Contract-Based Integration of Automotive Control Software

Tobias Sehnke, Matthias Schultalbers
Gasoline Engine Systems
IAV GmbH
{tobias.sehnke | matthias.schultalbers}@iav.de

Rolf Ernst
Institut für Datentechnik und Kommunikationsnetze
Technische Universität Braunschweig
ernst@ida.ing.tu-bs.de

***Abstract*—The functionalities of automotive control are distributed over a large number of independently developed components that are interconnected by complex data dependencies. During integration it is critical to ensure the functional correctness of each component, due to the safety-critical nature of the automotive system.**

Thus existing integration processes ensure that interfaces are syntactically correct. Still in many cases communicated signals are semantically incompatible. This results in complicated errors that are hard to detect and fix. Moreover, existing component languages do not provide applicable means for the description and control of correspondent requirements. In this paper we present a novel methodology for an automated identification of integration errors in automotive control software. The key aspect of our approach are contracts, which are used to disclose domain level requirements. These contracts are then checked during integration supported by existing tools. A case study involving an existing engine control software shows the applicability of our approach by detecting a significant number of formerly unknown integration errors.

I. Introduction

Automotive software is responsible for the safe operation of the physical components of a modern vehicle. It consists of a large number of components, that are connected by data dependencies as shown in Fig. 1. These components perform functions such as signal processing, control and system diagnosis. From system theory it is known, that these functions can be highly sensitive to timing properties such as input delays or sampling rates [1], [2]. Hence these properties must be constrained during design and guaranteed by tests.

The current state of the art for the development of automotive software addresses mostly top-down design processes based on the V-Model. These processes determine, that systems engineers plan all functional aspects of the cyber physical system (CPS) at a high level of abstraction, including timing properties. Therefore system engineers must be aware of all control theoretic aspects of each functionality in the system. This condition is unrealistic, as it does not reflect the reality of the industrial practice, where the majority of components is planned, developed and tested by domain experts. Therefore many details are neither specified nor tested, resulting in functional errors that are identified only very late in the development process. To ensure the correct functionality of each component in the system, requirements from domain experts have to be considered during integration. Hence appropriate means of documentation and test are required.

In this paper we present a novel integration method for component-based automotive systems. The approach derives system-level timing constraints from the domain knowledge of component designers, which are then tested by standard analysis. To describe these timing related properties, we extend existing event-based component models with so called logical timestamps. Logical timestamps can be regarded as references to the sampling time at the beginning of a data flow. From these logical timestamps we derive additional properties that are related to terminology, which is commonly used in the field of networked control systems (NCS). Based on these properties, assumptions on the system behavior can be expressed, even when details of the composed system are unknown. As a means of expression we use a contractual approach rather than specifications. To test the contracts during integration we derive functional dependencies between the specified properties and characteristics, that can be determined using existing analysis methodologies.

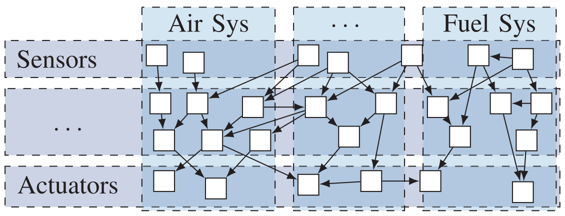
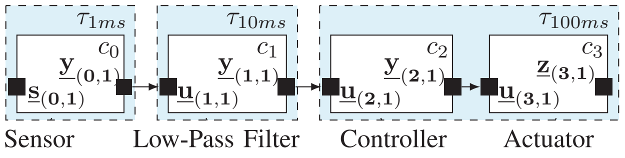


Figure 1. Data dependencies in a component-based engine control software.

II. Related Work

Previous work on the integration of automotive systems has focused mainly on *top-down* approaches. Here signal flows are specified and tested at the system-level based on timing models [3]. Several standards for system specification exist such as the AUTOSAR Timing Extensions [4], AMALTHEA [5] or MARTE [6]. Also several methodologies for the verifications of these specifications have been developed such as Real-time Calculus [7], Compositional Performance Analysis [8] or automata-based methodologies [9], [10]. It has been pointed out recently [11]–[13] that exclusive top-down processes may have only limited applicability in industrial processes. In [12] system-level timing constraints are refined, so that components can be individually developed and tested. In [13] the number of system-level timing constraints for a system is increased by an iterative process based on co-simulation.

