions to AUTOSAR SW-Cs in order to avoid the mapping of two functions that are allocated on distinct ECUs on the same AUTOSAR SW-C.

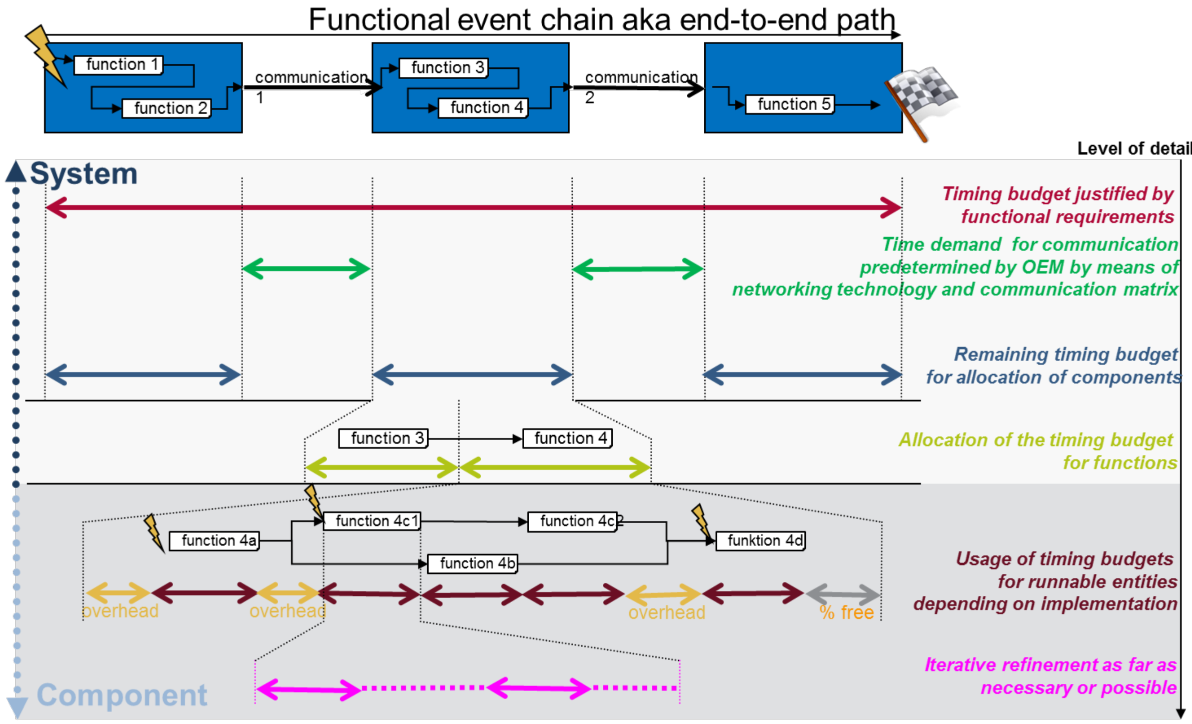

Figure 2.3: Iterative and hierarchical top down budgeting of timing requirements corresponding to response times

Moreover, in many cases timing demands of physical processes, e.g. the start-up and transient oscillation behavior of electrical actuators, consume more than a few microseconds and thus have to be considered carefully.

In a first step the overall timing budget can be split in component-internal and networking parts. As soon as the whole network communication and the type of network are known, the WCRT-analysis of a network can quantify the worst case timing demand for network communication. As shown in the picture above, this divides the overall timing budget in networking budgets and timing budgets for allocation in components (usually ECUs).

This can be enough for an OEM if the development and integration of the component is entirely done by a supplier. In practice a more detailed view considering the timing behavior of a basic software stack and the functions itself is required. Likewise functional relations are more complex, which induces a more complex analysis.

During further analysis steps the end to end timing path or chain of functions can be refined following the concepts of Figure 2.3.

In the following section we introduce methodologies that provide support for the general process described.

2.4 Methodologies for Timing Requirements Decomposition

As previously stated, the AUTOSAR methodology covers the implementation phase of the process of E/E systems development. However, timing requirements are introduced at the very beginning of the development cycle in the form of textual descriptions by OEMs. An extension of the AUTOSAR methodology is then needed to cover the system/functional architecture design phases where the first functional decompositions and timing requirements decomposition must occur. In fact, one of the most challenging activities in the development of systems is determining a system's dimensions in early phases of the development - and the most difficult one is the phase before transitioning from the functional domain to the hard and software domain.

Primarily, two questions must be answered. Firstly, how much bandwidth shall the networks provide in order to ensure proper and timely transmission of data between electronic control units; and secondly, how much processing performance is required on an electronic control unit to process the received data and to execute the corresponding functions. As a matter of fact, these questions can only be completely answered when the system is implemented, including a mapping of signals to network frames and first implementations of functions that are executed on the electronic control units. The reason for this is that one needs to know how much bits per second have to be transmitted and how much instructions shall be executed.

An important aspect that impacts the decisions taken during the task of specifying system dimensions is timing. Especially, information about data transmission frequencies, execution rates of functions, as well as tolerated latencies and required response times provide a framework for performing a first approximation of network and ECU dimensions. This framework allows to continuously refining the system dimensions during system development when more details about the system's implementation are becoming available. The basic idea is to abstract from operational parameters obtained during the implementation phase, like for example measured or simulated execution times of functions, and use them on higher levels of abstraction respectively earlier development phases. And, for new functions as a workaround for missing execution time, an activity called Time Budgeting allows the specification of so called time budgets to functions.

The remainder of this section defines the levels that will be considered for timing requirements decomposition. Then, some generic methodological guidelines will be given for conducting timing requirements refinement between these levels.

2.4.1 Functional and Software Architectures Modeling Levels

Prior to the AUTOSAR software architecture levels, we can consider two functional architecture modeling levels defined in [9] that are of interest for timing requirements:

- The *Functional Analysis level* which is centered on a logical representation of the system's functional units to be developed. Typically based on the inputs of automatic control engineering, system design at this level refines the vehicle level system feature specification by identifying the individual functional units necessary for system boundary (e.g., sensing and actuating functions for the interaction with target physical plant) and internal computation (e.g. feedback control functions for regulating the dynamics of target physical plant). The design focuses on the abstract functional logic, while abstracting any SW/HW based implementation details. Through an analysis level system model, such abstract functional units are defined and linked to the corresponding specifications of requirements (which are either satisfied or emergent) as well as the corresponding verification and validation cases.
- The *Function Design level* provides a logical representation of the system functional units that are now structured for their realizations through computer hardware and software. It refines the analysis level model by capturing the bindings of system functions to I/O devices, basic software, operating systems, communication systems, memories and processing units, and other hardware devices. Again, through a design level system model, the system functions, together with the expected software and hardware resources for their realizations, are defined and linked to the corresponding specifications of requirements (which are either satisfied or emergent) as well as the corresponding verification and validation cases. Moreover, the creation of an explicit design level system model promotes efficient and reusable architectures, i.e. sets of (structured) HW/SW components and their interfaces, hardware architecture, for different functions. The architecture must satisfy the constraints of a particular development project in automotive series production.

The AUTOSAR methodology (see [1] for a general introduction) provides several well defined process steps, and furthermore artifacts that are provided or needed by these steps. Figure 2.4 provides a simplified overview of the AUTOSAR methodology, using the Software & Systems Process Engineering Metamodel notation (SPEM) [3], focusing on the process phases which are of interest for the use of the timing extensions. These represented steps and artifacts are grouped by boundaries in the five following views:

- *VfbTiming* deals with timing information related to the interaction of SwComponentTypes at VFB level.
- *SwcTiming* deals with timing information related to the SwcInternalBehavior of AtomicSwComponentTypes.
- *SystemTiming* deals with timing information related to a System, utilizing information about topology, software deployment, and signal mapping.
- *BswModuleTiming* deals with timing information related to the BswInternalBehavior of a single BswModuleDescription.
- *EcuTiming* deals with timing information related to the EcucValueCollection, particularly with the EcucModuleConfigurationValues.

Further details of these timing views are given in [2].

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.

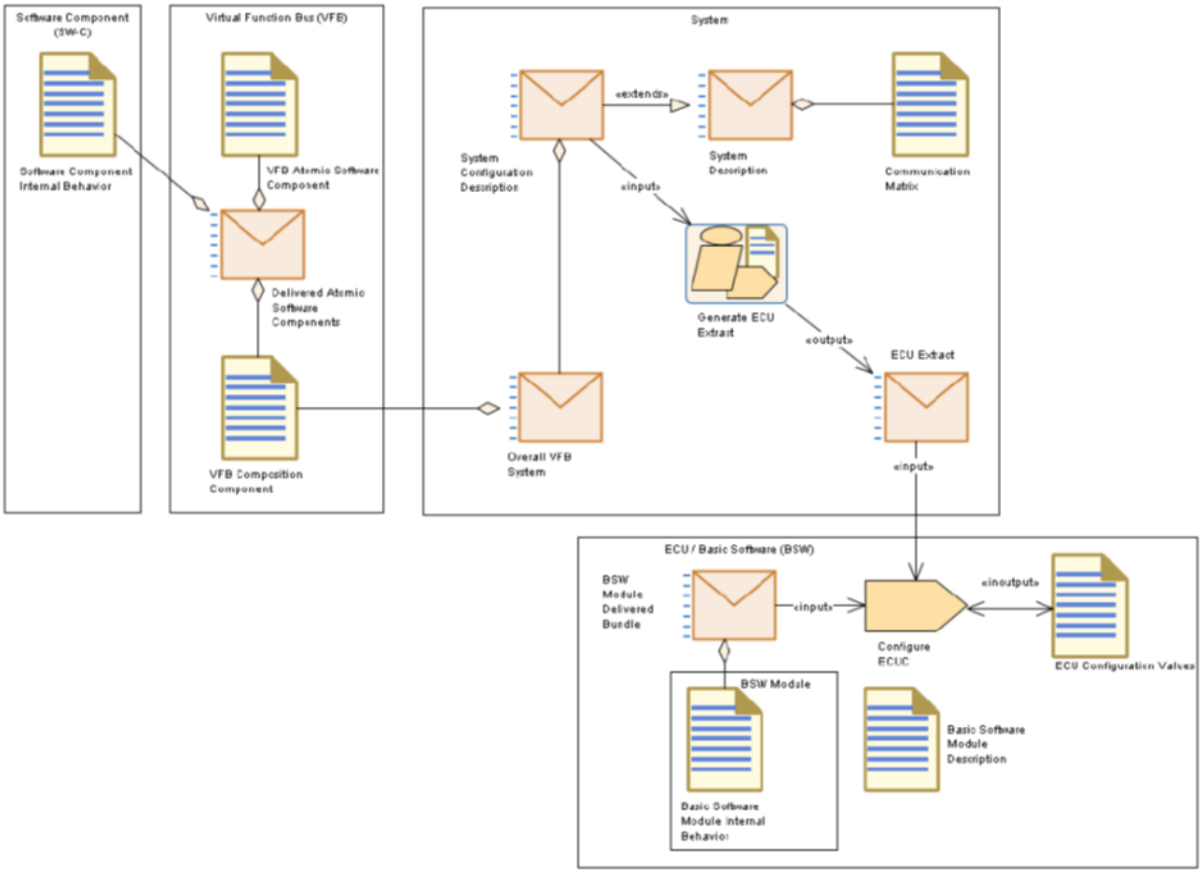

Figure 2.4: SPEM Process model from AUTOSAR Methodology for system design process

2.4.2 Guidelines for Timing Requirements Decomposition

The Generic Methodology Pattern (GMP) developed in the TIMMO-2-USE project [10] is an example of a process that defines generic steps for timing requirements refinement. Theoretically, those generic steps are applicable at every level defined in the previous section (including the AUTOSAR levels). Basically, at each abstraction level, GMP takes as input timing requirements and after a sequence of steps gives as output refined timing requirements. GMP defines six main steps. Some of them have been merged in the following short description:

- *Step1 - Create Solution*: describes the definition of the architecture without any timing information. This step can consist in a refinement of an already existing architecture coming from the upper level. Timing requirements shall guide the creation or revision of a solution.
- *Step2 - Attach Timing Requirements to Solution*: describes the formulation of timing requirements in terms of the current architecture. This can imply a transformation of timing requirements coming from the previous level, in order to be compliant with the timing model of the current level of abstraction. For instance in the AUTOSAR SwcTiming view a timing requirement can be modeled with a timing constraint attached to events or event chains.
- *Step 3 - Create, Analyze and Verify Timing Model*: describes the definition of a formalized model for the calculation of specific timing properties of the current architecture. In this step relevant timing analysis methods can be applied to verify timing requirements against calculated timing properties (e.g. maximal load for a bus). If timing requirements are not verified by timing properties resulting from the analysis, the previous steps shall be iterated until a satisfactory solution is found.
- *Step 4 - Specify and Validate Timing Requirements*: describes the identification of mandatory timing properties and their promotion to timing requirements for the next level.

Chapter 6 gives for each use-cases described in Chapter 3 and Chapter 4 timing properties and methods of interest to ensure correct timing requirements decomposition.

2.5 Languages for Timing Requirements Specification

The steps described in the previous section require one or more modeling languages to be used with. The AUTOSAR methodology is based on the AUTOSAR language and its timing extensions. AUTOSAR is the language for the software implementation levels but not applicable at the functional levels (analysis and design). Therefore, in order to ensure a complete model-based approach for timing requirements decomposition, a complementary modeling language for functional levels has to be used. EAST-ADL2 [9] and its timing extension TADL2 [11] allow functional levels specification with precise timing models. Moreover, TADL2 and AUTOSAR Timing extensions are sharing the same base concepts which may facilitate the translation of timing requirements from the functional level to the AUTOSAR level (where timing requirements are expressed with TIMEX).

Therefore, EAST-ADL / TADL is briefly presented as an example of modeling language for the support of the functional levels of a methodology for timing requirements decomposition.

2.5.1 EAST-ADL / TADL

EAST-ADL is an Architecture Description Language (ADL) for automotive embedded systems, developed in several European research projects. It is designed to complement AUTOSAR with descriptions at higher level of abstractions. Aspects covered by EAST-ADL include vehicle features, functions, requirements, variability, software components, hardware components and communication.

TADL2 (Timing Augmented Description Language) language concepts can be used in specific steps of the GMP methodology to describe timing information. TADL2 allows the specification of timing constraints that may express the following timing properties/requirements:

- Execution time (Worst-case, Best-case, Simulated, Measured)
- End-to-end Latency
- Sampling Rates
- Time Budget
- Response Time
- Communication Delay
- Slack
- Repetition pattern
- Synchronization
- ...

TADL2 base concepts are quite equivalent to those of AUTOSAR TIMEX presented in the following section.

2.5.2 Basic concepts of AUTOSAR TIMEX

According to [2], 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 been 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 must be contained in the timing augmented system model (such as the priority of a task, the activation behavior of an interrupt, the sender timing of a PDU and frame etc.). Such timing properties can be found all across AUTOSAR. For example the System Template provides means to configure and specify the timing behavior of the communication stack. Furthermore the execution time of ExecutableEntities can be specified. In addition, the overall specification must provide means to describe timing constraints. A timing constraint defines a restriction for the timing behavior of the system (e.g. bounding the maximum latency from sensor sampling to actuator access).

Timing constraints are added to the system model using the AUTOSAR Timing Extensions. Constraints, together with the result of timing analysis, are considered during the validation of a system's timing behavior, when a nominal/actual value comparison is performed.

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 purposes of the Timing Extensions are twofold. 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.

The remainder of this section describes the main concepts defined in the AUTOSAR Timing Extensions.

2.5.2.1 TIMEX Artifacts

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 each corresponding to one of the AUTOSAR views: Virtual Function Bus (VFB) Timing; Software Component (SW-C) Timing; System Timing; Basic Software (BSW) Module Timing; as well as ECU Timing. In particular, one uses these Events to specify the reading and writing of data from and to specific ports of SW-Cs, calling of services and receiving their responses (VFB Timing); sending and receiving data via networks and through communication stacks (System Timing); activating, starting and terminating executable entities (SW-C Timing and BSW Module Timing); and last but not least calling BSW services and receiving their responses (ECU Timing and BSW Module Timing).

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. 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 must occur within a given time interval (tolerance) to be said to occur simultaneous and synchronous respectively.

Additional Timing Constraints. In addition to the Timing Constraints that are imposed on Events and Event Chains, the AUTOSAR Timing Extensions provide Timing Constraints which are imposed on Executable Entities, namely the Execution Order Constraint and Execution Time Constraint.

2.6 Conclusions

To apply timing requirements decomposition in a comprehensive way several needs have to be fulfilled:

- All basic terms shall be unified. This means a term like WCRT has the same meaning and comprehensive understanding all over the industry.
- The structure of describing timing aspects shall be unified. For this need AUTOSAR TIMEX delivers an appropriate approach for the implementation driven perspective of AUTOSAR. It fails in higher levels of abstraction, because as soon as no AUTOSAR means like Software Components and Runnable exist, there is no meaning.
- The methodological approach for introducing timing analysis in a timing aware development process shall not be reduced to the definition of TIMEX artifacts referring to AUTOSAR system template artifacts. Additionally information of higher abstraction levels in earlier design phases shall be transferred to AUTOSAR modeling without losing exactness. This requires reference points valid within all phases and levels of abstraction.
- The methodology shall meet the needs of large scale organizations. This means the methodology shall be applicable tailor-made to the processes ruling a particular large scale organization.

The elements presented in this chapter allow a formal timing requirement decomposition described in the top level active steering example introduced in Chapter 1.

3 Timing Analysis for SW-Integration on ECU Level

This chapter outlines use-cases relevant for software integration into a single ECU with respect to timing issues. Network related aspects are covered by chapter 4 and have only an indirect impact on the timing on the ECU level. On the ECU level, the scheduling of tasks and interrupts together with the execution times of the various code fragments define the timing behavior of the overall software for this specific ECU. Depending on the scheduling and the execution times, given deadlines are met or missed. The use-cases in this chapter help to solve problems or tasks which are related to scheduling and/or execution times.

Although speaking of "ECU-level", it is important to bear in mind a single ECU can come with multiple processors each of which comes with its own scheduling. Even multiple cores on one processor are seen more and more often [12]. However, the principles in this chapter still remain valid and can be reflected on each "scheduling entity" (=core).

Typical terms used in this chapter are:

- Execution Time (e.g.: CET, BCET, WCET..), see section 2.1 and 6.4.
- CPU-Load , see section 6.4.
- Interrupt Load, see section 6.4.
- Response Time, see section 6.4.
- Latency, see section 6.4.

3.1 Example

In the introduction, a top-level-example for timing was given, see figure 1.2 on page 10. The ECU-level deals with a fragment of the top-level-example, namely the scheduling aspects and code execution aspects of the ECUs involved, see figure 3.1. The use-cases of this chapter will refer to this example.

