Real-time Ethernet Residual Bus Simulation: A Model-Based Testing Approach for the Next-Generation In-Car Network

Florian Bartols, Till Steinbach, Franz Korf, Bettina Buth, Thomas C. Schmidt
HAW-Hamburg, Department Informatik
Berliner Tor 7, D-20099 Hamburg, Germany
{florian.bartols, till.steinbach, korf, buth, schmidt}@informatik.haw-hamburg.de

ABSTRACT

The increasing complexity of automotive networks, their challenging timing constraints and their high bandwidth demands require new concepts for future in-car communication. Real-time Ethernet is meant to be a suitable candidate for the next-generation in-car interconnection. However, model-based testing capabilities must be available as well. Applications must be validated prior the first assembly, due to the distributed development process. Methods like *residual bus simulation* are of particular interest to allow for testing systems in early development stages by emulating unfinished or not available parts of the system.

In this paper, we present a methodology and a feasibility study of a residual bus simulation in automotive real-time Ethernet systems. The challenges of applying this testing method in real-time Ethernet based networks with parallel packet transmission are outlined and compared to today's automotive bus system simulation approaches. Furthermore, the combination of different model-based testing techniques, that are not used in state-of-the-art commercial tools, are applied for the validation of non-functional timing requirements. An extension to an existing abstract test case model is proposed, which allows modelling temporal attributes. It is simultaneously used as simulation model to drive the residual bus simulation. We demonstrate the approach's feasibility by implementing a prototype residual bus simulator for real-time Ethernet networks and applying it to an example application.

Categories and Subject Descriptors

B.4.5 [**Input/Output and Data Communications**]: Reliability and Fault Tolerance; D.2.5 [**Software Engineering**]: Testing and Debugging

General Terms

Design, Verification, Experimentation

RTNS '2014 , October 08 - 10 2014, Versailles, France
.

Keywords

Real-time Ethernet, TTEthernet, Residual Bus Simulation, Model-Based Testing, Validation, Modeling, UML-MARTE

1. INTRODUCTION

Today's cars are complex, mixed critical [20], distributed systems with more than 70 electronic control units (ECUs). These units are responsible for information and entertainment systems, e. g. radio and navigation or safety critical functions, such as emergency breaking or airbag systems.

The requirements for these systems include aspects such as bandwidth, reliable data transmission and real-time behavior in different combinations. While safety critical functions have high demands on real-time characteristics, entertainment components require high bandwidth capabilities. Due to these different requirements, the in-car network infrastructure became heavily heterogeneous in the past. For each kind of application class, a dedicated network system for on-board communication has been utilized. As a consequence, the topology has become more and more complex and the distributed applications are hardly manageable and maintainable nowadays.

Next-generation driver assistance functions relying e. g. on cameras and radar systems will increase the challenges of future in-car networks. These applications both have high real-time and bandwidth demands, since multiple data streams need to be merged and processed in real-time. State-of-the-art automotive bus networks like CAN [7] or FlexRay [5] already operate on their bandwidth limits or cannot fulfill the required rigid real-time requirements (MOST [9]) for safety critical applications. A possible solution for these challenges is the application of Ethernet-based networks for in-car communication [19]. Real-time Ethernet (RT Ethernet) is a suitable candidate for next-generation automotive on-board communication to overcome the issues imposed by new driver assistance functions. It provides reliable real-time data transmission while supplying sufficient bandwidth. Furthermore, the complexity of the network infrastructure can be reduced. This enables the integration of additional applications with reduced effort.

The utilization of software controlled automotive applications is rapidly increasing. It is the key driver for new innovations in the automotive industry. Features implemented in software increased in the last years from 20 % to 80 %. Forecasts claim that in this decade 90 % of automotive functions will be realized using software [2]. This fact directly influences the costs during the development process of a new in-car application. To minimize these costs, the

application has to be tested in early development stages. The later faults surface, the more expensive is the resolution. The distribution of functions over several ECUs and the distributed development process in the automotive industry [14] makes this goal even more difficult. In general, the Original-Equipment-Manufacturer (OEM) defines the application and its requirements along with its features in specifications and models, while external suppliers provide the implementation. Therefore, the suppliers have to test their developed subsystem prior to the first system integration. Unavailable or unfinished nodes that are essential for the application can make testing in the original operation environment impossible. Instead, testing frameworks such as residual bus simulations are applied by the suppliers. A residual bus simulator emulates the missing nodes and their behavior inside the distributed application to enable the testing procedure.

In this paper a methodology and a feasibility study for residual bus simulation of RT Ethernet components is presented, that is used to validate functional and non-functional timing requirements in distributed automotive applications. By applying a residual bus simulation for RT Ethernet based applications, we demonstrate the feasibility of our presented approach. It is successfully utilized to validate functional and non-functional requirements of an example application in a prototype next-generation in-car network.

The remainder of this paper is organized as follows: Section 2 introduces background information for the RT Ethernet residual bus simulation and presents related work. In Section 3 challenges of a real-time Ethernet based residual bus simulation and the differences to common automotive bus based residual bus simulations are described. Section 4 presents graphical methods to model non-functional timing requirements in distributed applications and furthermore introduces an abstract test-case model to validate the modeled requirements. The application of the proposed residual bus simulation approach is presented in Section 5. Finally, Section 6 concludes and gives an outlook on future work.

2. BACKGROUND & RELATED WORK

Residual bus simulation is the state of the art procedure to test distributed automotive systems in early development stages. It belongs to the *model-based testing (MBT)* methodology [15], where test cases are strictly inherited from models and executed in different development stages. In this section an overview of the utilized modeling language and an example of a generic residual bus simulation is given. Furthermore, TTEthernet, as an implementation of real-time Ethernet used in this paper, is presented.

2.1 Real-time Ethernet for Automotive Applications

The current in-car network architecture is a complex and inhomogeneous system with many different network protocols and systems. While the continuous increase of bandwidth consumption and number of transmitted messages will soon push today's infrastructure to its limits, new concepts for interconnecting nodes need to be investigated. New applications that rely on the current in-car network became difficult to implement, since its complexity is hardly manageable and maintainable [18].

Real-time Ethernet protocols like Time-Triggered Ethernet or IEEE AVB Ethernet can pave the path for new automotive applications, since it provides real-time characteristics and high bandwidth availability [19]. In this work, *Time-Triggered Ethernet (TTEthernet)* [16] is utilized as RT Ethernet protocol. It was developed by TTTech in cooperation with Honeywell as a fault-tolerant network protocol for safety-critical applications. TTEthernet allows for transmitting time-critical and non-critical messages on the same physical infrastructure, by defining three different message classes with different transmission characteristics:

Time-Triggered (TT) messages have the highest transmission priority and provide the strictest timing guarantees with low jitter. The transmission is accomplished in time-triggered manner on each participant and is used for periodically transmitted real-time data.