There exists several methods for the design and test of controllers and estimators in communicating networks [1], [2] based on the properties of single communications (e.g. sampling rate, input delays). To apply these methods at the domain level the complex data dependencies in automotive software must be abstracted in a similar fashion. None of the existing work has proposed such an abstraction.



τ_{1ms}
c_0
$\underline{\mathbf{y}}_{(\mathbf{0},\mathbf{1})}$
$\underline{\mathbf{s}}_{(\mathbf{0},\mathbf{1})}$
τ_{10ms}
c_1
$\underline{\mathbf{y}}_{(\mathbf{1},\mathbf{1})}$
$\underline{\mathbf{u}}_{(\mathbf{1},\mathbf{1})}$
τ_{100ms}
c_2
$\underline{\mathbf{y}}_{(\mathbf{2},\mathbf{1})}$
$\underline{\mathbf{u}}_{(\mathbf{2},\mathbf{1})}$
c_3
$\underline{\mathbf{z}}_{(\mathbf{3},\mathbf{1})}$
$\underline{\mathbf{u}}_{(\mathbf{3},\mathbf{1})}$
Sensor
Low-Pass Filter
Controller
Actuator

Figure 2. Component-based implementation of a typical control system.

III. System Model

In this section we detail the component model used in this paper. We will also define timing properties, which are used during design and test.

The software is represented by a composed set of components $C=\{c_1, c_2, \ldots\}$, which can be considered as extended AUTOSAR Software-Components (SW-C). An implementation of a component consists of a set of behaviors and ports, also called interfaces. The latter provide a means for communication and are equivalent to data ports of a SW-C. Each behaviour assigns values to output ports $\underline{\mathbf{y}}$ based on inner states and the values on the input ports $\underline{\mathbf{u}}$. The sampling ports $\underline{\mathbf{s}}$ and actuation ports $\underline{\mathbf{z}}$ provide a link to the physical environment. Furthermore $\underline{\mathbf{x}}$ is a placeholder for any port while $\underline{\mathbf{x}}_{(\mathbf{i},\mathbf{j})}$ addresses the j^{th} interface of the component c_i. Each component consist of one or more executable units called runnables which are assigned to schedulable units called a tasks τ. To simplify the presentation, a component will consist of exactly one runnable and is therefore shown as a single unit. A simple example system is shown in Fig. 2.

Each occurrence of a read, write, sampling or actuation operation at the respective interface $\underline{\mathbf{x}}$ is called an event x^k. More precisely an event $x^k = \left(v_x^k, \hat{t}_x^k, t_x^k\right)$ is a triple with the *value* v_x^k, the *tag* $\hat{t}_x^k$ and the *logical timestamp* t_x^k. A tag describes the time of the occurrence of the event. The value represents a physical state in the CPS at a certain point in time, which is called the logical timestamp.

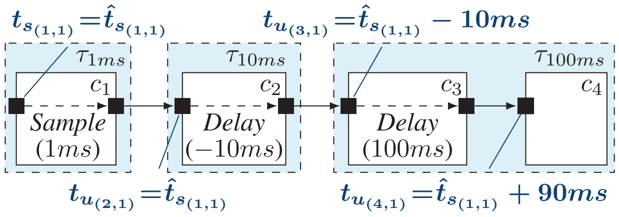
Logical timestamps are derived from the tags of sampling events and propagated in the system according to a set of rules. They can be altered in the components, by functionalities that change the time at which the signal represents the physical state. These alterations are called algorithmic delays.