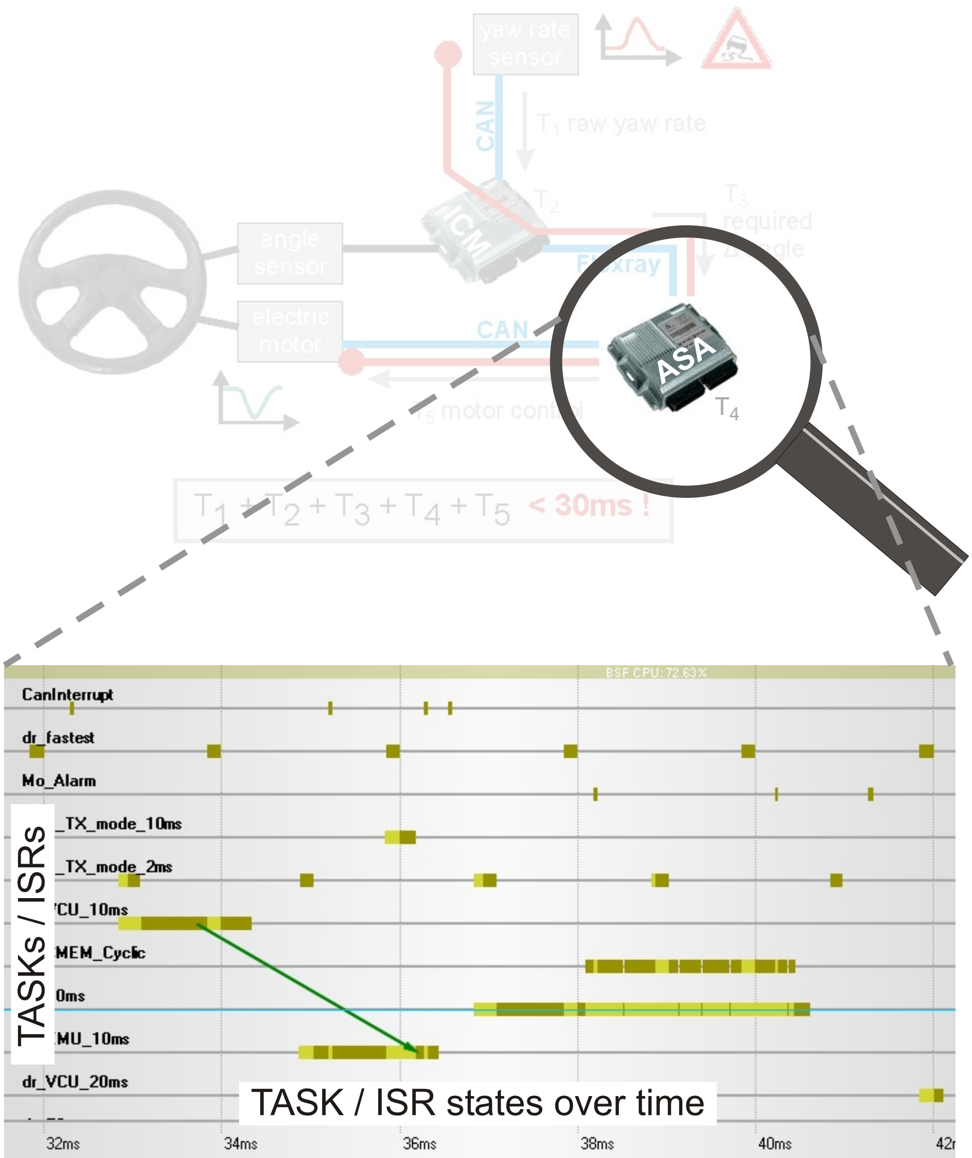

Figure 3.1: Focus of this chapter: scheduling and code execution time inside ECUs

3.2 Summary of Use-cases

This chapter describes the use-cases listed in Table 3.1. Figure 3.2 gives an overview.

Section	Use-case	Page
3.3	ECU use-case "Create Timing Model of the entire ECU"	35
3.4	ECU use-case "Collect Timing Information of a SW-C"	37
3.5	ECU use-case "Select an ECU Supplier"	39
3.6	ECU use-case "Validate Timing after SW-C integration"	40
3.7	ECU use-case "Validation of Timing"	41
3.8	ECU use-case "Debug Timing"	43
3.9	ECU use-case "Optimize Timing of an ECU"	45
3.10	ECU use-case "Optimize Scheduling"	47

Section	Use-case	Page
3.11	ECU use-case “Optimize Code”	49
3.12	ECU use-case “Verify Timing Model(s)”	50
3.13	ECU use-case “Compare Timing Properties”	52

Table 3.1: List of ECU specific use-cases

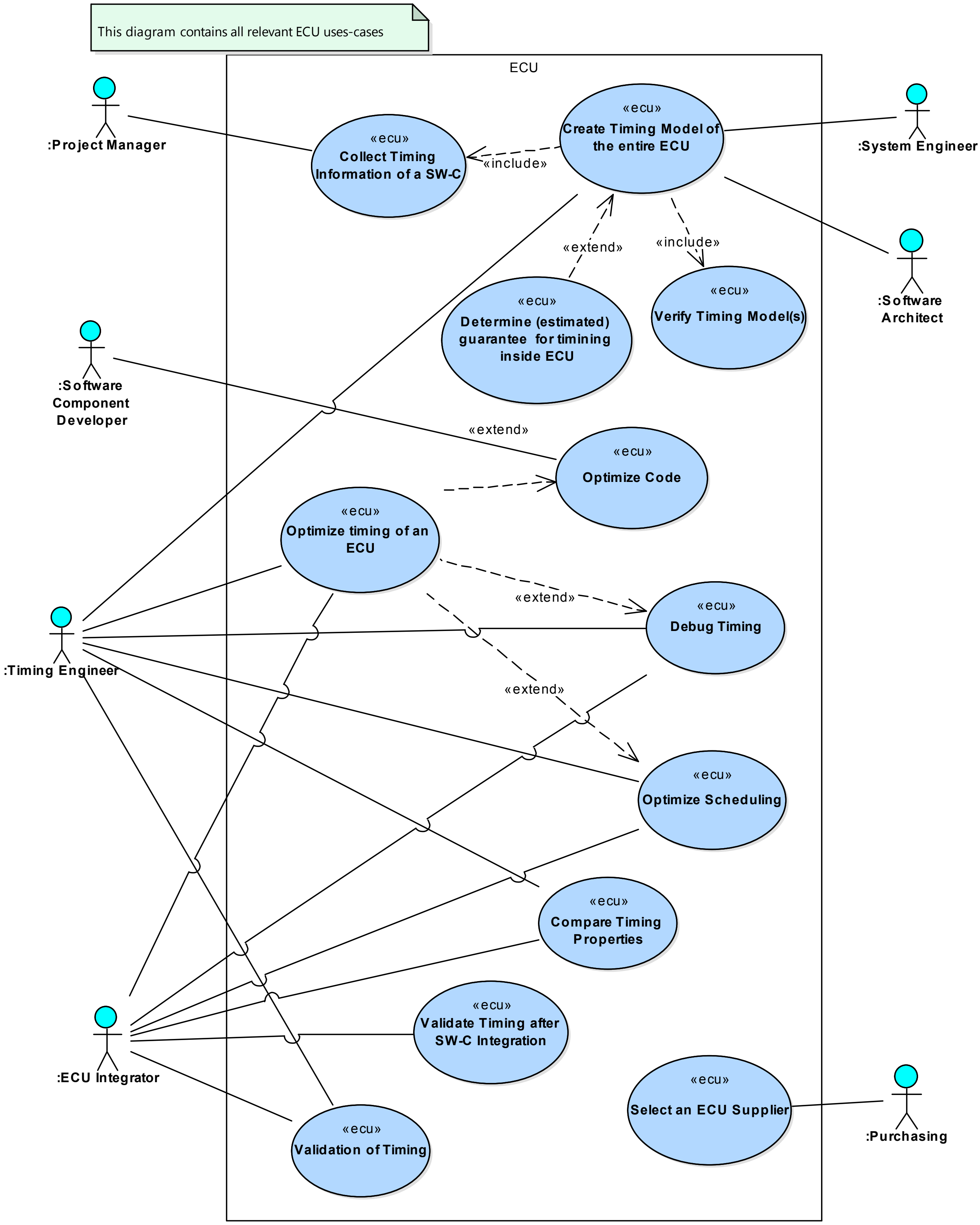

Figure 3.2: Use-case Diagram: Timing Analysis for ECU

3.2.1 Assumptions

If not otherwise stated the following assumptions hold true for all use-cases described in this chapter:

1. The ECU Extract for a specific ECU is available including the ECU Extract content for System Timing.
2. The VFB View (SW-C Template, hierarchy of SW-Cs) of all SW-Cs mapped onto the specific ECU is available.
3. SW-C descriptions are available
4. The interaction takes place between one OEM and one tier1 supplier
5. All SW-Cs including C source code and object files are available.
6. All required BSW Modules are available including C source code, o bject files and ECU configuration. (Only valid for use-case "ECU use-case "Validate Timing after SW-C integration"" described in section 3.6 on page 40.)
7. RTE can be generated
8. The contents of this chapter deal solely with the subject matter timing analysis. The assumption made is that any "system" subject to timing analysis is valid from the functional point of view.

Different phases/use-cases in the development of a vehicle system shall be considered, which are described in the following subsections.

3.3 ECU use-case "Create Timing Model of the entire ECU"

This section describes how to generate a timing model for a complete ECU. The difficulties to describe the use-case in a unique manner are justified by the fact that since the OEM and the Tier1 use different abstraction levels and semantics and work during different phases in the development process, their views on this use-case also differs. Nevertheless, some basic assumptions are valid for all levels of granularity and all development phases.

In the context of the example shown in figure 3.1 on page 32, the creation of timing model means to build up an abstract representation of the timing behavior of the ECU as the system under observation.

As a matter of fact, the creation of a timing model of the entire ECU is one of the important steps to gain a complete system understanding. All other use cases can be seen as somehow connected use-cases, since the existence of a timing model is a precondition in order to execute the steps in other use-cases.

A timing model of an ECU collects all timing data such as timing requirements, timing measurements and also timing relevant configuration data (such as RTE or BSW configuration) and can be used in other use-cases as well. Or in other words: Without a timing model it's difficult to handle the following use-cases.

Depending on the development phase, the timing model can be based mainly on assumptions and requirements (requirement timing model) or mainly based on measurements and existing configuration information. Ideally, both views are accessible in one model.

3.3.1 Characteristic Information

Goal In Context:	Collect all relevant timing information for a selected ECU
Brief Description:	Collect all relevant timing information for an ECU and create a timing model of the entire ECU
Scope:	ECU
Precondition:	Knowledge about basic functionality of the ECU and basic understanding about the functional requirements of the ECU and the application domain
Success End Condition:	Valid timing information
Failed End Condition:	n.a.
Actor(s):	ECU responsible person (This might be the project leader or the leading software developer.). Other actors: System Architect, Timing Expert, Project Manager (requests deliverables)

Table 3.2: Characteristic Information of ECU UC "Create Timing Model of the entire ECU"

3.3.2 Main Scenario

1. The ECU responsible person collects all available timing data for the specific ECU.
2. Checking of the collected data.
3. Add the retrieved timing data to timing model.
4. The use-case ends with ECU timing model. The timing information will be usable for further work.

3.3.3 Alternative Scenario

Due to the different levels of granularity and different phases different scenario extensions are possible. In concrete cases the timing expert must choose the matching scenario.

3.3.4 Related Information

Performance Target:	Timing model is usable for next integration level

Precondition:	
Frequency:	On request
Super-use-case:	n.a.
Sub-use-case(s):	n.a.

Table 3.3: Related Information for ECU UC "Create Timing Model of the entire ECU"

3.3.5 Related methods and properties

In short, all methods in section 6.5 and all properties in section 6.4 described in the chapter Properties and Methods for Timing Analysis are required to build a timing model of the entire ECU.

Furthermore a appropriate tool chain is required. Such a tool chain must be able to import and export the artifacts generated from different tools during the complete development cycle.

3.4 ECU use-case "Collect Timing Information of a SW-C"

In section 3.3, the creation of a timing model is described. Collecting timing information is required in order to build up a timing model. In this use-case, collecting timing information of a specific SW-C in described. For the example shown in figure 3.1 on page 32, this could mean getting information about a specific SW-C inside the ECU "ASA", e.g. its maximum execution time.

3.4.1 Characteristic Information

Goal In Context:	Collect all relevant timing information an selected SW-C
Brief Description:	Collect all relevant timing information for a SW-C
Scope:	SW-C for a specific target
Precondition:	Knowledge about basic functionality of the SW-C
Success End Condition:	Valid timing information
Failed End Condition:	n.a.
Actor(s):	SW-C responsible person (This might be the project leader or the leading software developer). Other actors: System Architect, Timing Expert, Project Manager (requests deliverables)

Table 3.4: Characteristic Information of ECU UC "Collect Timing Information of a SW Component"

3.4.2 Main Scenario

1. The use-case begins when the responsible SW-C person begins the collection of timing information which is usually triggered by an ECU-Integrator request.

2. The SW-C responsible person collects all available timing data for the specific SW-C and collects them in a timing model for SW-C scope.
    - Some estimation about previous and similar project, methods, see section 6.5
    - Runtime measurements on runnable level and below, methods, such as Processor-In-The-Loop Simulation (PIL) or Static Worst Case Execution Time Analysis , see section 6.5
    - Timing requirements for this SW-C based on functional requirements, for instance
        - Trigger events
        - Latencies
        - Jitters
        - Execution orders
        - Relations to safety-relevant requirements
    - Methods: see section 6.5
3. Add retrieved timing data to timing model.
4. The use-case ends with SW-C timing information. The timing information will be usable for SWC integration in the whole system.

3.4.3 Alternative #1 Scenario

At step #2 of the main scenario the sub-steps can be carried out in arbitrary order or might be skipped. The justification for skipping can be missing information at this specific phase in time.

3.4.4 Related Information

Performance Target:	Timing model is usable for next integration level
Precondition:	
Frequency:	On request
Super-use-case:	ECU UC Create Timing Model of the entire ECU see section 3.3
Sub-use-case(s):	n.a.

Table 3.5: Related Information for ECU UC "Collect Timing Information of a SW Component"

3.4.5 Related methods and properties

- Methods

  - Tracing, see section 6.5
  - PIL, see section 6.5
  - Scheduling Analysis, see section 6.5
  - Scheduling Simulation, see section 6.5
  - Static Worst Case Execution Time Analysis, see section 6.5
- Properties
  - Resource Load, see section 6.4
  - Execution Time, see section 6.4
  - Response Time, see section 6.4
  - Interrupt Load, see section 6.4

3.5 ECU use-case "Select an ECU Supplier"

The following use-cases do not match completely into the AUTOSAR methodology, but it is quite important anyway.

During the order phase for a new ECU some performance key indicators are used to evaluate metrics in order to decide the ECU design (e.g. processor type, memory, frequency). Additionally a supplier fulfilling the overall requirements must be selected.

At this stage timing experts from OEM and Tier1 must work together using some of the use-cases described in this document sketching a rough ECU architecture (regarding hardware and software) with the purpose to show the overall feasibility.

Typically an initial timing model will be available after this work was finished.

3.5.1 Related methods and properties

- Methods
  - Load , see section 6.5
- Properties
  - "Resource Load" , see section 6.4

3.6 ECU use-case "Validate Timing after SW-C integration"

An alternative title for this use-case is "Build up a system using existing, in the sense of most suitable from the timing perspective, SW-Cs". In this case the objective is that this system satisfies a given time constraint, for example "from sensor to actuator".

3.6.1 Characteristic Information

Goal In Context:	Validate timing after a SW-C has been replaced in an existing system
Brief Description:	In a given/already existing system one of the SW-Cs is replaced by a new version. The new version may consist of 1) the same number of RE as the previous version (but different implementations), or 2) a different number of REs than the previous version (fewer or more REs). From a timing analysis point of view it must be ensured that the new version still satisfies the given timing constraints. This requires to conduct 1) a response time analysis, and/or 2) a scheduling analysis which indicates that the given timing constraint is satisfied
Scope:	ECU
Precondition:	1. Definition of relevant timing constraints, which should be satisfied. 2. The SW-C/s has/have been mapped to a specific ECU which is the one subject to timing analysis. 3. The SW-C has been integrated from a structural point of view, which means that the "REs" have been mapped to the corresponding tasks and properly positioned within the tasks. Work product: ECU Configuration including OS configuration (Task Model and Task Parameters) and RTE configuration (RTE Event to Task Mapping) 4. All port interfaces are valid/compatible and all ports have been connected with the corresponding ports of the SW-Cs the SW-C subject to integration is exchanging data with. Work product: SW-C Description , System Description 5. System/ECU timing model of the system is available [StKu: Shall one state the granularity of the timing model here?] Work product: ECU Timing, System Timing
Success End Condition:	Timing analysis indicates that the given timing constraint defined in the precondition is satisfied (in all conditions). All relevant documentation has been updated.
Failed End Condition:	Neither response time analysis nor scheduling analysis indicate that the given timing constraint are satisfied. Timing measurements indicate that the timing constraint are violated more times than accepted by the customer.
Actor(s):	ECU Integrator. Other actors: Quality Manager, Timing Expert, [System] Project Manager, Architect

Table 3.6: Characteristic Information of ECU UC "Validate Timing after SW-C integration"

3.6.2 Main Scenario

1. The use-case begins when the actor receives the SW-C package together with a change order from the Change Control Board (CCB).
2. The actor performs the structural integration.
3. The actor replaces the timing model of the current version of the SW-C by the timing model of the new version of the SW-C [VFB or/and SW-C Timing] in the system's timing model.
4. In essence, this is simply referencing an event chain in the new SW-C's timing model from within the timing model of the system the new SW-C is "integrated".
5. The actor determines the differences between the previous and new SW-C. What does this tell the integrator?
6. The actor conducts the timing analysis
7. The actor reviews the result of the timing analysis and concludes that the given timing constraint is satisfied.
8. The actor marks the work products as valid, namely the timing model and the analysis report.

3.6.3 Related Information

Performance Target:	No performance key indicators identified for this use-case.
Precondition:	No constraints that may apply during the course of the use-case are identified.
Frequency:	Whenever a new version of a SW-C that is part of a system is becoming available and the decision has been taken to update the SW-C in the existing system.
Super-use-case:	n.a.
Sub-use-case(s):	n.a.

Table 3.7: Related Information for ECU UC "Validate Timing after SW-C integration"

3.7 ECU use-case "Validation of Timing"

3.7.1 Characteristic Information

Goal In Context:	Validate the timing of a defined system

Brief Description:	Validate the timing to ensure the schedulability of a system and that all given timing constraints are satisfied. The validation of the timing can be conducted via various timing analysis methodologies e.g. response time analysis and/or schedulability analysis depending on the nature of the timing constraints and/or expected level of confidence. If an implementation of the system exists the timing can be also measured and validated on basis of timing traces.
Scope:	ECU
Precondition:	At least one event chain with a stimulus specifying the receipt of a signal and a response specifying the transmission of a signal on a connected bus/network and a latency timing constraint is imposed on this event chain. The mapping of runnable entities to task is executed and RTE and OS configuration are available.
Success End Condition:	The schedulability analysis and the tracing confirm the given timing constraints are satisfied. This means timing is validated.
Failed End Condition:	The schedulability analysis and the tracing show the given timing constraints are violated.
Actor(s):	SW-C responsible person (This might be the project leader or the leading software developer.), Timing-Expert on ECU Level, ECU Integrator, [System] Project Manager, Architect

Table 3.8: Characteristic Information of ECU UC "Validation of timing"

3.7.2 Main Scenario

1. The use-case begins with the trigger event
2. The actor conducts the timing analysis with: 1) Response Time Analysis, 2) Scheduling Analysis, or 3) Measurement. → sub-use-cases
3. The actor reviews the result of the timing analysis and concludes that the given timing constraint is satisfied.
4. The actor marks the work products as valid, namely the timing model and the analysis report.

3.7.3 Related Information

Performance Target:	No performance key indicators identified for this use-case.
Precondition:	No constraints that may apply during the course of the use-case are identified.
Frequency:	Whenever the decision has been taken to validate the timing of the existing system.
Super-use-case:	n.a.
Sub-use-case(s):	n.a.