Rate-Constrained (RC) messages are transmitted in a bandwidth limiting manner and conform to the *ARINC-664 (AFDX)* [1] protocol specification. In contrast to TT-messages, RC-messages are used for asynchronous event-based critical messages with less rigid timing demands.

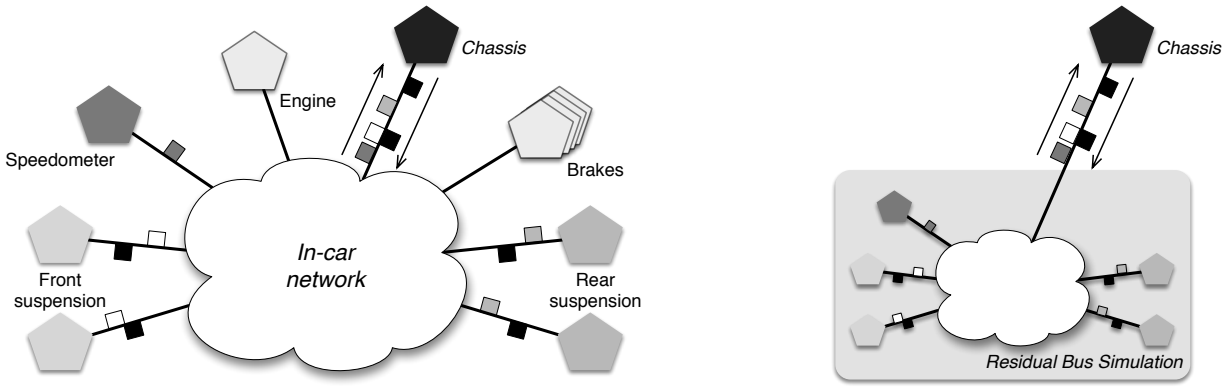
Best-Effort (BE) messages have the lowest priority and use the remaining bandwidth. This class is applied for best-effort communication without transmission guarantee.

Both, TT- and RC-messages are transmitted on statically configured network routes, by using the principle of *virtual links (Vlinks)*. A Vlink represents a virtual network path from one sender to many receivers, which allows for designing static group communication. In order to enable transmission of TT-messages with low latency variation, TTEthernet provides a transparent fault tolerant clock synchronization protocol to establish a globally synchronized time base on each network participant. For synchronization, TTEthernet defines dedicated roles for each end system. In a first step, *synchronization masters (SM)* initialize the synchronization process by transferring a special synchronization message containing their local time. These messages are received by a *compression master (CM)* which calculates an overall average of all time bases. In the second step, it sends a synchronization message containing the average time base back to the SMs and all remaining participants (*synchronization clients (SC)*). Afterwards, all nodes synchronize their local clock to the calculated common time base.

2.2 Residual Bus Simulation in the Context of Model-Based Testing

Current in-car applications are complex systems consisting of discrete and continuous subsystems, that interact which each other. To handle the system complexity during the development, suitable models are utilized in the automotive industry. The application, its components and behavior, and the requirements are designed as related models. To model functional and non-functional requirements in generic real-time and embedded systems, the profile *Modeling and Analysis for Real-time and Embedded Systems (MARTE)* [11] of the *Unified Modeling Language (UML)* [12] was developed. In contrast to the classic UML 2.0 specification, MARTE provides a more precise clock model and allows modeling timing requirements by using annotations in models. In our case, we use the MARTE-profile to model non-functional timing requirements in the communication between endsystems in real-time Ethernet systems.

Model-based testing is part of the model-based development methodology to validate the behavior of the developed system and is mainly applied for black box testing purposes.



(a) System executing a speed-dependent suspension control

(b) The residual bus simulation to validate the chassis controller

Figure 1: In-car network of the adaptive chassis system in comparison with the corresponding residual bus simulation

The idea of MBT is to simplify the complex task of generating test cases by systematically deriving them from models [23, 21]. In general, the generation can be classified in *online* or *offline* approaches. Online generated test cases are created during runtime and can be used to test the reactive behavior. In contrast, offline generated test cases are created before the actual execution and can be repeatedly executed with the same parameters. The models can either depict the complete developed system or a simplified abstract part which represents the system aspect to be tested. The systematically generated test cases allow for a better reusability, since the execution is defined for different testing platforms (i. e. *Model-in-the-Loop (MiL)*, *Software-in-the-Loop (SiL)*, *Processor-in-Loop (PiL)*, *Hardware-in-the-Loop (HiL)*), in order to validate the behavior in different development stages.

The residual bus simulation methodology is an approach to test distributed systems and can be categorized as an additional testing platform in the MBT context. In contrast to a pure software simulation, a residual bus simulation does not virtually simulate a complete system. During a residual bus simulation, simulated and physical systems are running side-by-side in the same environment. Only the missing nodes and their behaviors are simulated, while the physically available components of the system are tested.

Still, principles of software simulation need to be utilized to drive the actual residual bus simulator. In general, the physical system is represented as a *simulation model*, which is executed on a *simulator*. A residual bus simulator has to process data in real-time, since it must pretend to be a physical system. Therefore, the calculation of the simulation state is accomplished in a *discrete time-based* approach. In general, *discrete simulations* utilize an event list, which contains an event-actionpoint pair. The simulation time in discrete time-based simulations relies on a dedicated clock which constantly increments. Events are only executed, if the simulation time equals to the action-point of the event, as long as the event list is not empty. In contrast, *discrete event-based* simulations do not rely on a dedicated clock. They will immediately execute the next event and set the simulation time to the action-point of the execution. In general, discrete event-based simulations have a higher simulation throughput, since events can be immediately executed, however, the execution of events with high temporal precision has more priority than high simulation throughput. If events are executed too fast or too slow, the simulated environment does not conform to the physical system and thus, the *system-under-test (SUT)* cannot be properly tested with regard to its timing behavior.

The following small example will explain the application of a residual bus simulation in an automotive environment. Figure 1a presents an automotive network implementing an adaptive chassis system. It controls the front and rear suspension depending on the current velocity and acceleration to keep the car in balance. The chassis ECU collects the speed and suspension data and processes new suspension settings, which are applied by the suspension actuators afterwards. In addition to the speedometer and suspensions, the engine and brakes are connected to the same network but do not interact with the chassis control.