A signal $x=\left(x^1, \ldots, x^n\right)$ is an ordered set of all events that occur at a single interface. To each signal a set of signal paths $e_x=\{e_x^1, \ldots, e_x^n\}$ can be attributed, which describe the information flow from an information source to the corresponding interface of a signal. More specifically, a signal path e_x^m is an ordered tuple, whose elements can be read, write, sampling or actuator interfaces. Throughout this paper we will focus on signal paths of the form $e_x^m=(\underline{\mathbf{s}}, \ldots, \underline{\mathbf{x}})$, which connect the interfaces of interest to sampling ports. A simple example for such a signal path is shown in Fig. 2 whereas the set of signal paths for the input signal $u_{(3,1)}$ is:

$$e_{u_{(3,1)}} = \left\{ \left(\underline{\mathbf{s}}_{(\mathbf{0},\mathbf{1})}, \underline{\mathbf{y}}_{(\mathbf{0},\mathbf{1})}, \underline{\mathbf{u}}_{(\mathbf{1},\mathbf{1})}, \underline{\mathbf{y}}_{(\mathbf{1},\mathbf{1})}, \underline{\mathbf{u}}_{(\mathbf{2},\mathbf{1})}, \underline{\mathbf{y}}_{(\mathbf{2},\mathbf{1})}, \underline{\mathbf{u}}_{(\mathbf{3},\mathbf{1})} \right) \right\} \tag{1}$$

Definition 1 (Logical Timestamp): Given a signal x with a signal path e_x^m which connects a sampling interface $\underline{\mathbf{s}}$ to the port of the signal $\underline{\mathbf{x}}$. Also for each event x^k a sampling event s^r can be assigned based on a causal effect chain. Then the logical timestamp is the sum of the tag $\hat{t}_s^r$ and the set of algorithmic delays d_x^k, which exist in the signal path, such that

$$t_x^k = \hat{t}_s^r + d_x^k \quad \forall (k, r) \in P_x \tag{2}$$



$t_{s_{(1,1)}} = \hat{t}_{s_{(1,1)}}$
$t_{u_{(3,1)}} = \hat{t}_{s_{(1,1)}} - 10ms$
τ_{1ms}
c_1
Sample (1ms)
τ_{10ms}
c_2
Delay ($-10ms$)
τ_{100ms}
c_3
Delay ($100ms$)
c_4
$t_{u_{(2,1)}} = \hat{t}_{s_{(1,1)}}$
$t_{u_{(4,1)}} = \hat{t}_{s_{(1,1)}} + 90ms$

Figure 3. Example showing the propagation of a logical time stamp along the signal path. It is altered via the algorithmic delays in c_2 and c_3.

where P_x is the set of pairs, which relate the causal events in x and s.

The example in Fig. 3 shows the dependence of the logical timestamps on the sampling tag and the algorithmic delays in the signal path. From the logical timestamps the time-related properties *logical sampling rate* Δt_x, the *logical band-limit* l_x and the *logical data age* a_x are derived.

Definition 2 (Logical Sampling Rate): The logical sampling rate Δt_x is the nonzero difference of the logical timestamp of two consecutive read events in a signal such that

$$\Delta t_x^k = t_x^k - t_x^{k-1} \quad \forall k \in \{1, \ldots, n\} \tag{3}$$

It describes how fast the information can change at an interface. For a strictly uniformly sampled signal the logical sampling rate of the signal can be described by a single value Δt_x such that $\Delta t_x = \Delta t_x^k \in \{1, \ldots, n\}$.

Definition 3 (Logical Band Limit): The logical band limit l_x is a measure for the highest frequency $f_x^{\max}$ in which a signal x can have an amplitude that is nonzero assuming the values of the signal can be described in a spectrum. The band limit is derived from Shannon's sampling limit [14] and is calculated by the following equation

$$l_x = 1/(2 f_x^{\max}) \tag{4}$$

If there exists no spectrum (e.g. if the signal represents a state), the band limit describes a lower bound on the time, in which the signal does not change its values.

Definition 4 (Logical Data Age): The logical data age a_x of a signal is the difference between the tag $\hat{t}_x$ and the logical timestamp t_x of a signal, such that:

$$a_x^k = \hat{t}_x^k - t_x^k \quad \forall k \in \{1, \ldots, n\}\,. \tag{5}$$

It can be interpreted as an input delay. For a static system the logical data age for the signal can be described by a single value a_x such that $a_x = a_x^k \quad \forall k \in \{1, \ldots, n\}$.