Table 3.9: Related Information for ECU UC "Validation of timing"

3.7.4 Related methods and properties

- Methods
  - Tracing, see section 6.5
  - PIL, see section 6.5
  - Scheduling Analysis, see section 6.5
  - Scheduling Simulation, see section 6.5
- Properties
  - Resource Load, see section 6.4
  - Execution Time, see section 6.4
  - Response-Time, see section 6.4
  - Interrupt-Load, see section 6.4

3.8 ECU use-case “Debug Timing”

Whenever an ECU shows sporadic system crashes, data inconsistencies or unexpected overload scenarios, delays or jitters, a timing issue could be the cause of the problem. Tracking the problem down with conventional debug methods can be very painful and time consuming. This is also true even if a certain problem is very obviously related to timing.
Before any problem can be *solved*, it has to be understood. This is what timing debugging is about: understanding a timing problem that is present on a real ECU. Once the problem is understood, the solution finding and solving follows, see section 3.9 “ECU use-case “Optimize Timing of an ECU”” on page 45. This use-case focuses on debugging the timing of a single ECU, e.g. the “ASA” shown in figure 3.1 on page 32.

3.8.1 Characteristic Information

Goal In Context:	Understand a (timing) problem and isolate the cause of the problem.
Brief Description:	Using dedicated timing debugging methods (see chapter 6), debug a problem and find out, if it is a timing problem. If so, track down the cause of the problem so that it is completely understood. This makes solving the problem possible in a next step.
Scope:	ECU
Precondition:	A running system
Success End Condition:	Problem understood, cause of the problem isolated. Artifacts: documentation describing the problem, e.g. schedule traces

Failed End Condition:	• problem not understood or • problem is not caused by faulty timing or • problem is not reproducible or based on the data of previous occurrences not sufficiently analyzable.
Actor:	Timing Expert on ECU-level, Software developer

Table 3.10: Characteristic Information of ECU UC "Debug Timing"

3.8.2 Main Scenario

1. The use-case begins when the main-actor is confronted with a timing problem or a problem that could be caused by timing on a real ECU.
2. If the problem is reproducible, use the real system for timing debugging. If not, try to make it reproducible. When this implies a modification of the system, reflect the changes in all later steps on the results (are all assumptions still true even with the modification?). If the problem is still not reproducible, try using the data of previous occurrences
3. Debugging using dedicated timing analysing methods see chapter 6
4. Isolate the problem
5. If the cause is trivial, fix it and test it. If not, hand the results over to a use-case finding a more sophisticated solution, e.g. ECU UC08 Optimize Timing for an series ECU.

3.8.3 Related Information

Performance Target:	n.a.
Precondition:	n.a.
Frequency:	Whenever a not trivial problem is detected in the ECU.
Super-use-case:	n.a.
Sub-use-case(s):	n.a.

Table 3.11: Related Information for ECU UC "Debug Timing"

3.8.4 Related methods and properties

- Methods
  - Extract Timing Traces , see section 6.5
  - Evaluate Timing Traces, see section 6.5
- Properties
  - "Resource Load", see section 6.4

– Interrupt Load, see section 6.4

3.9 ECU use-case "Optimize Timing of an ECU"

The main idea behind this use-case is to optimize the timing behavior of a working ECU. Sometimes the resource consumption is higher than expected or it is required to integrate further SW-C into the ECU.

3.9.1 Characteristic Information

Goal In Context:	Remove timing violations (optimize resource consumption, data consistency, reduce jitter,..) or minimize resource consumption
Brief Description:	Based on timing requirements, while taking all timing constraints into account the overall timing architecture for an ECU is optimized
Scope:	ECU
Precondition:	A running system and/or ideally a useful system description (timing-model)
Success End Condition:	Found a better solution which fulfill all timing and resources requirements (even with additional functionality if applicable). Artifacts: • (New Schedule • Updated Timing model) and/or • (optimized code • New memory layout • New code generator options • New compiler options)
Failed End Condition:	No solution found
Actor(s):	Timing expert on ECU-level, ECU integrator, Software developer, SW-Architect

Table 3.12: Characteristic Information of ECU UC "Optimize Timing for an series ECU"

3.9.2 Main Scenario

1. The use-case begins when the main-actor becomes aware of timing violations or the need to add more functionality into an already heavily loaded system.
2. Analyze the current system (validate the timing of the system, see ECU use-case "Debug Timing") and find hot-spots . These are situations in the schedule, where either timing requirements or resource consumption constraints are violated already or would be if more load was added.
3. Definition of the optimization goal(s) on a per hot-spot basis.
4. Analysis of available options in order to relax the hot-spots. These options can include modification of the scheduling configuration (including the runnable to

task mapping, the runnable sequence/order inside tasks, the allocation of task to different cores, the partitioning of tasks into smaller entities for load balancing, the change of priorities/offsets/recurrences of task) and/or code optimization (including the re-mapping of data to memory). For each option, continue with the corresponding sub-use-case.

5. The actor performs a trade-off analysis to weight the different possibilities for the optimization of the timing and its impact on the system
6. The actor decides for a modification and changes the timing-model/the code of the system
7. The actor validates the timing of the ECU
8. Verification against optimization goal

3.9.3 Related Information

Performance Target:	Different performance key indicators possible: • load balancing (distribute load on time axis, load balancing over different cores) • minimize systematically response times, jitters etc. • reduce number of preemptions (and thus reduce OS overhead) • reduce number of migration (and thus reduce migration overhead) • reduce resource consumption (inter-core communication, memory (buffer sizes), load) • reduce number of scheduling interrupts • reduce waiting times See also chapter metrics 6
Precondition:	n.a.
Frequency:	Whenever a timing violation is detected in the ECU, an additional functionality is added/expected or existing functionality is modified
Super-use-case:	n.a.
Sub-use-case(s):	ECU use-case “Optimize Scheduling” and ECU use-case “Optimize Code”

Table 3.13: Related Information for ECU UC “Optimize Timing for an series ECU”

3.9.4 Related methods and properties

- Methods
  - Tracing, see section 6.5
  - PIL, see section 6.5

  - Scheduling Analysis, see section 6.5
  - Scheduling Simulation, see section 6.5
- Properties
  - Resource Load, see section 6.4
  - Execution-TIME, see section 6.4
  - Response-TIME, see section 6.4
  - Interrupt-Load, see section 6.4

3.10 ECU use-case "Optimize Scheduling"

The main idea behind this use-case is the optimization of an existing schedule of a working ECU with a defined goal such as "remove local overload" or "reduce response time of task xyz".

3.10.1 Characteristic Information

Goal In Context:	Fulfill predefined optimization goal
Brief Description:	Find a modified schedule configuration which fulfills the goal without causing new timing violations or violates resource constraints
Scope:	ECU
Precondition:	A running system and/or ideally a useful system description (timing-model)
Success End Condition:	Found a modified schedule configuration which fulfills the goal without causing new timing violations. Artifacts: • New Schedule, better than the original schedules with respect to a specific metric, see chapter 6.4 • Updated Timing model
Failed End Condition:	No solution found
Actor(s):	Timing expert on ECU-level, Software developer, SW-Architect

Table 3.14: Characteristic Information of ECU UC "Optimize Scheduling"

3.10.2 Main Scenario

1. The use-case begins when the main-actor is confronted with a certain optimization goal regarding the scheduling
2. Analysis of available options e.g. modification of the runnable to task mapping, the runnable sequence/order inside tasks, the allocation of task to different cores, the partitioning of tasks into smaller entities for load balancing, the change of priorities/offsets/recurrences of task

3. The actor performs a trade-off analysis to weight the different possibilities for the optimization of the schedule and its impact on the system
4. The actor decides for a solution and modifies the timing-model/code of the system.
5. The actor validates the timing of the ECU by conducting response time analysis, scheduling analysis or measurements
6. Verification against optimization goal

3.10.3 Related Information

Performance Target:	Different performance key indicators possible: • load balancing (distribute load on time axis, load balancing over different cores) • minimize systematically response times, jitters etc. • reduce number of preemptions (and thus reduce OS overhead) • reduce number of migration (and thus reduce migration overhead) • reduce resource consumption (inter-core communication, memory (buffer sizes), load) • reduce number of scheduling interrupts • reduce waiting times See also 6.4
Precondition:	n.a.
Frequency:	Whenever a timing violation is detected in the ECU, an additional functionality is added/expected or existing functionality is modified
Super-use-case:	ECU use-case "Optimize Timing of an ECU"
Sub-use-case(s):	• Conduct Response Timing Analysis • Conduct Scheduling Analysis • Conduct Measurements

Table 3.15: Related Information for ECU UC "Optimize Scheduling"

3.10.4 Related methods and properties

- Methods
  - Tracing, see section 6.5
  - PILTracing, see section 6.5
  - Scheduling AnalysisTracing, see section 6.5
  - Scheduling SimulationTracing, see section 6.5

- Properties
  - Resource Load, see section 6.4
  - Execution Time, see section 6.4
  - Response Time, see section 6.4

3.11 ECU use-case "Optimize Code"

Since the code and the deployment of code has a huge impact on timing, different optimization activities can be performed. The scope of the optimization can be different (memory, run-time, safety, re-usability, easy to understand, etc.), however in the scope of this document, the optimization scope is limited to timing effects. But it has to take into account, that such timing optimization influence other aspects of the system, such as memory and reusability and that such optimization is constrained by safety or security aspects

3.11.1 Characteristic Information

Goal In Context:	Optimize the code with respect to timing. Typically: minimize the WCET, the average execution time or both.
Brief Description:	Based on timing requirements optimize the overall timing architecture for an ECU
Scope:	ECU
Precondition:	Code available (ideally compilable, linkable and executable on the target platform)
Success End Condition:	Found a better code which respect to timing. Possible artifacts: • Optimized code • New memory layout • New code generator options • New compiler options
Failed End Condition:	No solution found
Actor(s):	Software developer, Timing expert, SW-architect

Table 3.16: Characteristic Information of ECU UC "Optimize Code"

3.11.2 Main Scenario

1. The use-case begins when the main-actor determines to optimize a certain code fragment (a task, an interrupt, a runnable, a function or part of a function)
2. Definition optimization goal
3. Analysis of available options
4. Modification

5. Test
6. Verification against optimization goal

3.11.3 Related Information

Performance Target:	Methods:	• measurement/tracing • static code analysis • review (including output generated by compiler) • mapping of symbols to memory • ...
Precondition:	n.a.	
Frequency:	Whenever a timing optimization in the ECU is needed.	
Super-use-case:	ECU use-case "Optimize Timing of an ECU"	
Sub-use-case(s):	n.a.	

Table 3.17: Related Information for ECU UC "Optimize Code"

3.11.4 Related methods and properties

- Methods
  - Tracing, see section 6.5
  - PIL, see section 6.5
- Properties
  - Resource Load, see section 6.4
  - Execution-TIME, see section 6.4
  - Response-TIME, see section 6.4
  - Interrupt-Load, see section 6.4
  - Code-Metrics, see section 6.4

3.12 ECU use-case "Verify Timing Model(s)"

Any model based design or verification process must undergo a model check to make sure, the model represents reality with respect to the relevant properties.
Example 1: a perfect static code analysis tool for WCET calculation on code level can easily produce wrong results (false positives!) when not configured correctly.
Example 2: a perfect static scheduling analysis tool for WCRT calculation on ECU level can easily produce wrong results (false positives!) when the real ECU suffers an operating system bug.

In both examples, the models and the model based algorithms might be absolutely correct but still they produce false positive results. In other words: a software might pass model based verification and still show drastic timing defects.

3.12.1 Characteristic Information

Goal In Context:	Verify that a model based method reflects the real system with respect to the relevant properties.
Brief Description:	Based on methods which profile the behavior of the real ECU/the real code, the results of the model based approach gets cross-checked.
Scope:	ECU or code
Precondition:	A running system or executable code or code fragments
Success End Condition:	A comparison of measured/traced timing metrics with the timing metrics provided by the model based approach shows the model based approach generates plausible results.
Failed End Condition:	Measurement/tracing uncovered timing behavior beyond the worst case results proclaimed by the model based approach.
Actor(s):	Timing expert on ECU-level or code-level, Software developer

Table 3.18: Characteristic Information of ECU UC "Verify Timing Model(s)"

3.12.2 Main Scenario

1. The use-case begins when model based approaches are or become available and the real system is or becomes available. Typically, this will happen in either of two set-ups: for an existing system, model based approaches are added or model based approaches are used in an early development phase and the (real) system becomes available.
2. Produce the metrics with the model based approach, e.g. WCET or WCRT
3. Measure or trace the comparable metrics with the real system, e.g. max. CET or max. RT
4. Compare the model based results with the measured or traced results. All the "worst case" results produced by the model based approaches must be "worse" than the observed results. For comparing timing properties, see also ECU use-case "Compare Timing Properties".
5. The use-case ends. However, this approach cannot guarantee the correctness of the model, because the test vectors the measurements were based on were not covering a case which would have uncovered a problem with the model. But at least it provides an additional and very important check.

3.12.3 Related Information

Performance Target:	n.a.
Precondition:	n.a.
Frequency:	Once model based approaches are added while the real system is available or when the real system becomes available and model based approaches are used already. Afterwards, the cross check should be done again at least for major software releases of the system's software.
Super-use-case:	n.a.
Sub-use-case(s):	n.a.

Table 3.19: Related Information for ECU UC "Verify Timing Model(s)"

3.12.4 Related methods and properties

- Methods
  - Tracing, see section 6.5
  - PIL, see section 6.5
- Properties
  - n.a.

3.13 ECU use-case "Compare Timing Properties"

Compare two sets of timing properties. These might be

- obtained by different timing analysis techniques, e.g. simulation and measurement,
- obtained by analyzing different versions of the ECU software or
- related to different constraint types, e.g. "requirement" and "guarantee". See chapter 6 "Properties and Methods for Timing Analysis" on page 76 for details.

3.13.1 Characteristic Information

Goal In Context:	Compare timing properties obtained by means of response time analysis and/or scheduling analysis with measured time values.
Brief Description:	Unlike measurement the validation of timing can be conducted early in the design phase by response time analysis and/or scheduling analysis based on legacy software components and/or budgets. In order to validate a) the execution times of individual software components and b) the resulting timing properties like response times gathered early in the design phase the timing properties have to be compared with measured time values.
Scope:	ECU

Precondition:	A timing model of the ECU exists (See ECU use-case "Create Timing Model of the entire ECU") and an implementation of the system exists
Success End Condition:	The timing properties of the timing model are compared to measured values • the elements of the timing model which differ from the measured values are updated or complemented with the measured values • a new timing model with the measured values is generated
Failed End Condition:	The timing properties cannot be compared
Actor(s):	Timing expert on ECU-level, ECU integrator, Quality, Timing Expert/Analyst, [System] Project Manager, Architect

Table 3.20: Characteristic Information of ECU UC "Compare Timing Properties"

3.13.2 Main Scenario

1. 1. The use-case begins when the actor decides to compare timing properties.
2. The actor measures the timing properties of the system
3. The actor compares timing properties obtained by response time analysis and/or scheduling analysis with measured values
   - The actor generates a timing model out of the measured values and compares the timing models with each other
   - The actor compares selected elements of the timing model with the measured values
4. The actor reviews the result of the comparison and chooses one of the following actions
   - the elements of the timing model which differ from the measured values are updated or complemented with the measured values
   - a new timing model with the measured values is generated

   See also section 6.5]

3.13.3 Related Information

Performance Target:	No performance key indicators identified for this use-case.
Precondition:	Target code exists which can be measured
Frequency:	Whenever the decision has been taken to compare timing properties
Super-use-case:	n.a.

Sub-use-case(s):	• Conduct Response Timing Analysis • Conduct Scheduling Analysis (e.g. find a schedule for a system) • Conduct Measurements • Create Timing Model

Table 3.21: Related Information for ECU UC “Compare Timing Properties”

3.13.4 Related methods and properties

- Methods
  - Tracing, see section 6.5
  - PIL, see section 6.5
  - Scheduling Analysis, see section 6.5
  - Scheduling Simulation, see section 6.5
- Properties
  - n.a.

4 Timing Analysis for Networks

This chapter outlines timing use-cases related to automotive network communication. The ECU related timing aspects covered by Chapter 3 may have direct or indirect impact on the network timing.

In an automotive communication network the timing behavior is mainly described by the communication matrix, which contains the communication frames with the protocol and timing specific parameters. (e.g. payload, IDs, frame triggering parameters).

Depending on the amount of traffic to be transmitted on the network and the communication protocol, a network may or may not satisfy given timing requirements, such as maximum latency of frames or a given bus load threshold. In general, the network architect must define the parameters such that the timing requirements are fulfilled.

The use-cases described in this chapter present problems and solutions related to the design of communication networks.

Typical terms used in this chapter are:

- Load, see section 4.3, 4.4, 4.5 and 6.4
- Latency, see section 6.4
- Response Time, see section 6.4

4.1 Example

In Chapter 1, an example of a system with end-to-end timing requirements was given, see figure 1.2 on page 10. On the network, the end-to-end timing requirements are influenced by the network configuration and the scheduling of the network components, mainly buses and gateways, see figure 4.1. The use-cases in this chapter refer to this example.

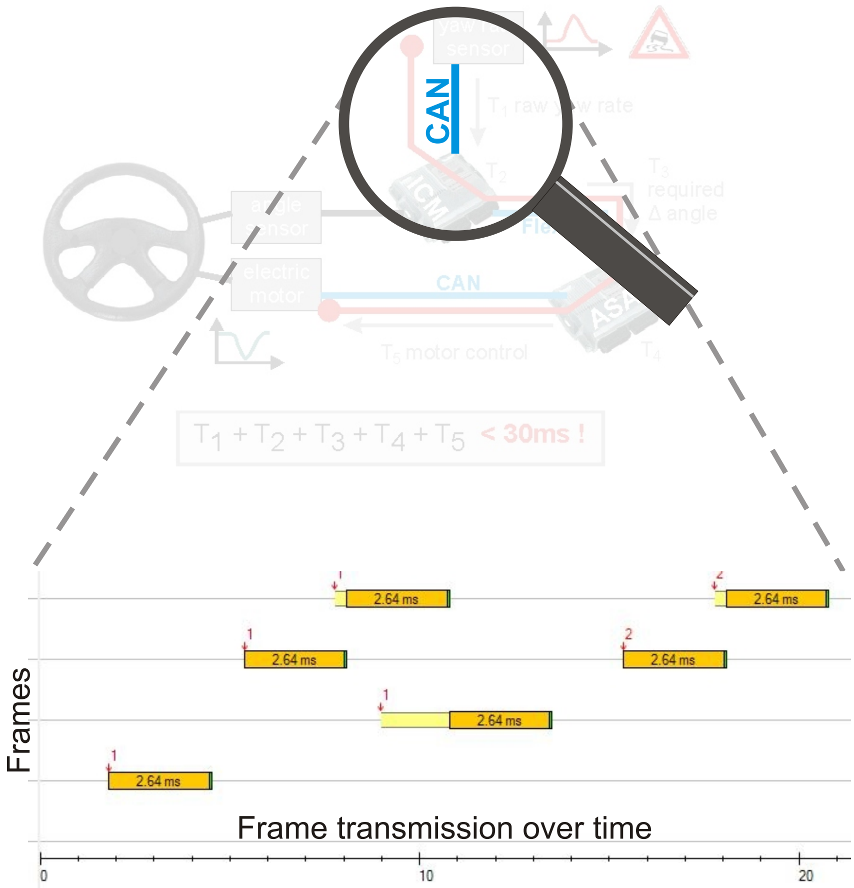

Figure 4.1: Focus of this chapter: bus timing in networks

4.2 Summary of Use-cases

This chapter describes the use-cases listed in Table 4.1.