Figure 1b presents the application of the corresponding residual bus simulation to validate the behavior of the chassis controller during the development. In this example, the chassis controller is treated as *system-under-test (SUT)* while the simulator has to emulate the behavior of the suspension and the speedometer. The residual bus simulation does not contain the engine and brakes. In early development stages, nodes that do not interact with the SUT are not simulated in order to reduce the simulation complexity and simplify debugging characteristics. Since the SUT is connected to the simulator via the physical network interface, the simulator has to generate emulated data frames to pretend a physical system. Furthermore, the simulator has to completely support the utilized network technology. I. e. in time-triggered systems, messages are to be generated corresponding to the network configuration (e. g. schedule). Thus, the residual bus simulation configuration is separated in static and dynamic attributes. Static attributes describe the configuration of the network interface, while dynamic attributes are used to describe simulation behavior, which is modeled within the simulation model.

2.3 Related Work

The application of residual bus simulation to validate developed systems in early development stages is a typical use case in the model-based development strategy of automotive systems. Commercially developed residual bus simulators support the state-of-the-art automotive bus network technologies, but currently none of these support RT Ethernet as network system for control data. Mainly all simulators have the same test setup topology in common [22, 4], which

is based on two different components. A standard *workstation* generates the static and dynamic configuration and is used for the analysis of the performed test case. The *residual bus simulator platform* executes the simulation model in real-time and generates emulated network packages. This established setup will be used in our implementation of a residual bus simulation environment for RT Ethernet networks as well. It provides adequate simulation capabilities while the analysis does not influence the simulation.

In previous work, an implementation that interconnects the event-based simulation platform OMNet++ [13] with RT Ethernet networks was presented in [8]. This platform can be used to analyze RT Ethernet network protocol implementations in the OMNeT++ simulation environment. It relies on an architecture with two different hardware platforms. A microcontroller executes a RT Ethernet stack, which is responsible for the protocol conform reception and transmission of messages from and to the real network. The associated simulation model of the same RT Ethernet protocol runs on a standard workstation inside the OMNeT++ simulation environment. Both components are interconnected via dual-port memory to exchange data in both directions. A software implementation of TTEthernet for the microcontroller [10] was used as well as a TTEthernet simulation model [17] for OMNeT++. In order to send and receive time-triggered messages from the simulation to the real world and vice versa, the simulation environment was synchronized with the real network, by accessing the highly precise clock of the microcontroller. The residual bus simulator architecture presented in this paper uses the same architecture with two dedicated hardware platforms. A workstation runs the simulation environment and generates the packages to be transmitted. The microcontroller is used to send the packages in conformance with their transmission strategy (TT, RC or BE). In contrast to the previous work, the simulation environment does not build on OMNeT++. The residual bus simulation is utilized for validation, such that the simulator is responsible for generating emulated network frames instead of simulating attributes of a RT Ethernet protocol. Therefore, the environment implements a discrete time-based simulation approach, in order to generate emulated frames at the desired point-in-time.

In order to formally describe test cases for functional requirements, Skruch et al. have presented an abstract test case model [15]. The basic idea of abstract test cases is to utilize a model that is independent of the actual test environment so that the same test case can be executed on different test platforms. Their presented abstract test case model bases on the state space model, which allows to formally model the inputs and outputs, and the internal states as vectors of a SUT (see Section 4.3). An abstract test case defines the data of the inputs to be generated and the expected outputs, and the internal states of the SUT in dependency of the time. This approach was successfully utilized to validate functional requirements of an automotive control unit with a CAN interface, which conforms to a residual bus simulation. The same abstract test cases model is utilized in this work for the description of functional requirements as well. Since non-functional timing requirements are to be validated too, this model is extended with temporal attributes such as latency. Furthermore, it is going to be utilized as simulation model to allow for setting up the dynamic residual bus configuration.

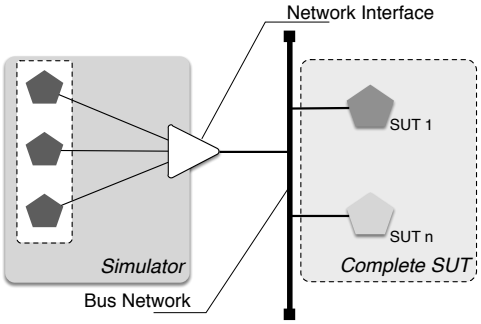


Figure 2: Cluster simulation setup for automotive network

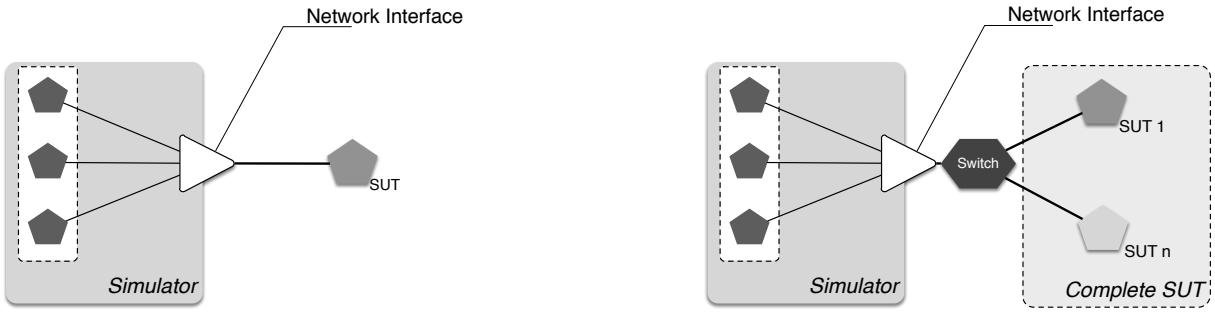
3. CHALLENGES FOR A REAL-TIME ETHERNET CLUSTER SIMULATION

Commercial residual bus simulators already support the state of the art automotive bus network technologies. Since the utilization of RT Ethernet as an in-car network system is novel, a suitable simulation environment has different attributes and requirements to be fulfilled. In general, a residual bus simulation is used to validate functional requirements of a developed distributed system. In RT Ethernet networks, the application behavior strictly relies on the configured network attributes, so that non-functional timing requirements during transmission are essential as well. In particular, this affects TT- and RC-messages which have to be statically configured on each node. If a message does not meet the configured timing attributes, the messages will be delayed or worse, dropped by the network. Therefore, it is preferable that functional and non-functional requirements can be verified with the same test environment, which enhances error detection.

3.1 Message Classes and Synchronization

A residual bus simulator is connected via the physical communication interface to the SUT and generates emulated data packages. In order to validate the behavior of the SUT, the residual bus simulator has to completely support the utilized network protocol. In comparison to state of the art in-car network systems, TTEthernet defines three message classes with different attributes and priorities. Moreover, the synchronization procedure relies on dedicated messages that have to be processed by the residual bus simulation environment. Depending on the synchronization role of the SUT, the residual bus simulator has to perform the opposite role in order to establish a common time base, that allows synchronized transmission of time-triggered messages. Thus, the simulation is more complex than a residual bus simulation of state-of-the-art automotive bus network technologies.