Section	Use-case	Page
4.3	NW use-case “Integration of new communication”	57
4.4	NW use-case “Design and configuration of a new network”	60
4.5	NW use-case “Remapping of an existing communication link”	63

Table 4.1: List of network specific use-cases

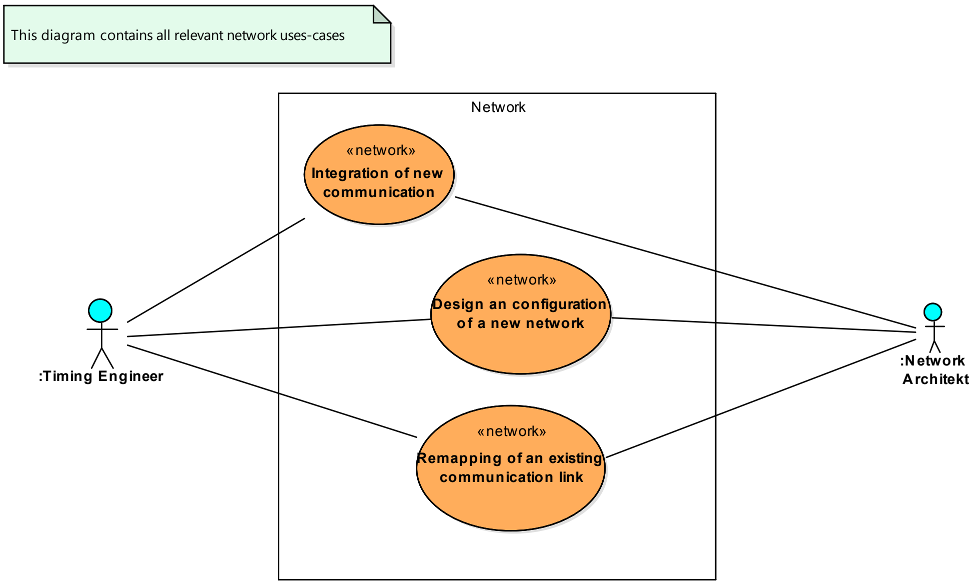

Figure 4.2: Use-case Diagram: Timing Analysis for Network

4.3 NW use-case “Integration of new communication”

This use-case focuses on integrating new communication into an existing automotive network.

Goal In Context:	Feasible integration of new communication into an existing networked architecture.
Brief Description:	Considering an E/E automotive architecture consisting of several ECUs connected via buses, it is required to integrate additional communication into the network, such that the legacy communication and the new communication fulfil the timing requirements. For example, the new communication is additional sensor data transmitted over the network from the *yaw rate sensor* to the *ASA*-ECU, as shown in figure 1.2 on page 10. The maximum latency of the new sensor data on the network bus must not exceed 100ms. The buses on which the new communication must be integrated may implement different communication protocols (e.g. CAN, LIN, Flexray, etc.). The communication on each bus is specified by a communication matrix containing the PDUs/frames with their protocol specific parameters and the communication behavior (timing parameters).
Scope:	System, System Timing

Precondition:	For the new communication following properties are defined: • The size of the communication signals (SW-C Template / GenericStructureTemplate). • The transmitter and receiver nodes / system mapping • The PDU/Frame timing/triggering • Required bandwidth Additionally, for the communication on the network a set of timing requirements is known: • Maximum bus load on each bus • Maximum latency for each PDU/Frame Furthermore, a specification of the communication paradigm for the existing bus controllers is available, e.g. the CAN controller sends CAN-frames with different identifiers via a queue (priority ordered or FIFO), while different instances of the same frame are sending via a register (always send the newest frame instance). It is assumed that the current network configuration satisfies the timing requirements.
Success End Condition:	The new communication was completely defined and the timing requirements are satisfied.
Failed End Condition:	The new communication cannot be defined without violating at least one timing requirement.
Actor(s):	System Architect / Network Architect

Table 4.2: Characteristic Information of NW UC "Integration of new communication"

4.3.1 Main Scenario

For the sake of clarity following notations are used: the existing E/E architecture consists of several ECUs (ECU1, ECU2, a.s.o) connected via multiple communication buses (denoted Bus1, Bus2, a.s.o) and one or multiple gateways. The new communication that has to be integrated into the existing network is assigned to a distributed function denoted F. F consists of multiple Software Components (SW-Cs) which are mapped on one or multiple ECUs. Each SW-C has it own communication interfaces through which it sends or receive information, i.e. communication signals packet in PDUs/frames.

1. The network architect maps the new communication to the existing PDUs/frames according to the timing parameters defined in the specification of F and the sender-receiver relation between the SW-Cs of F.
2. Depending on the links between the SW-Cs of F it might be necessary to additionally route PDUs/frames between different buses within the network. This happens when the SW-Cs of F are mapped to ECUs that are connected to different buses, e.g. on ECU1 on Bus1 and on ECU2 on Bus2.
3. Analysis 1: The bus load analysis describes the average use of the bus bandwidth. The bus load analysis has to consider the additional traffic generated by

the new communication. The bus load analysis must be applied to each bus affected by the new communication. The bus load analysis requires the data size and the average timing of the PDUs/frames. The output of the analysis is the timing property GENERIC PROPERTY Load. The bus load property is used to initially approve the traffic on each communication bus. The present value of the timing property load obtained for every single bus is compared to the maximum acceptable load on that bus. For typical requirements for the bus load see section 4.3.3.

4. Analysis 2: In order to validate the network after integrating the new communication, latency requirements have to be also verified on each bus for all PDUs/frames of the legacy and of the new traffic. The latency analysis computes the timing properties of the PDUs/frames under the resource sharing protocol. The results of the analysis are timing properties such as response times of the PDUs/frames GENERIC PROPERTY Latency (or SPECIFIC PROPERTY Response Time (CAN) specific for CAN buses), the jitter of the PDUs/frames, or the blocking times due to arbitration. The values of the timing properties are compared to the defined requirements. For typical requirements of the PDU/frame response times see section 4.3.3.

5. Analysis 3: In case that the PDUs/frames associated to the new communication are routed by one or more gateways, the routing times are relevant for the end-to-end timing. The routing time analysis of the routed PDUs/frames provides the delay values due to routing engine. These usually consist of buffering delay and arbitration delay. The results of the routing time analysis are the routing response times, the blocking times due to buffering and arbitration, or the memory requirements for buffering. The values obtained for these properties are compared to the defined requirements. For typical requirements of the routing times see section 4.3.3.

4.3.2 Alternative Scenario

At step #1 of the main scenario, if the new communication exceeds the size of the unused space in the existing PDUs/frames, new PDUs/frames are defined according to the timing parameters of the signals. The impact of the new traffic on the existing communication must be minimized. The methodology continues with Step 2 in the Main Scenario.

4.3.3 Performance/Timing Requirements

The maximum load on each bus shall not exceed a certain bound, for example 60%.

For each frame/PDU, the worst-case response time shall not exceed a certain bound, for example given by the timing requirements. Typically, the cycle time of the frame is used as a bound on the worst-case response time.

Routing times in gateways have to be short. Typically, for each frame/PDU the routing time shall not exceed for example 10% of the cycle time of the frame.

4.3.4 Related Information

Performance Target:	Bus load, Response Times, Routing Times
Precondition:	n.a.
Frequency:	Regular
Super-use-case:	n.a.
Sub-use-case(s):	n.a.

Table 4.3: Related Information of NW UC "Integration of new communication"

4.4 NW use-case "Design and configuration of a new network"

Goal In Context:	Design and feasible integration of a (domain specific) network into existing automotive platform architecture. Possible variants: • New design of the (on-board network) (total automotive network); • Replacement of an existing partial network by a new partial network under use of unaltered legacy ECUs (beside the network connectors). This network is connected to the residual on-board network by a gateway.
Brief Description:	Regarding an existing E/E automotive architecture consisting of several ECUs connected via several legacy networks, it is required to design and to integrate a new designed network (e.g. our active steering example, compare see figure 1.2). The new designed network shall be connected to the residual on-board network via a gateway (for instance to the body or the infotainment domain). Therefore the intra-communication within the new network and the inter-communication between different networks have to considered. Further, this new network shall be stable extensible in a-priori predictable way, i.e. it shall be possible to analyze the network with respect to all present and future timing requirements. The new network implements a communication protocols (e.g. CAN, LIN, Flexray, etc.) and possesses sufficient bandwidth to cover all communication requirements. The communication on the network is specified by a communication matrix containing the PDUs/frames/packages with their protocol specific parameters and the communication behavior (timing parameters).
Scope:	System, System Timing

Precondition:	For the new communication following properties are defined: • The size of the communication signals (SW-C Template / GenericStructureTemplate). • The transmitter and receiver nodes / system mapping • The PDU/frame/package timing/triggering • Required bandwidth • The residual on-board network including gateways and communication matrix Additionally, a set of timing requirements is defined for the communication on the network: • Maximum load on each network • Maximum latency for each PDU/frame/package Furthermore, a specification of the communication paradigm for the existing network controllers is defined, e.g. the CAN controller sends PDUs/frames with different identifiers via a queue (priority ordered or FIFO), while different instances of the same PDU/frame are sending via a register (always send the newest PDU instance). It is assumed that the current (residual) on-board network configuration satisfies the timing requirements.
Success End Condition:	The communication on the new (partial) network was completely defined and the timing requirements of the on-board network are satisfied.
Failed End Condition:	The new communication cannot be defined without violating at least one timing requirement of the on-board network.
Actor(s):	System Architect / Network Architect

Table 4.4: Characteristic Information of NW UC "Design and configuration of a new network"

4.4.1 Main Scenario

1. The network architect chooses an appropriate network technology to fulfil the communication requirements of the new functions. The consequences for the residual system have to be considered because many ECUs should not be altered if possible.
2. The network architect defines and /or designs the connection point (s) to the residual on-board network (via gateways).
3. The system architect connects the ECUs to the new network and partitions the functions onto these ECUs.
4. The network architect maps the new traffic according to the timing information and the transmitter/receiver relation.
5. Depending on the sender/receiver relation it might be necessary to additionally route PDUs/frame on several networks and gateways.
6. Analysis 1: Load analysis determines the average and the maximum use of the network bandwidth and the input buffer of the ECUs. The load analysis

must consider the total traffic on the new partial network and on the legacy on-board network as well. The analysis requires the data size and the timing of the PDUs/frames. The output of the analysis is the timing property load GENERIC PROPERTY Load. The timing property load is used to initially approve the chosen function mapping and architecture and if the new infrastructure is sufficient to cover the communication requirements in general. The present value of the timing property load for every single network is compared to the maximum acceptable load for this network.

7. Analysis 2: A detailed latency analysis of all PDUs/frames/packages and every communication relations on the networks is necessary. The method yields the timing properties response time GENERIC PROPERTY Latency, jitter, blocking time, etc. Every communication relation has to fulfil its corresponding latency requirement.

8. Analysis 3: In order to consider the ECU influence and the total communication the event chain / the routing time analysis of the PDUs/frames/package has to be considered. This leads to the new timing properties: routing response time SPECIFIC PROPERTY Response Time (CAN), blocking time, buffer requirements GENERIC PROPERTY Load.

9. Optimization of the design of the new network subject to the requirement to reduce resource needs, to increase system stability and robustness and to allow easily future extensions.

4.4.2 Performance/Timing Requirements

The maximum load on each bus shall not exceed a certain bound, for example 60%.

For each frame/PDU, the worst-case response time shall not exceed a certain bound, for example given by the timing requirements. Typically, the cycle time of the frame is used as a bound on the worst-case response time.

Routing times in gateways have to be short. Typically, for each frame/PDU the routing time shall not exceed 10% of the cycle time of the frame.

4.4.3 Related Information

Performance Target:	Bus load, Response Times, Routing Times
Precondition:	n.a.
Frequency:	Regular
Super-use-case:	n.a.
Sub-use-case(s):	n.a.

Table 4.5: Related Information of NW UC “Design and configuration of a new network”

4.5 NW use-case “Remapping of an existing communication link”

This use-case focuses on remapping an existing communication link within an existing network.

Goal In Context:	Validate the communication on the network after reconsidering the mapping of an existing communication link.
Brief Description:	Assuming an E/E automotive architecture that contains ECUs connected via one or more buses, it is required to remap an existing communication link to a new resource within the network (e.g.mapping the motor control signal from CAN to FlexRay assuming that the electric motor is directly connected to FlexRay, see figure 1.2). The buses within the network may implement different communication protocols (e.g. CAN, LIN, Flexray). The communication on each bus is specified by a communication matrix containing the PDUs/frames with their protocol specific parameters and the communication behavior (timing parameters, e.g. 10ms maximum latency for the motor control signal).
Scope:	System, System Timing
Precondition:	The signals describing the communication link is known and included in the communication matrix. Additionally, for the communication on the network is defined a set of timing requirements: • Maximum bus load on each bus • Maximum latency for each communication frame. Furthermore, the specification of the communication paradigm for the existing bus controllers is available. For example, the CAN controller sends CAN message frames with different identifiers via a queue (priority ordered or FIFO), while different instances of the same frame are sent via a register (always send the newest instance of the frame). It is assumed that the current network configuration satisfies the timing requirements.
Success End Condition:	The communication on the network after function remapping fulfils the timing requirements. The communication matrix needs to be updated.
Failed End Condition:	The communication on the network after function remapping cannot be defined without violating at least one timing requirement.
Actor(s):	System Architect / Network Architect

Table 4.6: Characteristic Information of NW UC “Remapping of an existing communication link”

4.5.1 Main Scenario

For the sake of clarity following notations are used: the communication link to be remapped is currently assigned to Bus1. The resource that will host the communication link after remapping is denoted Bus2.

1. The network architect identifies the PDUs/frames on Bus1 assigned to the communication link. These must be transmitted on Bus2 after remapping the communication link.

2. The PDUs/frames assigned to the communication link and additionally required by other links on Bus1 must be routed on Bus2 after remapping the communica-

tion link to Bus2. Otherwise, in case that these PDU/frame are not required by other nodes at Bus1, one may decide to remove them from Bus1.

3. The PDU/frames moved or copied to Bus2 should preserve the parameters of the communication protocol defined for Bus1, in order to ensure the function compatibility with the different architecture variants.
4. PDUs/frames required by the communication link at Bus2 and that are not originating at Bus2 need to be routed/transmitted to Bus2.
5. Analysis 1: The bus load analysis describes the average use of the bus bandwidth. The analysis has to consider the additional traffic on Bus2 after remapping the communication link to Bus2. The analysis requires the data size and the average timing of the PDUs. The output of the analysis is the static bus load GENERIC PROPERTY Load. The bus load property is used to initially approve the traffic on Bus2. Optionally, one can carry out bus load analysis on Bus1 to determine the freed performance slack after remapping the communication link to Bus2. The value of the timing property load obtained for every single bus is compared to the maximum acceptable load on that bus. For typical requirements for the bus load see section 4.5.2.
6. Analysis 2: In order to validate the communication on the network after remapping the communication link to Bus2, the latency requirements of the PDUs/frames on Bus2 must be verified. The latency analysis of the PDUs/frames computes the timing properties of the PDUs/frames under the resource sharing protocol. The results of the analysis are timing properties such as the response times of the PDUs/frames GENERIC PROPERTY Latency (or GENERIC PROPERTY Response Time specific for CAN buses), the jitter of the PDUs/frames, or the blocking times due to arbitration. The values of the timing properties are compared to the specified requirements. For typical requirements of the PDU/frame response times see section 4.5.2.
7. Analysis 3: In case that the PDUs/frames required at Bus2 are routed by one or more gateways, the routing times are relevant for the end-to-end timing. The routing time analysis of the routed PDUs/frames provides the delay values due to routing engine. These usually consist of buffering delay and arbitration delay. The results of the routing time analysis are the routing response times, the blocking times due to buffering and arbitration, or the memory requirements for buffering. The values obtained for these properties are compared to the specified requirements. For typical requirements of routing times see section 4.5.2.

4.5.2 Performance/Timing Requirements

The maximum load on each bus shall not exceed a certain bound, for example 60%.

For each frame/PDU, the worst-case response time shall not exceed a certain bound, for example given by the timing requirements. Typically, the cycle time of the frame is used as a bound on the worst-case response time.

Routing times in gateways have to be short. Typically, for each frame/PDU the routing time shall not exceed for example 10% of the cycle time of the frame.

4.5.3 Related Information

Performance Target:	Bus load, Response Times, Routing Times
Precondition:	n.a.
Frequency:	Regular
Super-use-case:	n.a.
Sub-use-case(s):	n.a.

Table 4.7: Related Information of NW UC “Remapping of an existing communication link”

5 End-to-End Timing Analysis for Distributed Functions

This chapter introduces use-cases to reason about the end-to-end timing of distributed functions. As a distributed function, we consider

- a function that executes locally but requires data from sensors or functions communicated over the network. In this case there exists at least an assumption about the maximum age of the data or
- a function that consists of several computation steps that are performed on different ECUs, connected via dedicated or shared buses. In this case event chains often exist with overall latency or periodicity constraints.

Most automotive functions today are distributed functions.

5.1 Relation to other chapters

This chapter is related to the other parts of this document as follows:

- Chapter 2 introduces the terminology of a function, what a function is, and informally discusses how an end-to-end timing requirement can be decomposed into timing requirements for each involved resource.
- Chapter 3 and 4 discuss use cases related to timing analysis on individual resources.
- Chapter 6 defines timing properties and how to derive these for individual resources and schedulables. The relevant timing properties (in particular the load of a resource and the latency of a schedulable) are introduced in Definition and Classification of Timing Properties.

Furthermore, the chapter (and the other chapters) is related to the AUTOSAR Timing Extensions [2], as it allows to derive the guarantees or assert the constraints specified therein.

5.2 Summary of Use-cases

This chapter describes the following use-cases listed in Table 5.1.

Section	Use-case	Page
5.3	E2E use-case "Derive per-hop time budgets from End-to-End timing requirements"	67
5.4	E2E use-case "Deriving timing requirements from the timing assessment of an existing implementation"	68
5.5	E2E use-case "Specify Timing Requirements for Signals/Parameters"	70
5.6	E2E use-case "Assert timing requirements against guarantees"	72

Section	Use-case	Page
5.7	E2E use-case "Trace-based timing assessment of a distributed implementation"	74

Table 5.1: List of use-cases related to end-to-end timing

5.3 E2E use-case "Derive per-hop time budgets from End-to-End timing requirements"

This use-case specifies the work flow for function(s) owners on how to decompose end-to-end timing requirements in order to derive and specify per hop time budgets (e.g. define local time budgets for functions(s) for each execution node and communication bus they are distributed).

This use-case becomes relevant when a customer functionality is identified in the specification. This functionality is decomposed into a set of functions that have to be integrated into an existing E/E network. An end-to-end timing requirement is identified for these functions and time budgets for each segment of the end-to-end chain has to be derived.

Goal In Context:	Specify time budgets for each ECU and communication network on which function(s) participating in an end-to-end timing requirement are distributed.
Brief Description:	This use case requires that an end-to-end timing chain specifying a function or a set of functions with an end-to-end timing requirement is specified. Moreover, functions participating to the end-to-end timing chain are decomposed in sub-functions that are allocated to ECUs interconnected with communication networks. Based on end-to-end timing requirements, this use case derives time budgets for each sub-function (segment of the end-to-end timing chain) allocated to an ECU.
Scope:	E2E
Precondition:	An end-to-end timing chain (with a function decomposition) with an end-to-end timing requirement is available.
Success End Condition:	A time budget has been found for each sub-function (segment) of the end-to-end timing chain and the sum of the budgets are not exceeding the end-to-end timing requirement.
Failed End Condition:	Some time budgets could not be derived or the sum of the found time budgets exceeds the end-to-end timing requirement.
Actor(s):	Function Owner

Table 5.2: Characteristic Information of E2E use-case "Derive per-hop time budgets from End-to-End time requirements"

5.3.1 Main Scenario

To implement this use case, the following steps can be performed:

1. a distributed functionality is identified and decomposed into a set of manageable functional components (functions). An end-to-end timing requirement is identified

from the specification (or established from experiments); see *Task Collect Timing Requirements* in Table 6.11

2. the function owner decomposes the function and distributes the sub-functions (functional contributions of components) on a network of ECUs. This results in an allocation of functions on computation and communication resources;
3. optionally, the ECU responsible and the bus responsible persons are asked for an estimation of the required processing time (estimated timing guarantees); see *Task Perform Timing Analysis* in Table 6.13
4. time budgets for each sub-function are derived from the end-to-end timing requirement, if possible respecting the estimated per hop timing.
5. a verification of the found time budgets with respect to the end-to-end timing requirement is performed;

5.3.2 Related Information

Performance Target:	Timing for the distributed function is applicable, optimal use of the resources
Precondition:	End to end timing constraint is known
Frequency:	Whenever a new distributed function has to be implemented
Super-use-case:	n/a
Sub-use-case(s):	NW use-case "Integration of new communication"

Table 5.3: Related Information for this use case

5.3.3 Related methods and properties

- Methods
  - GENERIC METHOD Determine Latency
- Properties
  - GENERIC PROPERTY Latency
  - A suitable syntax to store this and related properties in provided by the AUTOSAR Timing Extensions [2].