In contrast to automotive bus network systems, RT Ethernet bases on switched Ethernet which provides a dedicated collision domain for each link between two network participants. A node can start a frame transmission at each point in time, so that network wide parallel data transmission is likely to happen. In automotive bus based networks, parallel data transmission without collisions is physically impossible, due to the shared media access. A switched collision (more precisely congestion) domain influences the setup for a residual bus simulation, especially if the SUT is composed of more than one node. In automotive residual bus simulations, the same test setup can be used independently of the number of nodes of the SUT, as depicted in Figure 2 [6].



(a) Directly connected residual bus simulation (b) Indirectly connected residual bus simulation

Figure 3: Directly and indirectly connected test setups in comparison

3.2 Test Setup Attributes

Due to parallel data transmission, the test setup for a residual bus simulation in RT Ethernet networks differs from the previously presented approach. Setups can be classified in *directly* and *indirectly* connected approaches and are depicted in Figure 3. In a directly connected setup (Fig. 3a), the simulator communicates directly with the SUT, while an indirectly connected approach (Fig. 3b) utilizes an additional RT Ethernet switch for the communication between multiple devices of the SUT and the simulator.

The directly connected approach is limited to one single SUT and can only be utilized for unit testing. Parallel data transmission is impossible, since only one unit is connected to the simulator. If for integration purposes more units are to be verified at the same time, the previously presented approach is infeasible. An additional RT Ethernet switch has to be utilized in order to connect all nodes of the SUT to the residual bus simulator. A directly connected approach with more than a single network interface is neglectable for residual bus simulation. If the subsystems-under-test have to communicate with each other, the simulator has to provide switching capabilities and the traffic has to be handled. This handling can result in a negative simulation behavior, if the routing has to be done in software.

The setup for multiple SUTs is composed of a simulator with one network interface and is depicted in Figure 3b. In this approach, different messages can be sent by different subsystems, at the same point-in-time. Parallel frame transmission can result in *scheduling* and *bandwidth conflicts* since different data paths at the switch are to be integrated. *Scheduling conflicts* occur if different TT-messages have to be sent by the simulator and received by different subsystems at the same point-in-time. A parallel transmission of messages is physically impossible with one interface, so that the simulator has to sequentially transmit the TT-messages. The switch is then responsible for the parallel transmission. Scheduling conflicts can occur too when the simulator has to receive different TT-messages at the same point-in-time. In this case, the switch needs to sequence the different messages. *Bandwidth conflicts* occur due to the integration of multiple connections to a single connection. If the accumulated bandwidth of all paths exceeds the available bandwidth on the connection to the simulator, a residual bus simulation cannot be properly performed.

The depicted conflicts in indirectly connected simulations can be resolved by the application of a simulator architecture with multiple network interfaces. This avoids the necessary integration of multiple to single connections. Furthermore, subsystems can communicate with each other via a switch and thus, do not influence the simulator, since this traffic is not passed through its interfaces.

The presented approach with an additional RT Ethernet switch influences the timing behavior of the transmitted messages as it induces additional forwarding delay. Thus, it cannot be utilized if non-functional timing requirements are to be verified. The forwarding delay distorts the timing behavior, especially if frames are to be integrated and sequenced at the switch. To validate non-functional timing requirements only a directly connected residual bus simulator can be utilized.

4. MODELLING WITH MARTE & ABSTRACT TEST CASE MODELS

In the context of model-based testing, test cases are derived by models that represent system requirements. The presented residual bus simulation approach is utilized to validate functional and non-functional timing requirements. In this paper, the MARTE profile for UML is applied to model functional as well as non-functional requirements. These models are utilized to manually derive abstract test cases for the validation of the modeled requirements.

4.1 Description of the Sample Use-case

A sample application and its requirements are described in the beginning of this section, in order to explain the utilization of the UML profile MARTE. The presented application serves as an example for the application of the residual bus simulation later on. The application is embedded into a prototype RT Ethernet in-car network, which is used by us to demonstrate and analyze the next-generation in-car network. It combines time-critical (steer-by-wire and light control) and non-critical (multimedia) applications, and is visualized in Figure 4. Both time-critical (TT- and RC-messages) as well as BE-messages are concurrently transferred and are forwarded via two dedicated switches. The focus is set towards the headlight control application.

The light control dashboard transmits new light control states to the controller of the headlight subsystem which acknowledges the changing light states. Furthermore, each headlight subsystem announces its current light status to the network. The dashboard system utilizes the announcements to display the current status to the user. In this paper, the following functional and non-functional timing requirements of a single headlight unit are exemplarily depicted.

Functional requirements:

1. The headlight controller must acknowledge every received valid light status by replying the status.

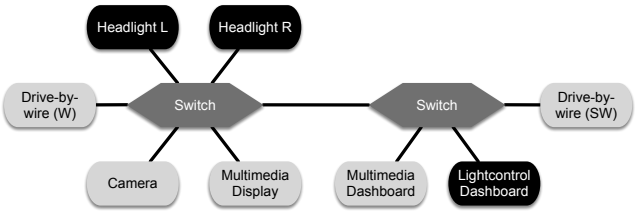


Figure 4: Distributed light control system in a prototype RT Ethernet in-car network

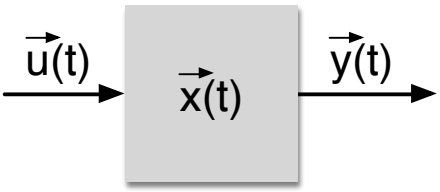


Figure 5: Graphical representation of a control system with multiple in- and outputs

2. The headlight controller must periodically announce its current light status to the network, so that further applications can use the state of the headlight.

Non-functional requirements:

1. The acknowledgement must arrive at a delay of 500 µs ± 50 µs
2. The transmission rate of the light status announcements must arrive at an interval of 5000 µs ± 5 µs

4.2 Modeling Non-Functional Timing Requirements with MARTE

In general, the UML profile MARTE is utilized to model and analyze real-time and embedded systems and builds on the wide range of supported UML models [3]. The profile extends UML with the *Value Specific Language (VSL)* that is used to annotate classic UML diagrams, and allows for describing different scale units, as well as mathematical expressions. Additionally, MARTE supports the modeling of temporal attributes by providing a detailed clock model. It is possible to use an ideal, synchronized clock, as well as developing realistic clocks with predefined drifts. Hence, the used clock model has to be referenced to point out the behavior of the clock. For modelling time spans or periods of time as non-functional timing requirements, the application of an ideal synchronized clock is appropriate. Realistic clocks, however, can be used to model distributed applications (i. e. clock synchronization procedures) so that differences on the local clock can be described. By using the VSL and a suitable clock model within classic UML sequence diagrams, we have modeled the presented functional and non-functional timing requirements.