5.4 E2E use-case "Deriving timing requirements from the timing assessment of an existing implementation"

This use-case specifies the work flow for function(s) or system owners to derive timing requirements (e.g. deadlines, end-to-end timing) for a system under development from a previous timing assessment of an existing implementation. This may be applied at early phases in the design process for a new build system where timing requirements can not be derived by the analysis-based/simulation-based timing assessment due to e.g. an incomplete system / missing functions, an not existing/specified hardware platform or a missing setup for timing assessments. Moreover this use-case may be applied in case of the migration from a single-core platform to a multi-core platform to where as the timing requirements derived from an existing single-core implementation may be applied to the multi-core implementation of this system. Further possible application are the migration from one network type to another or replacement of an old hardware by a new one.

The AUTOSAR Timing Extensions (TIMEX) [2] represent a suited grammar to formalize the description of timing constraints.

Goal In Context:	Specification of timing requirements for systems under development based on timing assessments of existing implementations (early in the design process without knowledge and access of the final and complete system implementation and behavior).
Brief Description:	This use-case describes the deriving of timing requirements for a system under development from timing assessments of an existing implementation of this system or parts of this system.
Scope:	E2E, NW, ECU
Precondition:	The function(s) or system owner has conducted (or access to) a timing assessment of an existing implementation of the system under development
Success End Condition:	The timing requirements derived from a timing assessment of an existing implementation are mapped and applied to a system under development
Failed End Condition:	The timing requirements derived from a timing assessment of an existing implementation could not be accessed, mapped and/or applied to a system under development
Actor(s):	Functions(s) owner, System Architect

Table 5.4: Characteristic Information of this E2E use-case

5.4.1 Main Scenario

To implement this use case, the following steps can be performed:

1. The function(s) owner has access to or performs a timing assessment of an existing implementation of the system. See *Task Perform Timing Analysis* in Table 6.13
2. The functions(s) owner derives the timing requirement and maps/applies it to the system under development. See *Task Collect Timing Requirements* in Table 6.11.

5.4.2 Related Information

Performance Target:	The timing behavior of the system under development sufficiently corresponds to the timing behavior of the existing implementation (i.e. timing is identical or better).
Precondition:	The timing of the existing implementation is accessible.
Frequency:	Whenever an existing function has to be implemented on a new system.
Super-use-case:	n/a
Sub-use-case(s):	NW use-case “Remapping of an existing communication link”, ECU use-case “Collect Timing Information of a SW-C”, ECU use-case “Create Timing Model of the entire ECU”

Table 5.5: Related Information for this use case

5.4.3 Related methods and properties

- Methods
  - GENERIC METHOD Determine Latency (for timing assessment of the existing implementation)
- Properties
  - GENERIC PROPERTY Latency

5.5 E2E use-case "Specify Timing Requirements for Signals/Parameters"

This use case specifies the work flow for function owners (e.g. of a distributed function), to derive and specify the relevant timing-related properties and requirements in order to consider its communication in the vehicle networking. The timing requirements will be

Goal In Context:	To specify precisely the requirements of a function with respect to the required data communication over the vehicle network.
Brief Description:	This use case requires dedicated reasoning about the timing requirements of a specific function. The function owner identifies for each signal/parameter (i.e. the data to be communicated over the network) the expected cycle time, jitter and latency. To ensure that the requirement is not over-specified the requirements are reviewed by the network designer.
Scope:	E2E, NW
Precondition:	The function owner has partitioned the end-to-end timing constraints for the involved signals into ECU-internal timing requirements and network-related timing requirements (see also Use Case 4.3)
Success End Condition:	The function’s timing requirements have been considered in an explicit a signal/parameter request. The request may also be empty in the case that the function has no timing requirements or these are not related to its signals/parameters.
Failed End Condition:	The function’s timing requirements could not be translated to a signal/parameter request (e.g. because they are not known).

Actor(s):	Function Owner, Network Designer

Table 5.6: Characteristic Information of this E2E use-case

The signal/parameter request shall contain the following information

- the name of the signal/parameter
- the size of the signal/parameter
- the receivers of the signal/parameter as a list of ECUs and/or software components
- the maximum tolerated age of the signal when transmission is completed on the target network.
- the expected update frequency of the signal/parameter
- the accepted jitter for the transmission of the signal/parameter
- a short description of the related functionality

From these values, typically, a signal-to-frame (parameter-to-package) mapping will be derived. In case of designing a CAN configuration, the requesting function owner receives the following parameters for consideration in ECU development:

- the name of the frame
- the transmission property (e.g. periodic)
- the frames cycle time (if relevant)

This information then becomes part the AUTOSAR System Description.

5.5.1 Main Scenario

To implement this use case, the following steps can be performed:

1. The function owner specifies a signal/parameter request that (if so implemented) enables a correct operation of the function.
2. The network designer investigates the signal/parameter request and reviews its content for completeness and adequacy. Indications for non-adequate signal/parameter requests may be if the maximum tolerated age is smaller than the update frequency or if an update frequency of less than the period of the involved tasks is requested. In case of such irritations, the network designer and the function owner iterate until an adequate request has been identified.
3. The network designer documents and files the signal/parameter request.
4. The use case ends.

5. (in the following, the network designer will consider the signal/parameter request to find a suitable signal-to-frame (parameter-to-package) mapping and routing entry for the signal)

5.5.2 Related Information

Performance Target:	Timing for the signal is applicable
Precondition:	Timing constraints for the signal under consideration are known
Frequency:	Whenever the communication requirements of a function changes
Super-use-case:	n.a.
Sub-use-case(s):	NW use-case “Integration of new communication”

Table 5.7: Related Information for this use case

5.5.3 Related methods and properties

- Methods
  - Potentially all methods listed in Table 6.46.
- Properties
  - Potentiall all properties listed in Table 6.15.

5.6 E2E use-case "Assert timing requirements against guarantees"

This use-case compares the metrics derived by analysis of the actual design ("guarantees") to the system's specification ("requirements").

The best outcome of this use case is if the requirements are fulfilled by the guarantees. Otherwise, either requirements need to be relaxed or the guarantees have to be improved (e.g. by reconfiguration of the system).

Goal In Context:	Assess whether the timing of a specific implementation adheres to the timing requirements.
Brief Description:	This use case establishes the comparison of the analysis results of the actual implementation ("guarantees") to the intended behavior as specified ("requirements").
Scope:	ECU, NW, E2E
Precondition:	All relevant timing requirements must be known, any timing requirement that has not been quantified and listed specifically will not be considered in the evaluation.
Success End Condition:	It is known whether all timing requirements are fulfilled by the current implementation. Best outcome is if the requirements are fulfilled by the guarantees. Otherwise, either requirements need to be relaxed or the guarantees have to be improved.

Failed End Condition:	Some timing requirements could not be asserted, possibly because the lack of corresponding timing guarantees.
Actor(s):	Timing Expert, Validation Engineer

Table 5.8: Characteristic Information of this E2E use-case

5.6.1 Main Scenario

This use case is implemented by the following procedure:

1. Establish known timing requirements (Task "Collect Timing Requirements"), for example according to E2E use-case "Deriving timing requirements from the timing assessment of an existing implementation" in E2E use-case "Deriving timing requirements from the timing assessment of an existing implementation"
2. Establish best known guarantees provided by evaluation of the implementation according to ECU use-case ECU use-case "Validation of Timing" and similar procedures for network and end-to-end timing (Task "Perform Timing Analysis").
3. The guarantees, requirements and the comparison of the two are reported (Task "Verify Timing").

5.6.2 Related Information

Performance Target:	n/a
Precondition:	Timing requirements are defined, timing guarantees have been derived.
Frequency:	Whenever timing requirements or timing guarantees change.
Super-use-case:	e.g. "Deriving timing requirements from the timing assessment of an existing implementation" in E2E use-case "Deriving timing requirements from the timing assessment of an existing implementation"
Sub-use-case(s):	none

Table 5.9: Related Information for this use case

5.6.3 Related methods and properties

- Methods
  - Potentially all methods listed in Table 6.46, and the generic methods described in Section 6.5.
- Properties
  - Potentially all properties listed in Table 6.15.

5.7 E2E use-case "Trace-based timing assessment of a distributed implementation"

Whether for understanding, debugging or verifying the timing behavior of a distributed system, tracing of the relevant buses and ECUs (hereafter referred to as subsystems) significantly simplifies timing analysis.

This is even more true if the traces from the various subsystems can be aligned (i.e. synchronized) in order to show cross-subsystem timing effects of event chains such as cross-core communication in a multi-core system, data-buffering-effects when an ECU send/receives data to/from a communication network or even complete end-to-end timing scenarios.

Goal In Context:	Understand, debug and verify the timing behavior of a distributed implementation.
Brief Description:	Tracing observes the real system. For dedicated events such as a start of a task or the presence of a certain message on a bus, time stamps together with event information is placed in a trace buffer which can later be used to reconstruct and analyze the observed scenario. For details, see Measurement and Tracing. For analyzing cross-subsystem timing effects, it becomes necessary to synchronize the traces from all of the relevant subsystems.
Scope:	E2E, ECU, NW
Precondition:	Existing and executable system
Success End Condition:	Tracing performed and data (=traces) ready for analysis; if necessary, traces from different subsystems (cores, ECUs, buses) are aligned, i.e. synchronized.
Failed End Condition:	No or not all relevant scheduling entities could be traced or traces could not be aligned (i.e. synchronized)
Actor(s):	Integrator, Timing Expert, Validation Engineer

Table 5.10: Characteristic Information of this E2E use-case

5.7.1 Main Scenario

This use case is implemented by the following procedure:

1. The timing expert/test engineer prepares the measurement and the system under test (tools, software...). See *Task Perform Timing Analysis* in Table 6.13
2. The timing expert/test engineer performs a correlated (i.e. synchronized) tracing of an existing implementation of the system under consideration.
3. The timing expert/test engineer checks the quality of the traces and if quality is sufficient.

5.7.2 Related Information

Performance Target:	Verification or determination of the actual timing behavior
Precondition:	Accessible resource

Frequency:	Whenever timing information about the actual implementation are needed
Super-use-case:	n/a
Sub-use-case(s):	• ECU use-case "Collect Timing Information of a SW-C" • ECU use-case "Validation of Timing" • ECU use-case "Verify Timing Model(s)"

Table 5.11: Related Information for this use case

5.7.3 Related methods and properties

- Methods
  - GENERIC METHOD Determine Latency
  - GENERIC METHOD Determine Load
- Properties
  - GENERIC PROPERTY Latency
  - GENERIC PROPERTY Load

6 Properties and Methods for Timing Analysis

6.1 General Introduction

This section describes the general relations between timing use-cases (see chapters 3, 4 and 5) and timing tasks. The timing tasks, the timing properties and the timing methods are specified in details in this chapter.

The timing use-cases (for example "Design and configuration of a new network") presented in the former chapters (related to ECU, network, or end-to-end views) usually consist of several smaller steps (listed under "main scenario" in each use case). Some of these steps are fundamentally related to timing and reappear in in several use cases. We call these steps "timing related tasks" and outline them in more detail in Section 6.3.

One particularly important timing-related task is *Perform Timing Analysis* defined in Table 6.13. This timing related task can again comprise a "timing method" (see Section 6.5, which specifies in more detail how to solve this task, i.e. through simulation or static analysis.

The inputs (e.g. "the communication matrix" or "measured core execution times") for the timing methods arise from the system specification or from observing the real system. Some of the methods deliver timing properties as an output (e.g. "worst case response time of the transmitted message") which can be evaluated against timing constraints (for example the function may require that the frame transmission is completed in less than 10ms) during the timing task *Verify Timing*.

Important, but out of scope in this document is the implementation of timing methods and timing properties in tools. The approach and the timing terminology are illustrated in Figure 6.1 and 6.2.

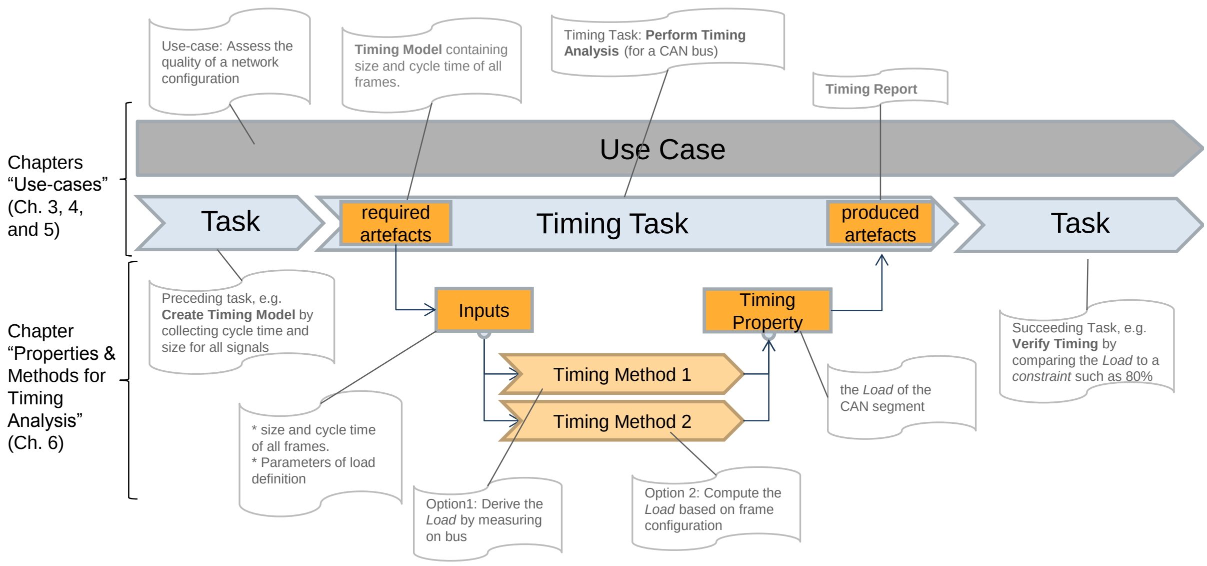

Figure 6.1: Illustration of hierarchy between use cases, timing properties, and timing methods (and related sections).

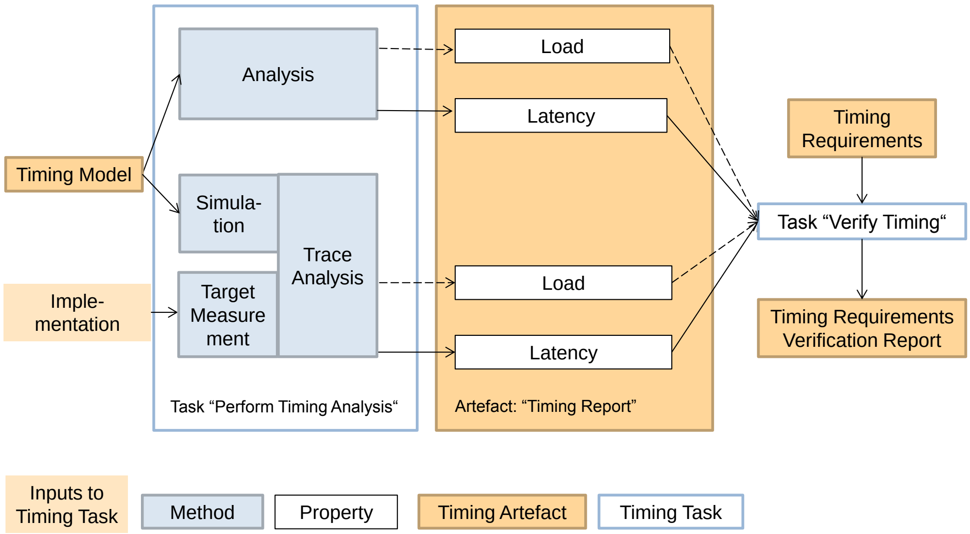

Figure 6.2: The interplay between different timing methods, timing properties and constraints

6.2 A Simple Grammar of Timing Properties

In order to avoid repeating similar definition of timing properties and methods in the following sections, this document follows a generic approach. Timing properties are described with supporting placeholders, such as for example “<schedulable>” and “<resource>”. A “<resource>” can be either e.g. a “CPU” or a “CAN bus”, and a “<schedulable>” can be the corresponding e.g. “RunnableEntity", "BswSchedulableEntity" or “frame”.

Not all combinations of such terms lead to relevant/valid definitions. Therefore the actual instances are listed with the definitions. For reasons of practicality, the document however presently does not formalize the placeholder structure into a complete and consistent grammar (but such refinement may be possible in future releases).

6.2.0.1 Resources and Schedulables

<Resources> are needed to execute <schedulables>. They can schedule between several <schedulables> over time, based on an online or offline scheduling scheme. <Resources> have the capability to compute, store, transmit or receive information.

<Resources> can be divided in two categories: <unary resources>, which can execute only one <schedulable> at any given time and <multi resources> which can execute multiple <schedulables> in parallel.

A <schedulable> computes, stores, or transmits information on a <resource>. In order to make progress it must be assigned the <resource> in the scheduling process.

<Resource>	
<Unary Resource>	**Allowed <Schedulable>**
CAN bus segment	CAN frame
Single-Core CPU	Task
FlexRay Segment	FlexRay frame
Ethernet Link	Ethernet message
LIN bus	LIN frame
<Multi Resource>	
Switched Ethernet-Network	Ethernet message
Multi-Core CPU	Task

Table 6.1: Resource Overview

Note: <Multi Resources> are not covered by any of the present definitions in the document.

The timing of a schedulable is defined by its <activate> and its <terminate> events. The <activate> is the moment in time at which the <schedulable> becomes ready to perform its operation, and the <terminate> is the moment in time when it is finished.

A <schedulable> may contain <subschedulables> to differentiate between different operations.

<Schedulable>	**Allowed <Subschedulable>**
Processor Task (equivalent: ISR)	Runnable BSW function
OS-Function	RunnableEntity, BswSchedulableEntity
CAN frame	PDU Signal

Table 6.2: Allowed Schedulable

6.2.0.1.1 Example: Scheduling situation with tasks on a single-core CPU

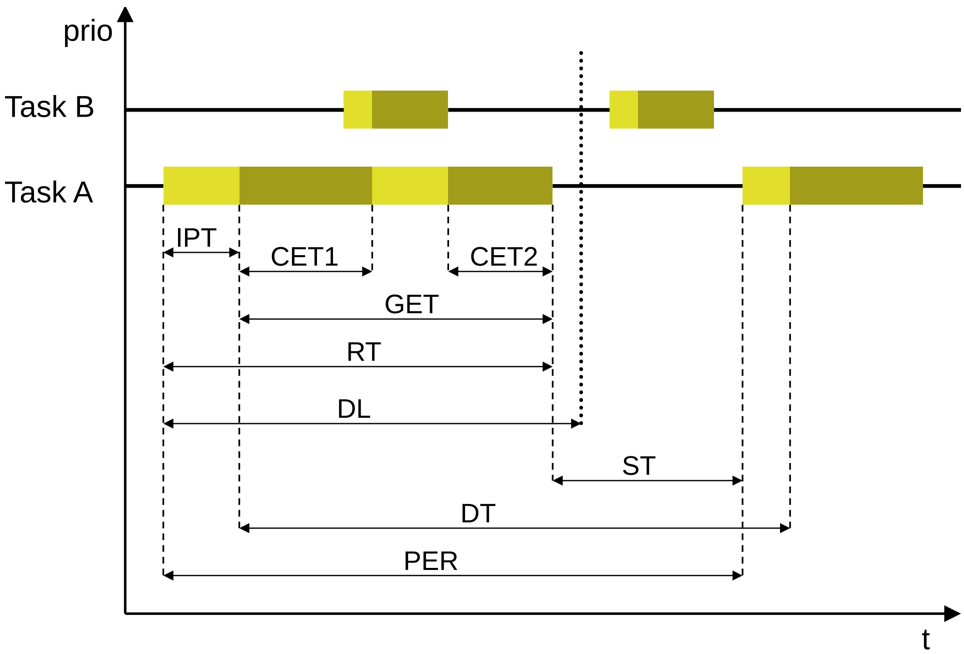

Figure 6.3: Timing properties relevant for tasks scheduled on a single-core CPU

Abbreviation	Description
IPT	Initial pending time
CET	Core execution time
GET	Gross execution time
RT	Response time
DL	Deadline
DT	Delta time
PER	Period
ST	Slack time
PRE	Preemption time *(not shown in the figure)*
JIT	Jitter *(not shown in the figure)*
CPU	CPU load *(not shown in the figure)*

Table 6.3: Some important timing parameters (run-time situation on a single-core CPU)

6.2.0.2 Method of Derivation

The different timing properties can be derived with various methods, while not every property can be properly derived with every method (but often approximated). For example, during simulation, the message load can be observed, but it is difficult to derive the real worst-case latency. For the purpose of this document, we differentiate between the following methods (see for more details in Definition, Description and Classification of Timing Methods :

<TimingMethod>	Explanation

Analysis	Computation or theoretical estimation of the value of the timing property
Simulation	Simulation of a system to determine the temporal development of the value of the timing property
Measurement	Measurement of a target to determine the temporal development of the value of the timing property

Table 6.4: Method of Derivation

6.2.0.3 Statistical Qualifier

Many timing properties can be tailored to different <Statistical Qualifiers>. For example, one may be interested in the average latency of a message in one case and in the maximum latency in another (for example if it is a time critical message as e.g. the total time in the active steering example). Base to do this is to determine the temporal development of the latency over the time by means of e.g. the simulation and to derive the relevant quantities like the average latency. This can be more generalized to the determination of the temporal development of an arbitrary quantity ("‘x-over-Time"’) and to derivation of the distribution and its momenta.

For this reason, the following <Statistical Qualifiers> are introduced:

Method	**<Statistical Qualifier>**	**<Statistical Qualifier>** derived quantity
Analysis	Best-Case	
	Worst-Case	
Simulation / Measurement	Distribution / X-over-time	Minimum
		Maximum
		Average

Table 6.5: Different Types of Timing Methods and the resulting Statistical Qualifiers

The x-over-time and the distribution depend on the related timing method, the input parameters and the boundary conditions. In contrast, the analysis approach delivers the timing property as a single value (e.g. worst-case). The (best-)worst-case denotes the state of the system with the (minimum) maximum system requirement, sometimes overestimated by the applied algorithm. However, the (minimum) maximum represents the actual observed value of the timing property here in this context.

6.2.0.4 Constraint Type

Finally, in accordance with the definition in TIMEX, the actual value of the timing property can be interpreted as a requirement (a priori to an analysis) or the worst-case can be regarded as a guarantee for the system specification (a posteriori to an analysis).

<ConstraintType>
Requirement
Guarantee

Table 6.6: ConstraintType

Figure 6.4 sketches the interplay between the value of the timing property (and its development over time and its distribution) and the constraints. The value of the timing property results from the timing method. Some of the Statistical Qualifiers are indicated on the left hand side of the distribution. Here, the guarantee results from the worst case analysis of the timing property of interest (in more general case defined in the performance specification) whereas the external requirement (defined in the requirement specification) for this timing property cannot be fulfilled in this case.

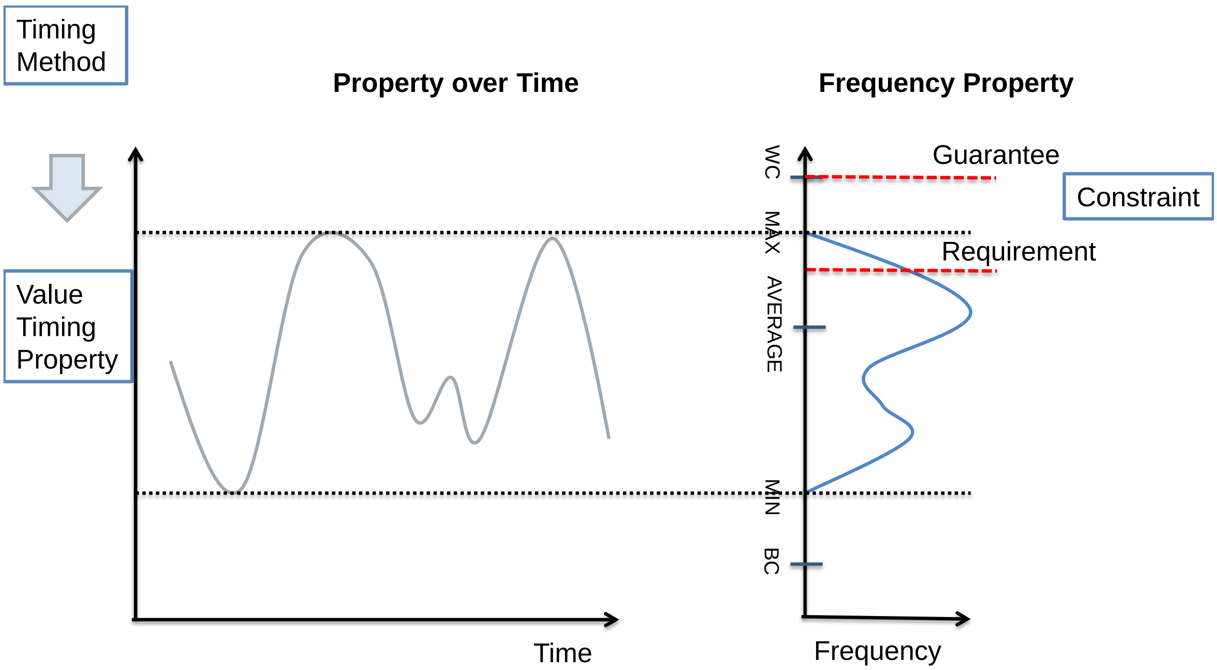

Figure 6.4: The figure illustrates the relation between the timing method, the timing property, the constraint and qualifiers (see text for more details). Here, the actual implementation does not fulfill the requirement.

6.2.1 Protocol Specifica

6.2.1.1 CAN

In order to define properties for the CAN bus the following definitions are used:

A CAN frame consists of	
Header	Standard 19 bit Extended 37 bit
Payload	0..8 byte (CAN-FD 0..64 byte)

Stuff bits	0..19 (Extended 0..25) bit
Footer	25 bit
Inter frame space	3 bit

Table 6.7: Definition length parameter for a CAN

Summing up all parameters together yields the frame length/time (l_{frame}/t_{frame}). Thus,the general CAN properties and parameters are given by:

Scheduling	Static Priority Non-Preemptive
Activation	Periodic and/or event triggered
ID (priority)	Standard 0..0x7FF
Speed	Standard 100.. 1000 KBaud
Frame length	Standard 47..130 bit

Table 6.8: Definition general parameter for a CAN

For the application of the generic description to the CAN bus the following relations are applied:

Generic parameter	**Actual value**
<resource>	CAN bus segment
<schedulable>	CAN frame
<activate>	Event TDEventFrame.frameQueuedFor Transmission on sender ECU
<terminate>	Event TDEventFrame.frameTransmittedOnBusbetween network and receiver ECU

Table 6.9: Relation between the general and the CAN specific parameters

6.2.1.2 Activation

Frame	Definition
Periodic Frame	A frame that is activated periodically with period defined by the “cycle time”
Event-Triggered Frame	A frame that is activated sporadically by an external event.
Mixed Frame	A frame that is activated by the passing of the period or an external event. Different concepts on treating the periodic part exist (i.e. resetting of the periodic timer on arrival of sporadic events).

Table 6.10: Definitions of the frame activation

More complex activation pattern for frames in the scope of Autosar can be defined. Furthermore, OEM specific transmission modes exist.

6.3 Description of Timing Tasks

This section introduces timing tasks that have to performed in order to accomplish the use-cases described in chapter 3, 4 and 5.

Activity	**Collect Timing Requirements**		
Brief Description	Collect Timing Requirements		
Description	Collect the known timing requirement from the specification documents. If necessary, find timing requirements in discussions/interviews with function owners and system architects. Derive new timing requirements from traces, measurements and experiments.		
Relation Type	***Related Element***	***Mul.***	***Note***
Consumes	Function specification document	1	n.A.
Performed by	Timing expert, function owner and system engineer	1	n.A.
Produces	Timing requirements document (TIMEX Extract)	1	n.A.

Table 6.11: Task "Collect Timing Requirements"

Activity	**Create Timing Model**		
Brief Description	Create analyzable timing-model		
Description	The model parameters are derived from timing-related information. Incomplete information is estimated or generated based on synthesis-rules. Additional assumptions/operation scenarios/boundary conditions are documented.		
Relation Type	***Related Element***	***Mul.***	***Note***
Performed by	Timing expert, network designer, system engineer	1	n.A.
Produces	Timing model	1	n.A.

Table 6.12: Task "Create Timing Model"

Activity	**Perform Timing Analysis**		
Brief Description	Gain timing properties by performing timing analysis		
Description	**Option 1 (implementation-based analysis)** Set up the environment (e.g. HIL or car) for the device to be analyzed. Define stimulation model. Measure/trace the observable events and derive the timing properties. See Measurement and Tracing on page 101. **Option 2 (model-based analysis)** Based on the timing model, analyze the timing (e.g. by simulation or static analysis; see Analytical calculation on page 97) and derive the timing properties.		
Relation Type	***Related Element***	***Mul.***	***Note***
Predecessor	Create Timing Model	1	n.A.
Performed by	Timing expert, software developer, integrator or test engineer	1	n.A.
Produces	Timing analysis report with timing properties according to Definition and Classification of Timing Properties on page 84	1	n.A.

Table 6.13: Task "Perform Timing Analysis"

Activity	**Verify Timing**		
Brief Description	Verify adherence of timing requirements against timing properties		
Description	Compare the timing requirements to the results of the timing analysis (timing properties). Generate a report that documents which timing requirements are fulfilled and which not.		
Relation Type	***Related Element***	***Mul.***	***Note***
Predecessor	Collect Timing Requirements	1	n.A.
Predecessor	Perform Timing Analysis	1	n.A.
Consumes	Timing requirements document	1	n.A.
Consumes	Timing analysis report	1	n.A.
Performed by	Timing expert	1	n.A.
Produces	Timing requirements verification report	1	n.A.

Table 6.14: Task “Verify Timing”

6.4 Definition and Classification of Timing Properties

6.4.1 Classification and Relation of Properties

The properties can be grouped in two main fields: capacitive (<resource> capacity) and latency property (<schedulable> latency). Capacitive properties are the ratio of the capacity requirement by the <schedulables> to the capacity of the <resource>. Latency properties are the delays of <schedulables> due to the schedule (priority schema) on the common used <resource>.

6.4.2 Summary of regarded Timing Properties

NW/ECU	**Group**	**Name**
Generic	Load	GENERIC PROPERTY Load
NW	Load	SPECIFIC PROPERTY Load (CAN)
Generic	Latency	GENERIC PROPERTY Latency
Generic	Latency	GENERIC PROPERTY Response Time
NW	Latency	SPECIFIC PROPERTY Response Time (CAN)
Generic	Latency	GENERIC PROPERTY Transmission Time
Generic	Latency	SPECIFIC PROPERTY Transmission Time (CAN)
ECU	Latency	SPECIFIC PROPERTY Execution Time

Table 6.15: Overview about the here described Timing Properties

6.4.3 GENERIC PROPERTY Load

6.4.3.1 Scope and Application

Name	Load
Description	The load is the total share of time that a set of <schedulables> occupies a <single resource>. If the time for the occupation is calculated it can exceed the available resource time (overload). In the practical realization using simulation or measurement this scenario cannot occur. But, if the transmission load of all <schedulables> exceeding 100% (amount of send requests) is not buffered the required to transmit information can be lost or overridden.
Application	The property supports the estimation of the resource needs in ECUs and gateways and of the network, respectively.
Assumptions and Preconditions	• The time of the occupation for every individual <schedulable> is known. • The partition for the total communication amount in individual <schedulables> is done.

Table 6.16: Scope and Application

6.4.3.2 Relation

Process Steps	The method can be used at every iteration step at which a timing model or a measurement of the system is available.
Referencing Use-cases	• NW use-case “Integration of new communication” on page 57 • NW use-case “Design and configuration of a new network” on page 60 • NW use-case “Remapping of an existing communication link” on page 63 • ECU use-case “Collect Timing Information of a SW-C” on page 37 • ECU use-case “Validate Timing after SW-C integration” on page 40 • ECU use-case “Validation of Timing” on page 41 • ECU use-case “Optimize Scheduling” on page 47
Belonging (Pre) Methods	GENERIC METHOD Determine Load
Relation to AUTOSAR Timing Extension	The AUTOSAR Timing Extensions define no constraint related to load.

Table 6.17: Relation

6.4.3.3 Interface

Notation	$L(t, t_{window}, ...)$
Possible<Statistical Qualifiers>	All which were mentioned in the introduction

Parameters	t_{window}	The size of the time interval over which the load is determined. Recommended value: large but finite value
	t	The end of the time interval over which the load is determined. This parameter is required for load-over-time analysis. Default value: not specified
Range	0 to 100% (0.. infinity for calculation)	

Table 6.18: Interface

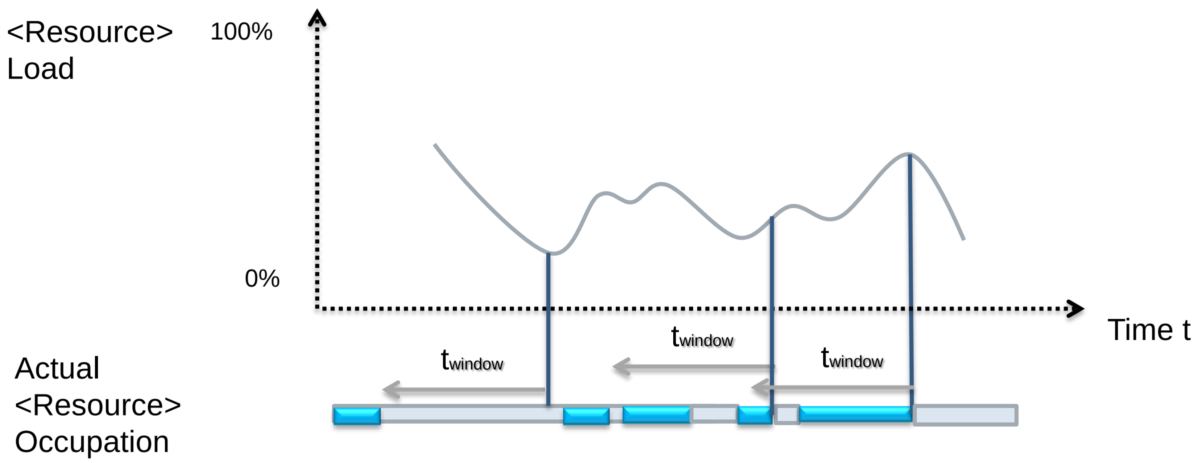

Figure 6.5: Illustration of the relation of the actual occupation and the load over time. The load $L(t, t_{window})$ is the average of the occupation over the interval t_{window} till the point in time t.

6.4.3.4 Expressiveness

The “load” indicates the overall utilization of a given <single resource>. A small load is better for stable operations due to safety and extensibility reasons. However, it shows that the <single resource> is not fully utilized, possibly missing opportunities for cost-optimization. On the other hand the remaining free resources can be used for future extension and therefor are intentionally reserved.

From perspective of real-time applications and schedulable with timing constraints, the expressiveness of load is limited. A load value below 100% allows deducing the guarantee that eventually every instance of each <schedulable> will be scheduled and executed on the <resource>. However, the completion time of a schedulable may be larger than its period or any given deadline.

Actually, the correlation to the <schedulable’s> worst-case response time is small. Depending on the schedule there are examples with high load and small over-all response times and with low (but highly variable) load and high over-all response time. (compare latency, timing property worst-case response/execution time).

In [13], it was shown that given only periodic <schedulables> with deadlines equal to their periods, all <schedulable> will be serviced before their deadline if the load is smaller than 69% (independent <schedulables>). However, in practice, the presence of sporadically activated <schedulables> avoids a direct applicability of this statement.

6.4.4 SPECIFIC PROPERTY Load (CAN)

6.4.4.1 Scope and Application

Name	Bus Load (CAN)

Table 6.19: Scope and Application

6.4.4.2 Relation

Process Steps	The method can be used at every iteration step at which a timing model or a measurement of the system is available.
Use-case	see table 6.17
Super Property	GENERIC PROPERTY Load
Sub Property (s)	n/a
Belonging (Pre) Methods	SPECIFIC METHOD Determine Load (CAN)
Post Methods	n/a

Table 6.20: Relation

The share of time can be calculated from the <activate> and <terminate> for the target measurement/simulation and from the frame length (see CAN) for analysis and the activation pattern (see Activation).

6.4.4.3 Interface

The Bus load (CAN) is an instance of the property GENERIC PROPERTY Load with the parameters described in Table 6.9.

Depending on the activation patterns, the following CAN loads are differentiated:

Periodic load	The share of time that the set of periodic frames occupies the bus.
Total load	The share of time that all frames (periodic and event-triggered, including the mixed-triggered frames) occupy the bus.

Table 6.21: Different kinds of Bus Load of a CAN Segment depending on the frame activation

6.4.4.4 Expressiveness

During run time, the CAN bus and the transmitted frames typically exhibit dynamic behavior:

- frame periods may slightly fluctuate from the specified cycle time (jitter and drift)
- the number of stuff bits depend on the actual payload
- the frame may not always carry the same amount of payload with each transmission

Depending on the selected <Statistical Qualifier> (i.e. average, maximum, ...) the properties of the CAN configuration may need to be interpreted differently due to this dynamism.

6.4.5 GENERIC PROPERTY Latency

6.4.5.1 Scope and Application

Name	Latency
Description	The latency is the amount of time between the <activate> of the first <schedulable> (it is ready to transmit on/occupy a <resource>) in a list of <schedulables> and the <end> of the last <schedulable> (freed from the occupation) in the list. This includes scheduling effects. Depending on the timing property of interest and the nature of an application, two types of latency (also called "‘semantics"’) can be distinguished: the reaction time latency, which is the amount of time of the first reaction to a change in the input, and the data age latency, which is the amount of time that a that a certain input data may be processed before updated input values are available. See AUTOSAR Timing Extensions (TIMEX) [2] LatencyConstraint for details.
Application	The property supports the estimation of the resource needs and the rescheduling in ECUs and gateways and of the network, respectively.
Assumptions & Preconditions	For each <resource> is known: • The access schema/arbitration strategy like bus protocol or OS scheduling • All occupation of a <resource> is error free, i.e. every utilization by the <schedulable> takes place exactly once. For each individual <schedulable> is known • The priority of the <schedulables> • The repsonse times • The triggering/activation schema including any send delay

Table 6.22: Scope and Application

6.4.5.2 Relation

Process Steps	The method can be used at every iteration step at which a timing model or a measurement of the system is available.
Referencing Use-cases	• NW use-case "Integration of new communication" on page 57 • NW use-case "Design and configuration of a new network" on page 60 • NW use-case "Remapping of an existing communication link" on page 63
Belonging (Pre) Methods	GENERIC METHOD Determine Latency
Post Methods	The property can be used for computation of the real-time slack of the system.
Relation to AUTOSAR Timing Extension	The AUTOSAR timing extension defines the LatencyTimingConstraint to specify constraints on the latency of TimingDescriptionEventChain. TimingDescriptionEventChain refers to specific events in the system.