Figure 6 depicts UML sequence diagrams, that apply the presented MARTE syntax. The diagrams describe functional and non-functional timing requirements of the presented light control system. The *dashboard* system transmits a new light status (`setNewLightStatus(status)`) to the light controller of a headlight system, which acknowledges the new light status by replying the same light status to the dashboard (`sendCurrentLightStatus(status)`). The procedure is presented in Figure 6a. In order to display the current light status, the light controller transmits the status in a fixed interval (see Figure 6b) and represents the repetition rate. The maximum reply delay is modeled within VSL and an associated clock model in Figure 6a. Since a time span is described as requirement, the diagram has to refer to the ideal clock model (`[MARTE_Library::TimedLibrary::idealClock]`). The span is defined via the points in time `messageReceived` and `replySent` which are represented with an additional `@`. The associated requirements are described within the `timedConstrained` annotation, where the maximum reply delay is modeled as the difference between `messageReceived` and `replySent` with the value 500 µs. Furthermore, an associated jitter (100 µs) of the previously defined difference is modeled. In this case jitter is defined as a window which allows defining the permitted variation of the reply delay so that values on the interval of 450 µs to 550 µs are valid. The requirements of the repetition rate are modeled in Figure 6b and are described via `timedContrained`, too. Again the time span is depicted as the difference of two points in time $\texttt{sendStatus}_\texttt{i}$ and is defined as 5000 µs. The associated jitter is defined with 10 µs so that the repetition rate is valid in the range of 4995 µs to 5005 µs.

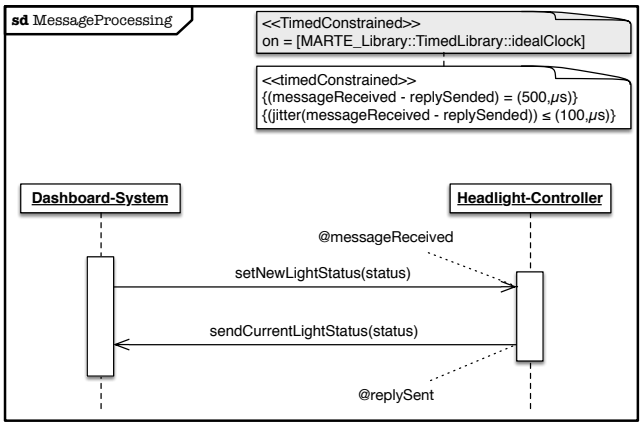
Functional and non-functional timing requirements for the sample headlight control application were modeled using MARTE as an extension to standard UML. Thus, associated test cases for the RT Ethernet residual bus simulation can be effectively derived from the developed sequence diagrams.

4.3 Extending the Abstract Test Case Model

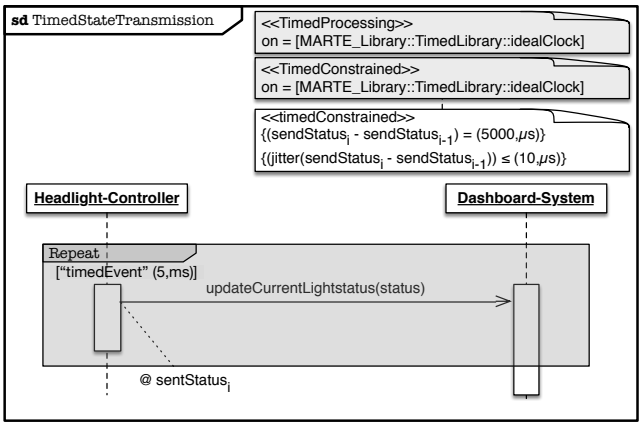
The taxonomy of model-based testing defines the application of the same test case in different test environments (i. e. MiL, SiL, PiL, HiL) [23]. A system is tested in different development stages, which enhances error detection by forcing failures to surface earlier and thus simplifying their resolution. For executing the same test case in different test environments, the test case model needs to provide a certain abstraction level. Information that is used by a dedicated test environment must be omitted in the test case model. Skruch et al. [15] have presented an abstract test case model which has been utilized to validate functional behavior of a CAN based automotive ECU.

The approach is based on the state space model that defines a control system with multiple inputs and multiple outputs. In general, state space models are utilized in control engineering to analyze dynamic systems. A system modeled as a state space model consists of a certain number of inputs (U) and outputs (Y), and internal states (X), as depicted in Figure 5. It is defined as vectors with a predefined length, where $u(t) \in \mathbb{I}^r$, $x(t) \in \mathbb{S}^n$ and $y(t) \in \mathbb{O}^m$. Here, $\mathbb{I}$ and $\mathbb{O}$ define the set of allowed input and output signals, data or messages, and $\mathbb{S}$ defines the set of possible states.

Since the inputs and outputs, and states are time-dependent, the associated test case model must be able to model a discrete time space. Concrete values to the inputs and outputs, and states must be assigned to dedicated points in time. The abstract test case model for functional requirements ATC_{FR} is either defined as a quadruple (Equ. 1) or as a triple (Equ. 2) if the internal states cannot be retrieved (i. e. black box testing).



(a) Requirements of the status reply feature



(b) Requirements of the time-triggered transmission

Figure 6: UML MARTE diagrams model functional and non-functional requirements

$$ATC_{FR} = (T, U, X, Y) \in (\mathbb{N}^{1\times k} \times \mathbb{I}^{r\times k} \times \mathbb{S}^{n\times k} \times \mathbb{O}^{m\times k}) \quad (1)$$

$$ATC_{FR} = (T, U, Y) \in (\mathbb{N}^{1\times k} \times \mathbb{I}^{r\times k} \times \mathbb{O}^{m\times k}) \quad (2)$$

In analogy to the state space model, U and Y represent the inputs and outputs, and X the internal state. Additionally, T describes the discrete points in time, at which specific values are located at the vectors. This model defines all required attributes to validate functional system requirements.

However, this model is insufficient to validate non-functional timing requirements such as the presented reply time of a system and the repetition rate of a message. It needs to be extended by specific attributes that allow for modeling the required time spans. Thus, *latency*, *latency jitter*, *rate* and *rate jitter* are added to the presented model. In this case, latency defines the maximum reply time of the SUT to react to a specific input. Rate defines the time span of two consecutive signals or messages of the same type that are generated by an output. The jitters are used to model the maximum allowed variation of both requirements. The internal state is immaterial in the case of residual bus simulation and can be neglected. The abstract test case model for non-functional requirements ATC_{NFR} is presented as 7-tuple in Equation 3.

$$ATC_{NFR} = (T, U, Y, L, R, \Delta_L, \Delta_R) \in (\mathbb{N}^{1\times k} \times \mathbb{I}^{r\times k} \times \mathbb{O}^{m\times k} \times \mathbb{R}^a \times \mathbb{R}^b \times \mathbb{R}^y \times \mathbb{R}^z) \quad (3)$$

In this model, the first three attributes are the same as previously presented. Furthermore, L depicts the latency, R the rate and Δ_L as well as Δ_R the variations. The latency requirement L contains a elements $l_a(u_g, y_h)$, where $g = 1, 2, \ldots r$, $h = 1, 2, \ldots m$ and $a \geq 1$. Thus, it is composed of an in- and output pair. The rate R contains b elements $r_b(y_h)$, where $h = 1, 2, \ldots m$ and $b \geq 1$ and is used on a specific output. The relation of the latency jitter Δ_L contains y elements: $j_{Ly}(l_p)$, $p = 1, 2, \ldots a$ and $y \geq 1$ and in analogy for the rate Δ_R: $j_{Rz}(r_q)$, $q = 1, 2, \ldots b$ and $z \geq 1$. In contrast to inputs and outputs, the timing requirements are independent of the current time and are valid for the whole test case.

This model allows defining a specific value of an input and the associated expected output. Furthermore, time spans to inputs and output pairs, as well as output generation rates are defined of a SUT. Therefore, this model simultaneously defines functional and non-functional requirements and can be used for any distributed application that strictly relies on timing requirements. An abstract test case is expressed in the table notation in Table 1. This notation is further utilized in order to depict the executed test cases.

4.4 Utilization as Simulation Model

To apply the modeling work to the residual bus simulation, a suitable simulation model has to be executed on the simulator. In general, this is realized by dedicated models, that represent the physical system. The simulation behavior is either online processed during execution or offline generated in the run-up to simulation. In the context of residual bus simulation where requirements are to be verified during

T	t_0	t_1	$\ldots$	t_k
U	$u_1(t_0)$	$u_1(t_1)$	$\ldots$	$u_1(t_k)$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	$u_r(t_0)$	$u_r(t_1)$	$\ldots$	$u_r(t_k)$
Y	$y_1(t_0)$	$y_1(t_1)$	$\ldots$	$y_1(t_k)$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	$y_m(t_0)$	$y_m(t_1)$	$\ldots$	$y_m(t_k)$
L	$l_1(u_1, y_1)$			
	$\vdots$			
	$l_a(u_r, y_m)$			
R	$r_1(y_1)$			
	$\vdots$			
	$r_b(y_m)$			
Δ_L	$j_{L_1}(l_1)$			
	$\vdots$			
	$j_{L_y}(l_a)$			
Δ_R	$j_{R_1}(r_1)$			
	$\vdots$			
	$j_{R_z}(r_b)$			

Table 1: Abstract test case model in table notation

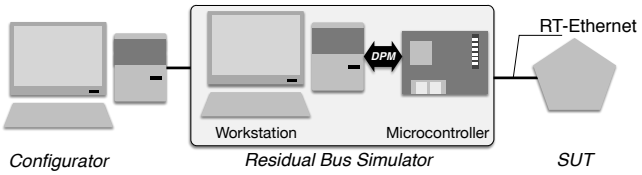


Figure 7: Implementation of the presented residual bus simulator architecture

simulation, offline generated simulation behavior is appropriate. The same test case with the same attributes can be executed several times to validate the implementation.

To model the behavior of the simulated subsystem the point in time of data generation at the input of the SUT needs to be known. This conforms to the time-based discrete simulation approach, where an event list is utilized containing an event–action-point-in-time pair. Our presented abstract test case model ATC_{NFR} provides this information using the time vector T and the time dependency of the inputs. Thus, the abstract test case model is an appropriate alternative to model simulation behavior as well.

With this abstract test case model, we can describe two aspects of a residual bus simulation. First, it can be used as a simulation model to design the simulation behavior, and second, to develop test cases for the validation of functional and non-functional requirements of a SUT.

5. APPLICATION OF THE RT ETHERNET RESIDUAL BUS SIMULATION

The presented approaches of a residual bus simulation and an abstract test case model are implemented on a prototype platform, to demonstrate the application of a RT Ethernet residual bus simulation. The previously presented requirements of the automotive prototype application are exemplarily verified.

5.1 Implementing a RT Ethernet Residual Bus Simulator

The implementation of the residual bus simulator is realized using the presented setup topology of common residual bus simulators with three dedicated components and is depicted in Figure 7. The configurator is used for the creation of the static network configuration of the simulator, the dynamic behavior by creating test cases, as well as the analyses of the results. The architecture is equivalent to the approach presented in [8] and is composed of two different hardware platforms. They are interconnected via a dual port memory (DPM) interface in order to allow fast data exchange in both directions. The workstation is running the discrete time-based simulation environment that generates the Ethernet frames as raw data and transfers the data to the microcontroller. The microcontroller executes the RT Ethernet software stack and transmits the generated frames to the SUT regarding its message class. In order to measure time spans, which is mandatory to validate non-functional timing requirements, the microcontroller adds timestamps to any received and transmitted frame.

To be able to analyze the results, transmitted as well as received frames are stored with their associated timestamp on the workstation. The workstation is responsible to create Ethernet trace files that can be analyzed after the test case was executed. The used microcontroller is composed of a dedicated high precision system clock, so that every frame can be equipped with a timestamp with nanosecond precision. Furthermore, the presented abstract test case model is implemented as a XML data model. It is processed by the simulation environment to be utilized as simulation model. Since a test case only contains abstract representations of the data to be generated, the simulation environment further has to convert the abstract data into an executable test case. Thus, the abstract test case is converted to the required event list by extracting the data of an input and the value of the time vector. This residual bus simulation architecture can be adequately utilized to validate functional, as well as non-functional timing requirements, which will be shown in the next section.

5.2 Applying a Residual Bus Simulation for Verification Purposes

The previously presented requirements were modeled by the application of UML-MARTE and were shown in the sequence diagrams (see Figure 6). Abstract test cases have been manually inherited by the utilization of the developed sequence diagrams. In order to validate the modeled requirements of the headlight controller, the residual bus simulator has to emulate the behavior of the dashboard system.