Table 6.23: Relation

6.4.5.3 Interface

Notation	$T(t, t_{window}, X...)$	
Possible<Statistical Qualifiers>	All in the introduction mentioned	
Parameters	X	The information package for which to compute the response time
	t_{window}	The size of the time interval over which the latency is determined, i.e. the temporal interval of a trace in which the latency is determined. Default value: INF
	t	The beginning or end of the time interval over which the latency is determined. This parameter is required for X-over-time analysis. Default value: not specified
Range	0 to infinity	

Table 6.24: Interface

6.4.5.4 Expressiveness

For every hop (element) of the sequential <schedulable> list the latency per hop of the <schedulable> measures the temporal delay for its utilizations of related <single resource>. A small latency is better for stable functional operations due to safety and extensibility reasons. However, it shows that at least a part of the <resources> are not fully utilized if the latency is too small against the end-to-end deadline, possibly missing opportunities for cost-optimization. Nevertheless the latency must be smaller than the end-to-end deadline, otherwise information loss may occur. If a considerable part of <schedulables> misses their deadlines for one of the <single resource> it has not enough capacity or the schedule is not sufficiently good.

Errors during a transmission or an execution of <schedulable> may lead to a re-transmission/re-execution of specific <schedulable> which increases both the load and the latency.

The worst-case of the latency can be derived by model based formal analysis methods such as presented in [14]. By this, the latency property is conservatively computed.

The worst-case of the latency can be approximated by simulation, albeit only optimistically. The related transmission/execution requests and transmission/ execution complete events can be randomly generated and observed. The maximum of the observed values is an optimistic approximation of the worst-case latency.

When the property is derived using different methods (especially simulation/analysis and measurement) the following must be true (WC abbreviates worst case) considering only one element of the sequential <schedulable> list:

$WC\ Latency_{Analysis}(\text{<Schedulable>}) \geq WC\ Latency_{Simulation}(\text{<Schedulable>})$ and
$WC\ Latency_{Analysis}(\text{<Schedulable>}) \geq WC\ Latency_{Measurement}(\text{<Schedulable>})$

6.4.6 GENERIC PROPERTY Response Time

6.4.6.1 Scope and Application

Name	Response Time
Description	The response time is the special case of the latency concerning only one single <schedulable>, i.e. is the amount of time between the <activate> of the <schedulable> and the <end> of the <schedulable>.This includes scheduling effects of a concurrent access to a shared <resource>. One can distinguish between a static priority pre-emptive access (in case of OSEK and other operating systems) and a static priority non-pre-emptive access (in case of CAN and most other networking systems). The response time of a <schedulable> is equal to its GENERIC PROPERTY Transmission Time or SPECIFIC PROPERTY Execution Time in the case where the resource is exclusively available to this <schedulable>. In the presence of multiple <schedulables> that are ready at the same time, the resulting response times are defined by the actual schedule.

Assumption and Precondition	For each periodic and for each mixed activation the following is known: • Period • Reference clock (optional) • Offset to reference (optional) For each event triggered and for each mixed activation the following is known: • Event model of sporadic events including minimum arrival time

Table 6.25: Scope and Application

6.4.6.2 Relation

Process Steps	The method can be used at every iteration step at which a timing model or a measurement of the system is available.
Use-case	see table 6.23
Super Property	GENERIC PROPERTY Latency
Sub Property (s)	SPECIFIC PROPERTY Response Time (CAN)
Belonging (Pre) Methods	n/a
Post Methods	n/a
Relation to AUTOSAR Timing Extension	The AUTOSAR timing extension defines the LatencyTimingConstraint to specify constraints on the latency of EventChains. The response time of a <schedulable> corresponds to the Latency of a (short) EventChain constituted by the activation event and the termination event of the considered executable entity.

Table 6.26: Relation

6.4.6.3 Interface

The response time is an instance of the property GENERIC PROPERTY Latency with the following parameters:

Generic parameter	Actual value
<resource>	Single <resource>
<schedulable>	Single <schedulable>

Table 6.27: Relation between the general latency and the response time

Notation	$T_{Response}(t, t_{window},$ <schedulable X>, <schedulables>)	
Parameter	*<schedulable X>*	The <schedulable> for which to compute the response time.
	<schedulables>	The remaining <schedulables> interacting with <schedulable X>.

Table 6.28: Interface

6.4.6.4 Expressiveness

The expression of the response time as defined is limited in some sense:

1. In the case of a large number of non-harmonic time bases, analysis time can grow beyond acceptable times. In this case, some offset relations can be ignored during analysis which may slightly decrease accuracy.

6.4.7 SPECIFIC PROPERTY Response Time (CAN)

6.4.7.1 Scope and Application

Name	Frame response time (CAN)
Description	The property provides the total time from when a frame is ready to send (<activate>, i.e. placement of the frame in an output message buffer of the CAN driver) until a frame is completely transmitted over a bus (<end>, i.e. usually leading to a Tx IRQ on a receiving ECU).
Application	The property allows assessing the communication delay of a timing critical frame.
Assumptions and Precondition	• It is assumed that all communication on the bus is error free, i.e. every transmission takes place exactly once. • It is assumed that of all frames in a network that are ready to send, the CAN bus always selects the one with the lowest CAN-ID for transmission. • For each frame on the bus, the following is known: – Frame length including stuff bits – CAN-ID

Table 6.29: Scope and Application

6.4.7.2 Relation

Process Steps	The method can be used at every iteration step at which a timing model or a measurement of the system is available.
Use-cases	see table 6.22
Super Property	GENERIC PROPERTY Response Time
Sub Property (s)	n/a
Belonging (Pre) Methods	SPECIFIC METHOD Determine Response Time (CAN)

Post Methods	The property can be used for computation of the real-time slack (available bandwidth after accommodating all frames specified in the communication matrix).
Relation to AUTOSAR Timing Extension	The AUTOSAR timing extension defines the LatencyTimingConstraint to specify constraints on the latency of EventChains. The response time of a <schedulable> corresponds to the Latency of a (short) EventChain constituted by the activation event and the termination event of the considered executable entity. In case of the response time of a CAN frame the events are identified by the Timing Description Events on the System View level. The activation event is given by the Timing Description Event TDEventFrame(TDEventType=frameQueuedForTransmission) and the termination event is given by TDEventFrame(TDEventType=frameTransmittedOnBus).

Table 6.30: Relation

6.4.7.3 Interface

Notation	$T_{Response}(t, t_{window},$ frame X)	
Parameter	*frame X*	The frame for which to compute the response time.
	stuff bits	The number of stuff bits to be assumed during analysis.

Table 6.31: Interface

6.4.7.4 Expressiveness

The expression of the response time for CAN as defined is limited in some sense:

1. Due to internal buffer structure, some CAN controllers may not be able to always provide the frame with the lowest CAN-ID (highest priority) that is ready to send to the bus arbitration. This can lead to a priority inversion with potentially larger response times than as defined by this property.
2. It is difficult to measure latency in target setups. While it is easy to identify the transmission complete events by probing the bus, the point in time when a frame becomes ready to send is more difficult (black box measurement). One option is to estimate the time by checking the bus busy time before the transmission complete event. Another option is to combine an ECU internal trace with the network trace using a reference time base.

These constraints are in part relaxed by current research such as [15], [16].

6.4.8 GENERIC PROPERTY Transmission Time

6.4.8.1 Scope and Application

Name	Transmission time
Description	The property is the special case of the response time without concerning any scheduling effects. The property provides the pure time for transmitting a <schedulable> on a single <resource> without considering any other <schedulable> on this <resource>.

Table 6.32: Scope and Application

6.4.8.2 Relation

Process Steps	The method can be used at every iteration step at which a timing model or a measurement of the system is available.
Use-cases	see table 6.22
Super Property	GENERIC PROPERTY Response Time
Sub Property (s)	n/a
Belonging (Pre) Methods	n/a
Post Methods	n/a
Relation to AUTOSAR Timing Extension	There is no direct constraint related to transmission times defined in the AUTOSAR timing extension.

Table 6.33: Relation

6.4.8.3 Interface

The transmission time is an instance of the property GENERIC PROPERTY Latency (and of course a special case of GENERIC PROPERTY Response Time) with the following parameters:

Generic parameter	Actual value
<resource>	Single <resource>
<schedulable>	Single <schedulable>

Table 6.34: Relation between the general latency and the transmission time

Notation	$T_{Response}(t, t_{window},$ <schedulable X>)	
Parameter	*<schedulable X>*	The <schedulable> for which to compute the transmission time.

Table 6.35: Interface

6.4.9 SPECIFIC PROPERTY Transmission Time (CAN)

6.4.9.1 Scope and Application

Name	Transmission time (CAN)

Assumptions and Precondition	• For each frame on the bus, the following is known: Frame length including stuff bits

Table 6.36: Scope and Application

6.4.9.2 Relation

Process Steps	The method can be used at every iteration step at which a timing model or a measurement of the system is available.
Use-cases	see table SPECIFIC METHOD Determine Response Time (CAN)
Super Property	GENERIC PROPERTY Transmission Time
Sub Property (s)	n/a
Belonging (Pre) Methods	n/a
Relation to AUTOSAR Timing Extension	There is no direct constraint related to transmission times defined in the AUTOSAR timing extension. A related constraint is the ExecutionTimeConstraint to specify constraints on the execution time of ExecutableEntities, but this is aimed at software timing only.

Table 6.37: Relation

6.4.9.3 Interface

The Transmission time (CAN) is an instance of the property GENERIC PROPERTY Transmission Time with the parameters described in Table 6.9.

Notation	$T_{Transmission}(t, t_{window},$ frame X)	
Parameter	*frame X*	The frame for which to compute the transmission time.
	stuff bits	The number of stuff bits to be assumed during analysis.

Table 6.38: Interface

6.4.10 SPECIFIC PROPERTY Execution Time

6.4.10.1 Scope and Application

Name	Execution Time (ET)
Description	The execution time indicates a time required for a certain computation. In this context a computation can be a runnable, a subfunction or just a sequence of commands.
Goal	This property is a required input information for run time budgeting and the ECUs schedule feasibility.
Assumptions	The execution time is deterministic.

Table 6.39: Scope and Application

6.4.10.2 Classification

System	ECU
Applied Network	n/a

Table 6.40: Classification

6.4.10.3 Relation

Process Steps	The method can be used at every iteration step at which a timing model or a measurement of the system is available.
Use-cases NW	n/a
Methodology/Task ECU	The property execution time (ET) is related to the methodologies
Use-case ECU	• ECU use-case "Collect Timing Information of a SW-C" • ECU use-case "Validate Timing after SW-C integration" • ECU use-case "Optimize Timing of an ECU" • ECU use-case "Optimize Scheduling" • ECU use-case "Optimize Code" • ECU use-case "Compare Timing Properties"
Super Property	GENERIC PROPERTY Latency
Sub Property (s)	n/a
Belonging (Pre) Methods	• Simulation via target simulator
Post Methods	Worst-Case Response Time (WCRT) analysis
Relation to AUTOSAR Timing Extension	The AUTOSAR timing extensions define an ExecutionTimeConstraint to specify constraints on the execution time of ExecutableEntities. The "'net"' and "'gross"' execution times are distinguished. The Property Execution Time as defined above corresponds to the "'gross"' executionTimeType (which includes calls to external functions).

Table 6.41: Relation

6.4.10.4 Interface

Output	The scalar result value is usually stated in micro-, milli- or nanoseconds.
Range	0 to infinity

Table 6.42: Interface

6.4.10.5 Expressiveness

For hard real-time systems an important statistical qualifier (also see Constraint Type) is the worst case execution time (WCET) that is required to complete a certain computation. The WCET is an indicator for resource consumption usually a predefined value must be reached or derived. To predict and proof the correct software execution the WCET is an important property. In practice it is recommended to use different timing methods to determine the WCET in order to gain the confidence of the result. These methods are static, dynamic and hybrid approaches. If it is not possible to determine the WCET in the field an upper safe limit needs to be used as equivalent. Based on the worst case execution time of several computations one or more WCRT (worst-case response time) might be determined which in most cases are more relevant.

For ECU use-case "Optimize Code" next to the WCET the average execution time can be interesting. Huge differences between the both of them or the average execution time and the maximum execution time usually indicate optimization potential.

6.5 Definition, Description and Classification of Timing Methods

6.5.1 Classification and Relation of Methods

Roughly, the methods can be grouped in three main fields: simulation, analytical calculation and measurement. Another criterion to distinguish methods is to consider the origin of the data: model-based or measurement-based. This classification is closely related to the moment in which stage of the timing process the method can carry out (in the specification phase or verification phase).

6.5.1.1 Analytical calculation

Static Code Analysis works on the source code or binary code level of an executable software or part of it. A distribution of SPECIFIC PROPERTY Execution Time is determined. Therefore the call graph and the instruction sequence is reconstructed and analyzed. A lower limit for the BCET (best case execution time) and the upper bound for the WCET (worst case execution time) is calculated for a given code fragment (e.g. a function) by applying Statistical Qualifier. Beside the software that should be analyzed, symbol information and annotations for additional constraints (e.g. build options, range of input values, integration/hardware specific constraints) must be provided. Any real core execution time is guaranteed to be within this interval, as long as this fragment is not interrupted. Furthermore, any data present only at run-time (e.g. upper bounds on the loop iterations and the content of dynamic function pointers) has to be provided manually in the form of additional annotations. For a proper static code analysis the target hardware behavior must be known in detail (e.g. access time for different memory areas, caching, and so forth). In modern systems the behavior model can be quite complex and therefore limitations regarding the results precision may occur.

The results of static code analysis should be validated with the results from alternative methods secripted in Definition, Description and Classification of Timing Methods.

Scheduling analysis Based on the model of a certain scheduler (e.g. a certain RTOS), scheduling analysis tools take minimum/maximum core execution times and an application model as input and provide e.g. the guaranteed WCRT. This allows checking whether any deadlines will be missed under the given conditions. For each task's and interrupt's worst case, a trace is generated allowing to analyze the run time situation under which it occurs. The execution times fed into the analysis can be either budgets, estimations, or outputs from other tools, e.g. statically analyzed BCET/WCET or traced/measured data. Thus, scheduling analysis allows to verify new concepts without implementing them as an advandage. Furthermore existing concepts can be amended for concept verification or solution space exploration.

Network analysis Network analysis for a single network segment computes the worst-case response time for each frame/package transferred via the network. This is usually possible based on the same type of design data that is needed to configure the connected ECUs (e.g. AUTOSAR System Extracts). The main information is the (gross) size of each frame/package (e.g. based on the size of the contained signals/parameters and the protocol header), the frame's transfer properties (i.e. its cycle time or debounce time of external triggers), and of course the transmission speed of the network. The analysis takes conflicts on the networks and synchronization between frames into account when computing the worst-case response times. This basic result can be aggregated into a complete timing profile for a bus or gated network. In case of highly dynamic timing behavior network analysis can be mixed with measurement-based approaches by replacing model-based design data with event traces from actual measurements.

Compositional analysis Compositional analysis allows to consider an activity chain consisting of different <schedulable> on different <resources>. It adds the chained response time of different <schedulable> on one <resource> in a first step and then the response time of the resources. If a worst case consideration is made, this method can be very conservative as in reality the probability of a worst case response time on several chained resources is by far lower than the probability of a single resource worst case, which by itself is conservative.

6.5.1.2 Simulation

In general, a simulation needs enough runs (simulation time) to ensure a statistical relevance of the results and to cover the parameter space of all degrees of freedom (e.g. the jitter of the send requests, the sending arrangement of the frames).

Code simulation Code simulators simulate the execution of given binary code for a certain processor. A wide variety of code simulators exist. Simple instruction set simulators provide very limited information about the execution time whereas complex simulators consider also pipeline- and cache-effects. To achieve reliable WCET information from a code simulator, it has to be embedded into a test environment which actually causes the worst case to be simulated.

Scheduling simulation Scheduling simulators provide similar functionality as the scheduling analysis. Instead of calculating the results, they simulate run time behavior. The observed timing information and generated traces are the main output. If the worst case scenarios are simulated, the observed response times will equal the WCRTs. Some simulators allow Task definitions in C language so that complex applications models are supported while offering a specification language well known to automotive engineers.

Network simulation Network simulation is the technique of predicting the actual timing of a bus segment or network of segments based on models of the actual configurations. These models are typically derived from the same design data is needed to configure the connected ECUs (e.g. AUTOSAR System Extracts). The main information is the size of each frame/package (e.g. based on the size of the contained signals/parameters and the protocol header), the frame's transfer properties (i.e. its cycle time or debounce time of external triggers), and of course the transmission speed of the network. The network simulator is specific to a particular network protocol and will typically create random traffic within the bounds specified by the model data and unroll specific schedules. These schedules can be investigated with respect to resulting frame response times, network load and so on. As another kind of network simulation the remaining bus simulation is not a timing specific method, but many timing issues like arbitration latency, jitter, high load behaviour, etc. can be carried out on a real physical layer for experimental purposes. As a result other simulation methods can be verified.

Processor-In-The-Loop Simulation (PIL) is used to determine timing properties like SPECIFIC PROPERTY Execution Time or GENERIC PROPERTY Load of a specific software system. Therefore the compiled software will be executed in the embedded target processor on an evaluation board, a prototype hardware or the actual ECU. In order to be able to execute the software correctly the required run-time environment will be simulated. The simulation platform stimulates and calls the software under investigation. During the execution the required output data is captured. The output data is analyzed to derive the required timing properties. To carry out a PIL the analyzable executable (e.g. elf file) and input vectors for stimulation must be provided. The results of the PIL simulation should be validated with the results from alternative methods described in Definition, Description and Classification of Timing Methods.

The input stimuli vector which will be used for the PIL needs to stimulate the software in a way that the highest physically possible code coverage is reached. The quality of the input stimuli vector shall be shown in a separate "input stimuli vector acceptance test" which proofs an appropriate coverage. The accuracy of the result strongly depends on the quality of the input stimuli vector.

The tracing solution which captures the output data must have the capability to measure the execution time between defined profiling points. Profiling points define the start and end point for the measurement.

Referencing Use-cases	• ECU use-case "Collect Timing Information of a SW-C" • ECU use-case "Validate Timing after SW-C integration" • ECU use-case "Validation of Timing" • ECU use-case "Optimize Timing of an ECU" • ECU use-case "Optimize Scheduling" • ECU use-case "Optimize Code" • ECU use-case "Compare Timing Properties"
Referencing Timing Properties	• SPECIFIC PROPERTY Execution Time • GENERIC PROPERTY Load • GENERIC PROPERTY Response Time

Table 6.43: Relation

Discrete-Event-Simulation (DES) is used to simulate the dynamic behavior of the system. It models the operation of a system as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. The method can be applied whenever a timing model of the system is available. The results of this method are timing properties of the system. A Timing Model of the system must be available and the accuracy of the result strongly depends on the quality of the input model.

Referencing Use-case	• ECU use-case "Collect Timing Information of a SW-C" • ECU use-case "Validate Timing after SW-C integration" • ECU use-case "Create Timing Model of the entire ECU" • ECU use-case "Validation of Timing" • ECU use-case "Optimize Scheduling"
Referencing Timing Properties	• GENERIC PROPERTY Response Time • GENERIC PROPERTY Load • SPECIFIC PROPERTY Execution Time

Table 6.44: Relation

Hardware-In-The-Loop Simulation (HIL) can be used to determine timing properties like GENERIC PROPERTY Response Time or GENERIC PROPERTY Load of a specific ECU software.