The test setup of the residual bus simulation is applied with a directly connected approach, since only this test setup allows validating functional and non-functional timing requirements at the same time. The induced forwarding delay when applying a indirectly connected residual bus simulation distorts the timing behavior of a transmitted frame. The static residual bus simulation configuration could be directly taken from the headlight controller, so that the simulator processes Ethernet frames that are normally generated and received by the dashboard controller. In this case, the residual bus simulator transmits the state change messages and receives acknowledge and current state announcement frames. The test case verifies the presented functional and non-functional requirements at the same time and is shown in Table 2. A headlight subsystem is composed of dimmed headlights and additional LEDs for the application of daytime running lights, which are adjustable in the brightness.

To validate the functional requirements, the SUT is reset with the `HL_OFF`-message at the beginning of the test. It is used to power off all lamps and transfer the headlight system into a defined state. At two seconds, the actual test starts and the LEDs are set to a brightness of 0 %. Afterwards, the LEDs are set to 100 % brightness. To demonstrate that messages with irregular values do not influence the headlights behavior, the brightness is set to 50 % and afterwards to 101 %, where the last value is irregular. The test is finished by setting a regular value to the headlights again. The non-functional requirements are modeled in the same test-case and are presented in the previously proposed way. The expected acknowledge latency is defined as 500 µs with an associated variation of 50 µs. The rate is 5000 µs with a variation of 10 µs.

The results of the first execution of the test case are presented with the expected and actual values in Table 2 as well. The non-functional timing requirements are fulfilled and are highlighted with a light gray background. The latency (L_{act}) of the in- and output pair messages is measured exactly with 518 µs. Thus, the latency requirements are fulfilled, since the measured values stay within the al-

T	1 s	2 s	5 s	7 s	9 s	11 s
U	u_1 = HL_OFF	u_1 = LED_0	u_1 = LED_100	u_1 = LED_50	u_1 = LED_101	u_1 = LED_75
Y	y_1 = HL_OFF	y_1 = LED_0	y_1 = LED_100	y_1 = LED_50	y_1 = LED_50	y_1 = LED_75
Y_{act}	y_1 = HL_OFF	y_1 = HL_OFF	y_1 = HL_OFF	y_1 = HL_OFF	y_1 = HL_OFF	y_1 = HL_OFF
L	$l_1(u_1, y_1) = 500\,\mu s$					
Δ_L	$j_{L1}(l_1) \leq 100\,\mu s$					
L_{act}	$l_1(u_1, y_1) = 518\,\mu s$ to $518\,\mu s$, MED = $518\,\mu s$, AVG = $518\,\mu s$					
R	$r_1(y_1) = 5000\,\mu s$					
Δ_R	$j_{R1}(r_1) \leq 10\,\mu s$					
R_{act}	$r_1(y_1) = 4998\,\mu s$ to $5002\,\mu s$, MED = $5000\,\mu s$, AVG = $5000\,\mu s$					

Table 2: Tested application with negative test result. Dark gray cells depict negative, light gray positive test results

lowed range. The rate of the output is in the interval of 4998 µs to 5002 µs with a median and an average of 5000 µs. The requirements are fulfilled as well, since the measured values are within the allowed range.

On contrary, the functional requirements are not fulfilled during the first execution and are highlighted with a dark gray background. Light state changes are acknowledged with an Ethernet frame. The reset is acknowledged with the expected value, however, the state changes of the LED brightness are not acknowledged with the expected value. For each generated state change message, the associated acknowledge message contains the same `HL_OFF` value. This behavior indicates a faulty headlight controller implementation. After manually reviewing the source code, a wrong implementation of the acknowledgement function has been confirmed and afterwards resolved.

The same test case has been executed again and is depicted in Table 3. The analysis shows, that all expected values of the functional and non-functional requirements are fulfilled, so that the execution results in a positive test result, which is highlighted with a light gray background. The execution of this test case does not validate the complete functional behavior of the headlight system. It is used to demonstrate the application of a residual bus simulator in RT Ethernet networks to validate functional and non-functional timing requirements.

6. CONCLUSION & OUTLOOK

In this paper a residual bus simulation methodology for the next generation in-car network was presented, which can be used to validate functional and non-functional timing requirements at the same time. A residual bus simulation is a methodology that is used during the validation process to validate modeled requirements. RT Ethernet is a promising technology to overcome the issues given by current in-car networks. It provides high bandwidth while fulfilling real-time characteristics. Thus, validating non-functional timing requirements are of particular interest using RT Ethernet.

Model-based development and testing methods are required to support the development process of applications using RT Ethernet. In general, systems are designed in dedicated models to present requirements and implementation realizations. The UML profile MARTE was applied to model functional and non-functional timing requirements in sequence diagrams. These models are used to derive test cases for the system validation in the model-based testing principle. A residual bus simulator emulates parts of the distributed system and is connected to the SUT via the physical communication interface. It has to undertake synchronization mechanisms and generates data frames that emulate a physical system for the SUT.

TTEthernet was used as a real-time aware Ethernet protocol, but the challenges with parallel frame transmission can be transferred to any RT Ethernet protocol. TTEthernet provides three dedicated message classes with different transmission attributes and a time synchronization procedure to provide a global clock. A residual bus simulation for TTEthernet networks must provide support for these message classes and the synchronization process. In contrast to a residual bus simulation for automotive bus based networks, residual bus simulation test setups in RT Ethernet networks depend on the number of nodes to be tested inside the SUT. It can be classified in directly and indirectly connected approaches. Ethernet as base network technology provides a dedicated transmission domain for each connected device, so that parallel frame transmission has to be taken into account. A directly connected residual bus simulation must be used to validate non-functional timing requirements, but is limited to a single node as SUT. Indirectly connected approaches provide a solution to connect multiple nodes as SUT, but scheduling and bandwidth conflicts may occur, if the simulator is composed of only one network interface.

In order to test non-functional requirements, an existing abstract test case model was extended with the required temporal attributes *latency*, *rate* and their related jitters. This abstract test case model conforms to the model-based testing methodology. It can be utilized on different testing facilities to validate the system in different development stages. Nevertheless, this model can be used to model timing requirements in any real-time aware network such as FlexRay or AVB Ethernet. Due to the provision of a discrete time scale, it can be further utilized as simulation model to drive the residual bus simulation.

The presented methodology was successfully applied to a distributed headlight control system inside a prototype RT Ethernet in-car network to validate modeled requirements.

RT Ethernet residual bus simulation can support the transition to next generation in-car networks. In the future, we are going to investigate how established model-based methodologies such as AUTOSAR and EAST-ADL could co-exist with the presented work. Furthermore, we are going to analyze the real-time and performance aspects of the presented approach, by implementing it to a more suitable hardware platform without the necessity of dual-port memory.

Acknowledgements

This work is funded by the Federal Ministry of Education and Research of Germany (BMBF) within the RECBAR project.