To carry out a HIL simulation the software must be integrated to the actual ECU. The ECU is connected to a so called Hardware-In-The-Loop simulator which is able to simulate car's environment that is required for the proper functionality of the ECU. During the simulation the desired output data is captured. The output data is analyzed to derive the required timing properties.

The input stimuli vector needs to stimulate the ECU software in a way that the highest physically possible code coverage is reached. The accuracy of the result strongly depends on the quality of the input stimuli vector.

The tracing solution must have the capability to measure the execution time between defined profiling points. Profiling points define the start and end point for the measurement.

Referencing Use-case	• ECU use-case "Collect Timing Information of a SW-C" • ECU use-case "Validate Timing after SW-C integration" • ECU use-case "Validation of Timing"
Referencing Timing Properties	• GENERIC PROPERTY Response Time • GENERIC PROPERTY Load • SPECIFIC PROPERTY Execution Time

Table 6.45: Relation

6.5.1.3 Measurement and Tracing

Measurement on ECU level Timing measurement is often based on hook routines which are invoked by the RTOS. The real system is analyzed and the observed timing information is provided.

Measurement on Network level The timing measurement is done by special hardware connected to the hardware of the real network. Depending on the protocol and the applied measurement device the time stamp is imprinted at different point in time during the transmission of the relevant <schedulable>. The accuracy is given by the tracing device and shall fulfill the sampling theorem.

Tracing observes the real system. Tracing means persistent recording of measurement data streams. This can be recording of discrete events or sampled and quantized data from time contiguous sources in combination with a time stamp. For dedicated events, time stamps together with event information is placed in a trace buffer. The selection of events can be very fine grained like for flow traces which allow reconstructing the execution of each machine instruction or coarse grained like when tracing scheduling related events only. Tracing can base on instrumentation (i.e. software modification) or on special tracing hardware. Traces can be visualized and analyzed offline, e.g. for debugging purposes. Different kinds of timing information can be extracted from a trace. Sometimes an implicit protocol overhead has to be included for the correct computation (e.g. stuff bits for load computation on CAN).

6.5.1.4 Determination of the Comparability of the Different Methods

Comparing analysis on one hand and simulation/measurement on the other hand the loads shall be coincident in the long-time limit (under identical boundary conditions).

The difference vanishes if all parameters are chosen in the same manner. In general, the simulation and the observation yield an optimistic approximation in the same manner depending on the sample/probe size (measurement/simulation time).

In order to compare the results of different methods (especially simulation/analysis and measurement) a check that all <schedulables> are contained in the output is highly recommended.

6.5.2 Summary of regarded Methods

NW/ECU	Group	Field	Name
Generic	Load	Analysis, Simulation, Measurement	GENERIC METHOD Determine Load
NW	Load	Analysis, Simulation, Measurement	SPECIFIC METHOD Determine Load (CAN)
Generic	Latency	Analysis, Simulation, Measurement	GENERIC METHOD Determine Latency

Table 6.46: Summary of regarded Methods

6.5.3 GENERIC METHOD Determine Load

6.5.3.1 Scope and Application

Description	The method yields the load (distribution) over a defined time interval.
Reasoning	The method supports the estimation of the resource needs in ECUs and gateways and of the network, respectively.

Table 6.47: Scope and Application

6.5.3.2 Classification

System	NW / ECU
Applied Protocol	CAN / FlexRay / OSEK / AUTOSAR etc.
Approach	Analysis / Simulation / Measurement

Table 6.48: Classification

6.5.3.3 Relation

Requirements	Interface input and boundary conditions (see Table 6.50)
Process Steps	The method shall be applied during the following process steps: • Verification of a software implementation / of data definition and of the configuration of communication networks • Requirement analysis for further development • Resource optimization during development phase
Referencing Use-cases	• NW use-case "Integration of new communication" on page 57 • NW use-case "Design and configuration of a new network" on page 60 • NW use-case "Remapping of an existing communication link" on page 63 • ECU use-case "Collect Timing Information of a SW-C" on page 37 • ECU use-case "Validate Timing after SW-C integration" on page 40 • ECU use-case "Validation of Timing" on page 41 • ECU use-case "Optimize Scheduling" on page 47
(Pre) Timing Property	Depending on implementation this method requires the timing properties transmission time and/or execution time of all <Schedulables> on the considered <Resource> (e.g. SPECIFIC PROPERTY Transmission Time (CAN) and SPECIFIC PROPERTY Execution Time)
Belonging Post Timing Property	GENERIC PROPERTY Load

Table 6.49: Relation

6.5.3.4 Interface

Input	The method requires parameters such as: • <Schedulables> (e.g. tasks/frame/PDUs) with their overall times (transmission time, execution time), their activation pattern (e.g. periodic/cyclic, sporadic) and potentially other parameters (e.g. stuff-bits in case of CAN Bus communication) • Transmission/execution speed of the regarded <single resource> (e.g. CAN bus speed or processor speed) • Model of the spontaneous occurrence of <schedulable> (e.g.event-triggered frames) / approximation of the occurrence of the spontaneous events • Scheduling/priority rules on the <resource>

Boundary condition, Settings and Variants, Precondition	• Environmental states (like driving states)
Output	The result of this method is the load on a <resource> (NW/ECU) captured by the timing property GENERIC PROPERTY Load.

Table 6.50: Interface

6.5.3.5 Implementation

The implementation of the method for deriving the load of a network or of an ECU depends on the considered approach, namely analysis, simulation or measurement. Implementation details can be found in the corresponding specific methods.

6.5.4 SPECIFIC METHOD Determine Load (CAN)

6.5.4.1 Scope and Application

Brief Description	The method yields the load (distribution) on a CAN bus over a defined time interval.
Reasoning	The method supports the determination of the resource needs of a CAN network.

Table 6.51: Scope and Application

6.5.4.2 Classification

System	NW
Applied Protocol	CAN
Approach	Analysis / Simulation / Measurement

Table 6.52: Classification

6.5.4.3 Relation

Requirements	Interface input (see Table 6.54)
Process Steps	The method shall be applied during the following process steps: • Verification of data definition and of the configuration of a CAN bus • Requirement analysis for further development • Resource optimization during development phase

Referencing Use-cases	• NW use-case "Integration of new communication" on page 57 • NW use-case "Design and configuration of a new network" on page 60 • NW use-case "Remapping of an existing communication link" on page 63
(Pre) Timing Property	The method requires the SPECIFIC PROPERTY Transmission Time (CAN) of all frames on the considered CAN bus.
Belonging Post Timing Property	SPECIFIC PROPERTY Load (CAN)

Table 6.53: Relation

6.5.4.4 Interface

Input	The method requires the following CAN defined in 6.8 and 6.9.
Output	The result of this method is the load on a CAN bus captured by the SPECIFIC PROPERTY Load (CAN).

Table 6.54: Interface

6.5.4.5 Limitation in Application

At the moment, there is no established treatment for the spontaneous occurrence of event-triggered frames. Therefore, a unified model or activation pattern for the spontaneous occurrence has to be applied in order to achieve comparable results between different configurations.

A general treatment for the calculation of stuff-bits is missed. Therefore, different assumptions regarding the number of stuff bits shall be considered, i.e. a minimal, an average and a maximal number of stuff-bits.

6.5.4.6 Implementation

6.5.4.6.1 Analysis

The method for deriving the bus load for a CAN bus by analysis is based on mathematical formulas. These formulas can be implemented in a tool which supports the import of input parameters, the calculation of the load values and the export of the results.

The method has to enable the calculation of optimistic (best-case), average and pessimistic (worst-case) bus load values. For that purpose, different assumptions regarding the number of stuff-bits shall be implemented, i.e. a minimum number (for the optimistic approach), an average number, and a maximum number (for the pessimistic approach). Furthermore, different models of the event-triggered frame activation patterns shall be supported. The derivation of the load by analysis takes into the consideration the cyclic events with their periods and the spontaneous events with an event model. For example, the spontaneous events can be modeled with their debounce times as a "cycle" or with their maximum occurrence rate. Depending of the pessimistic or optimistic approach the calculation can estimate the upper bound with the lower limit of the period or with a specified period for the latter one.

The formula to calculate the bus load includes the pessimistic/optimistic approach depending on estimation of the stuff-bits for the analysis (see the formula for the stuff bits below, for CAN frames with 29-Bit Identifier there are deviations). The CAN parameters are given in 6.7.

$$t_{frame} = t_{\text{stuff bits(frame)}} + (47 + 8 * payloadlength(frame)[Byte]) * \tau_{Bit} \tag{6.1}$$

$$L(t_{frame}, t_{cycle(frame)}, payloadlength(frame)) = \sum_{frame} \frac{t_{frame}}{t_{cycle(frame)}} \tag{6.2}$$

Whereas payload length (in Byte) is the length of the data part of the CAN frame, t_{bit} is the time for the transmission of one bit, t_{cycle} is the specified period.

This approach estimates the bus load generated by the periodic messages on a bus during an infinitely long time window (t_{window} is infinity, the present point in time t does not play any role). The time for a frame is maximized due to a conclusion of all possible stuff bits. The event-triggered frames are neglected.

For CAN, different assumptions for stuff bits shall be implemented (minimal, average, maximal). Depending on the implemented approach, the calculation shall include a minimum (optimistic approach), an average or a maximum (pessimistic approach) number of stuff-bits. For each frame the following calculation formula for the maximal stuff-bit time shall be used. The average number of stuff-bit time can be derived by dividing by 2.

$$t_{\text{stuff bits(frame)}} = \left\lfloor \frac{34 + 8 * payloadlength(frame)[Byte]}{4} \right\rfloor * \tau_{Bit} \tag{6.3}$$

6.5.4.6.2 Simulation

Every frame is simulated with its individual activation pattern (periodic, event triggered or mixed activation). Furthermore even for event triggered frames, different models for their activation patterns shall be supported. Different payloads may lead to different numbers of stuff bits which have to be considered for the computation of the frame time. In the simulation all frames try to access the network at their trigger points in time, but only the frame with the highest priority (lowest ID) gains the access to the bus. Regarding a temporal averaging interval t_{window} the bus load is given as a ratio of the time for the sending of all frames to this interval:

$$L(t_{frame}, t_{window}, t) = \sum_{frame \in t_{window}} \frac{t_{frame}}{t_{window}} \tag{6.4}$$

where t_{frame} is the time for each individual frame.

6.5.4.6.3 Measurement

The formula to calculate the bus load for CAN from a measurement is equal to the formula of the simulation and given by:

$$L(t_{frame}, t_{window}, t) = \sum_{frame \in t_{window}} \frac{t_{frame}}{t_{window}} \tag{6.5}$$

t_{frame} is again the time for each individual frame. The result is strongly dependent on the averaging (measurement) interval t_{Window}. In the short time the limes of the load can reach 100%. Important is to include the stuff bit overhead for the correct computation of the frame time and therefor of the load.

6.5.5 GENERIC METHOD Determine Latency

6.5.5.1 Scope and Application

Brief Description	The method yields the latency of <schedulables> when executed on <resources>.
Reasoning	The method supports the estimation of the resource needs in ECUs and gateways and of the network, respectively.

Table 6.55: Scope and Application

6.5.5.2 Classification

System	ECU / Network
Applied Protocol	CAN / FlexRay / OSEK / AUTOSAR etc.
Approach	Analysis / Simulation / Measurement

Table 6.56: Classification

6.5.5.3 Relation

Requirements	Interface input, see Table 6.58.

Process Steps	The method shall be applied during the following process steps: • Verification of an software implementation / of data definition and of the configuration of communication networks • Requirement analysis for further development • Resource optimization during development phase
Referencing Use-cases	• NW use-case "Integration of new communication" on page 57 • NW use-case "Design and configuration of a new network" on page 60 • NW use-case "Remapping of an existing communication link" on page 63 • ECU use-case "Validation of Timing" on page 41 • ECU use-case "Optimize Timing of an ECU" on page 45 • ECU use-case "Optimize Scheduling" on page 47 • ECU use-case "Verify Timing Model(s)" on page 50 • ECU use-case "Compare Timing Properties" on page 52
(Pre) Property	Depending on implementation this method requires the timing properties transmission time and/or execution time of all <Schedulables> on the considered <Resource> (e.g. SPECIFIC PROPERTY Transmission Time (CAN) and SPECIFIC PROPERTY Execution Time).
Belonging Post Property	GENERIC PROPERTY Latency

Table 6.57: Relation

6.5.5.4 Interface

Input	The method requires parameters such as: • Implementation of the software, analyzable executable (e.g. elf file), input vectors for stimulation a.s.o. • <Schedulables> (e.g. tasks/frame/PDUs) with their overall times (transmission time, execution time), their activation pattern (e.g. periodic/cyclic, sporadic) and other parameters (e.g. stuff-bits in case of CAN communication) • Scheduling/priority rules (e.g. preemptive, non-preemptive, mixed-preemptive) on the <resource> • Transmission/execution speed of the regarded <resource> (e.g. CAN bus, processor speed) • Model of the spontaneous occurrence of <schedulables> (e.g. event triggered frames) / Approximation of the occurrence of the spontaneous events
Boundary condition, Settings and Variants, Precondition	• Environmental states (like driving states)
Output	The result of this method are timing properties of type latency GENERIC PROPERTY Latency for all <schedulables> (frames/-tasks) on a <resource> (NW/ECU).

Table 6.58: Interface

6.5.5.5 Implementation

The implementation of the method for deriving the latencies of <schedulables> on <resource> (i.e. networks or ECUs) depends on the considered approach, namely analysis, simulation or measurement. Implementation details can be found in the corresponding specific methods.

6.5.6 SPECIFIC METHOD Determine Response Time (CAN)

6.5.6.1 Scope and Application

Brief Description	The method yields the latency (response time) of a frame when transmitted on a CAN bus.
Reasoning	The method supports the determination of the resource needs of a CAN bus.

Table 6.59: Scope and Application

6.5.6.2 Classification

System	Network
Applied Protocol	CAN
Approach	Analysis / Simulation / Measurement

Table 6.60: Classification

6.5.6.3 Relation

Requirements	Interface input, see Table 6.62.
Process Steps	The method shall be applied during the following process steps: • Verification of data definition and of the configuration of the CAN bus • Requirement analysis for further development • Resource optimization during development phase
Referencing Use-cases	• NW use-case "Integration of new communication" on page 57 • NW use-case "Design and configuration of a new network" on page 60 • NW use-case "Remapping of an existing communication link" on page 63
(Pre) Property	Depending on implementation this method requires the timing properties transmission time of all frames on the considered CAN bus (SPECIFIC PROPERTY Transmission Time (CAN).)

Belonging Post Property	GENERIC PROPERTY Latency

Table 6.61: Relation

6.5.6.4 Interface

Input	The method requires parameters such as defined in 6.7, 6.8 and 6.9, i.e.: • The length / transmission time of all frames, their activation pattern (e.g. periodic/cyclic, sporadic) and stuff-bits • Execution speed of the CAN bus • Model of the spontaneous occurrence of event triggered frames
Boundary condition, Settings and Variants, Precondition	• Environmental states (like driving states)
Output	The result of this method is the response time of an individual frame on a CAN bus captured by the SPECIFIC PROPERTY Response Time (CAN).

Table 6.62: Interface

6.5.6.5 Limitation in Application

At the moment, there is no established treatment for the spontaneous occurrence of event-triggered frames. Therefore, a unified model or activation pattern for the spontaneous occurrence has to be applied in order to achieve comparable results between different configurations and between the applied approaches i.e. analysis/simulation/measurement.

6.5.6.6 Implementation

The implementation of the method for deriving the response time for CAN depends on the considered approach, namely analysis, simulation or measurement.

6.5.6.6.1 Analysis

The method for deriving response times for a CAN bus by analysis is based on mathematical formulas. These formulas can be implemented in a tool which supports the import of input parameters, the calculation of the individual frame response times and the export of the results.

The method has to enable the calculation of optimistic (best-case), average and pessimistic (worst-case) response time values. For that purpose, different assumptions regarding the number of stuff-bits shall be implemented, i.e. a minimum number (for the optimistic approach), an average number, and a maximum number (for the pessimistic approach). Furthermore, different models of the event-triggered frame activation patterns shall be supported. The derivation of the response times by analysis takes into the consideration the cyclic events with their periods and the spontaneous events with an event model. For example, the spontaneous events can be modeled with their debounce times as a "cycle" or with their maximum occurrence rate. Depending of the pessimistic or optimistic approach the calculation can estimate the upper bound with the lower limit of the period or with a specified period for the latter one.

Typical model elementes required for deriving response times for CAN buses by analysis are: (i) the underlying scheduling policy, for CAN buses this being SPNP (Static Priority Non-Preemptive), (ii) for each CAN frame, the priority given by the CAN frame identifier, the frame length (see 6.7 , 6.8) and the activation pattern (see 6.10) and (iii) the CAN bus speed (e.g. 100kBaud)(see table 6.8).

Based on these elements a formal analysis method, as for example presented in [14], computes response times for frames transmitted on CAN buses.

6.5.6.6.2 Simulation

Every frame will simulate with its activation (periodic, event triggered or mixed). In the simulation all frames try to access the network at their trigger points in time. If the simulated network is occupied by another frame the frame in question is delayed at least as long the virtual occupation lasts. Further, the blocking time is so long as the frame in question has a lower priority than all other frames tried to transmit at the same time. The response time is given by the difference of the point in time of the completed transmission and of the point in time of the send request.

6.5.6.6.3 Measurement

The measurement of the response time of the individual frame is only possible if the actual point in time for the send request is known. Thus, a correlated measurement with a common time base of the internal processes inside the ECU and on the network is necessary. One gets a distribution of the response time for each frame.

6.5.6.6.4 General remarks

The analysis, the simulation and the measurement should be implemented in a similar way. All boundary condition shall be revealed. Different algorithms can be applied as long as the results are identical under identical conditions. Any approximation shall be signalized and the parameter for the cut-off shall be open.

The frames shall be implemented with different deviation from the specified period in case of cyclic activation. Different possibilities for modeling event-triggered activation patterns shall be supported. For the response time analysis, one has to take into consideration the cyclic events with their periods and the spontaneous events with an event model. E.g. the spontaneous events can be modeled with their debounce times as a "cycle" or with their maximum occurrence rate. Depending of the pessimistic or optimistic approach the calculation of the response time can estimate the upper bound with the lower limit of the period or with a specified period for the latter one.

6.5.6.7 Determination of the Comparability of the Different Methods

Comparing analysis on one hand and simulation/measurement on the other hand the values of simulation/measurement for the response time shall be approach the analysis results in the long-time limit (under identical boundary conditions). Depending on the analysis method the difference should be small. In general, the simulation and the observation yield an optimistic approximation in the same manner depending on the sample/probe size (measurement/simulation time).

In order to compare the results of different methods (especially simulation/analysis and measurement) a check that all frames are contained in the output is highly recommended.

A History of Constraints and Specification Items

A.1 Constraint History of this Document related to AUTOSAR R4.1.3

A.1.1 Changed Constraints in R4.1.3

No constraints were changed in this release.

A.1.2 Added Constraints in R4.1.3

No constraints were added in this release.

A.1.3 Deleted Constraints in R4.1.3

No constraints were deleted in this release.

A.2 Specification Items History of this Document related to AUTOSAR R4.1.3

A.2.1 Changed Specification Items in R4.1.3

No specification items were changed in this release.

A.2.2 Added Specification Items in R4.1.3

No specification items were added in this release.

A.2.3 Deleted Specification Items in R4.1.3

No specification items were deleted in this release.

List of Figures

List of Tables

Index