T	1 s	2 s	5 s	7 s	9 s	11 s
U	u_1 = HL_OFF	u_1 = LED_0	u_1 = LED_100	u_1 = LED_50	u_1 = LED_101	u_1 = LED_75
Y	y_1 = HL_OFF	y_1 = LED_0	y_1 = LED_100	y_1 = LED_50	y_1 = LED_50	y_1 = LED_75
Y_{act}	y_1 = HL_OFF	y_1 = LED_0	y_1 = LED_100	y_1 = LED_50	y_1 = LED_50	y_1 = LED_75
L	$l_1(u_1, y_1) = 500\,\mu s$					
Δ_L	$j_{L1}(l_1) \leq 100\,\mu s$					
L_{act}	$l_1(u_1, y_1) = 517\,\mu s$ to $518\,\mu s$, MED = $518\,\mu s$, AVG = $518\,\mu s$					
R	$r_1(y_1) = 5000\,\mu s$					
Δ_R	$j_{R1}(r_1) \leq 10\,\mu s$					
R_{act}	$r_1(y_1) = 4998\,\mu s$ to $5002\,\mu s$, MED = $5000\,\mu s$, AVG = $5000\,\mu s$					

Table 3: Tested application with positive test result. Dark gray cells depict negative, light gray positive test results

7. REFERENCES

[1] Aeronautical Radio Incorporated. Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network. Standard ARINC Report 664P7-1, ARINC, 2009.

[2] E. Bringmann and A. Krämer. Model-Based Testing of Automotive Systems. In *2008 1st Int. Conf. on Software Testing, Verification and Validation*, pages 485–493, Apr. 2008.

[3] S. Demathieu, F. Thomas, C. Andre, S. Gerard, and F. Terrier. First Experiments Using the UML Profile for MARTE. In *11th IEEE Int. Symp. on Object Oriented Real-Time Distributed Computing 2008*, pages 50–57, Piscataway, New Jersey, May 2008. IEEE Press.

[4] Eberspächer. FlexConfig RBS - Remaining Bus Simulation for FlexRay and CAN with FlexConfig RBS. http://www.eberspaecher-electronics.com, 2014.

[5] FlexRay Consortium. FlexRay Communications System Electrical Physical Layer Specification. Specification 3.0.1, FlexRay Consortium, Stuttgart, Oct. 2010.

[6] T. M. Galla. *Cluster Simulation in Time-Triggered Real-Time Systems*. PhD-Thesis, TU Wien, Dec. 1999.

[7] International Organization for Standardization. Road vehicles – Controller Area Network (CAN). ISO 11898, ISO, Genf, 2003.

[8] O. Karfich, F. Bartols, T. Steinbach, F. Korf, and T. C. Schmidt. A Hardware/Software Platform for Real-time Ethernet Cluster Simulation in OMNeT++. In *Proc. of the 6th Int. ICST Conf. on Simulation Tools and Techniques*, pages 334–337, New York, Mar. 2013. ACM-DL.

[9] MOST Cooperation. MOST Specification Rev. 3.0 E2. Technical report, MOST, July 2010.

[10] K. Müller, T. Steinbach, F. Korf, and T. C. Schmidt. A Real-time Ethernet Prototype Platform for Automotive Applications. In *2011 IEEE Int. Conf. on Consumer Electronics - Berlin*, pages 221–225, Piscataway, New Jersey, Sept. 2011. IEEE Press.

[11] Object Management Group. MARTE - Modeling And Analysis Of Real-Time Embedded Systems.

[12] Object Management Group. UML - Unified Modeling Language.

[13] OMNeT++ Community. OMNeT++ 4.4.1.

[14] A. Pretschner, M. Broy, I. H. Krüger, and T. Stauner. Software Engineering for Automotive Systems: A Roadmap. In *2007 Future of Software Engineering*, FOSE '07, pages 55–71, Washington, DC, USA, May 2007. IEEE Computer Society.

[15] P. Skruch, M. Panek, and B. Kowalczyk. Model-Based Testing in Embedded Automotive Systems. In J. Zander, I. Schiefdecker, and P. J. Mosterman, Editors, *Model-Based Testing for Embedded Systems*, Computational Analysis, Synthesis and Design of Dynamic Systems Series, Chap. 19, pages 545–578. CRC Press, Boca Raton, Florida, Sept. 2011.

[16] Society of Automotive Engineers - AS-2D Time Triggered Systems and Architecture Committee. Time-Triggered Ethernet AS6802. SAE Aerospace, Nov. 2011.

[17] T. Steinbach, H. Dieumo Kenfack, F. Korf, and T. C. Schmidt. An Extension of the OMNeT++ INET Framework for Simulating Real-time Ethernet with High Accuracy. In *Proc. of the 4th Int. ICST Conf. on Simulation Tools and Techniques*, pages 375–382, New York, Mar. 2011. ACM-DL.

[18] T. Steinbach, F. Korf, and T. C. Schmidt. Real-time Ethernet for Automotive Applications: A Solution for Future In-Car Networks. In *2011 IEEE Int. Conf. on Consumer Electronics - Berlin*, pages 216–220, Piscataway, New Jersey, Sept. 2011. IEEE Press.

[19] T. Steinbach, H.-T. Lim, F. Korf, T. C. Schmidt, D. Herrscher, and A. Wolisz. Tomorrow's In-Car Interconnect? A Competitive Evaluation of IEEE 802.1 AVB and Time-Triggered Ethernet (AS6802). In *2012 IEEE Vehicular Technology Conf. (VTC Fall)*, Piscataway, New Jersey, Sept. 2012. IEEE Press.

[20] W. Steiner and G. Bauer. Mixed-criticality Networks for Adaptive Systems. In *2010 IEEE/AIAA 29th Digital Avionics Systems Conf. (DASC)*, pages 5.A.3–1–5.A.3–10, Oct. 2010.

[21] M. Utter, A. Pretschner, and B. Legeard. A Taxonomy of Modelbased Testing. Technical Report Working Paper: 04/2006, University of Waikato, Hamilton, New Zealand, Apr. 2006.

[22] Vector Informatik. CANoe 8.2 - ECU Development & Test with CANoe. http://vector.com, 2014.

[23] J. Zander, I. Schiefdecker, and P. J. Mosterman. A Taxonomy of Modelbased Testing for Embedded Systems from Multiple Industry Domains. In J. Zander, I. Schiefdecker, and P. J. Mosterman, editors, *Model-Based Testing for Embedded Systems*, Computational Analysis, Synthesis and Design of Dynamic Systems Series, Chap. 1, pages 3–22. CRC Press, Boca Raton, Florida, Sept. 2011.