IV. Timing Contracts

The correct functionality of the components may depend on a number of signal properties, which have to be ensured by system integrators. To communicate such requirements a contractual approach is used. Their role in the design process is highlighted in Fig. 4. Formally, a contract $T_i = (G_i, A_i)$ describes a model of a software component, which determines the set of assumptions A_i under which the component c_i may be used by its environment, and the corresponding promises G_i that are guaranteed by c_i under such correct use [15]. In the following we present a set of assertions on temporal properties that may result from system design or component development. These properties are derived from the literature of NCS [1], [2] and state estimation [16]. Each assumption implies, that the values of the corresponding signals are correct according to a chosen norm.

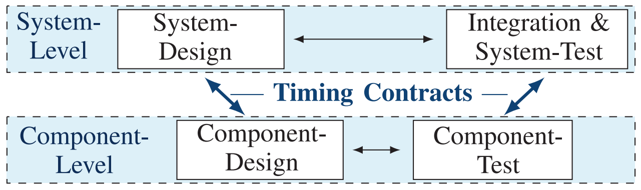


Figure 4. Role of the timing contracts in regard to the development process.

1) Data Age: The assumption $A^{Age}_{(i,l)}$:

$$a_{\min} \leq a^k_{x_{(i,j)}} \leq a_{\max} \quad \forall k \in \{1, \ldots, n\} \tag{6}$$

restricts the data age $a^k_{x_{(i,j)}}$ of all events in the signal $x_{(i,j)}$ to a certain range. This type can be used to constraint the input delays of control functions.

2) Data Synchronicity: The assumption $A^{Sync}_{(i,l)}$:

$$a_{\min} \leq a^k_{x_{(i,j)}} - a^s_{x_{(i,h)}} \leq a_{\max} \quad \forall (k,s) \in S_{(i,l)} \tag{7}$$

specifies that difference of the data ages $a^k_{x_{(i,j)}}$,$a^s_{x_{(i,h)}}$ remains inside of a specified range for all events in $S_{(i,l)}$. The set $S_{(i,l)}$ describes all events pairs of the signals $x_{(i,j)}$,$x_{(i,h)}$, whose values are processed mutually by the component. Synchronization of data ages is crucial when models or controllers compute their outputs based on multiple physical-based inputs.

3) Logical Sampling Rate: The assumption $A^{Sr}_{(i,l)}$:

$$\Delta t_{\min} \leq \Delta t^k_{x_{(i,j)}} \leq \Delta t_{\max} \forall k \in \{1, \ldots, n\} \tag{8}$$

restricts the logical sampling rate $\Delta t^k_{x_{(i,j)}}$ for all events in the signal $x_{(i,j)}$ to a certain range. This assumption can be used among others to constraint the maximum jitter of a signal.

4) Logical Band Limit: The assumption $A^{Bl}_{(i,l)}$:

$$l_{\min} \leq l^k_{x_{(i,j)}} \leq l_{\max} \forall k \in \{1, \ldots, n\} \tag{9}$$

restricts the logical band limit $l_{x_{(i,j)}}$ for all events of the signal $x_{(i,j)}$. This constraint can be used to suppress undesirable frequencies in the signal.

5) No-Aliasing: A no-aliasing assumption $A^{Na}_{(i,l)}$ for the signal $x_{(i,k)}$ demands that there is no aliasing at any input interface in the signal path $e_{x_{(i,j)}}$. Taking into account Shannon's sampling theorem, aliasing occurs when the logical band limit of an output signal is larger than the logical sampling rate of the input signal. If the signal is sampled uniformly, then the no-aliasing assumption for $x_{(i,k)}$ requires that for every pair $\left(y_{(k,l)}, u_{(s,t)}\right)$ in the signal path $e_{x_{(i,j)}}$ the expression

$$l_{y_{(k,l)}} \geq \Delta t_{u_{(s,t)}} \tag{10}$$

is true.

To showcase aliasing in a signal path, we refer back to the example shown in Fig. 2. It originates from a real development project. A sensor samples a signal every $1ms$, which is then processed by a low-pass filter every $10ms$. A controller then calculates a control signal, which is set by an actuator every $100ms$. Obviously the the low-pass filter undersamples its input value. This aliasing is propagated and will not be removed by the following filter. As a result the controller can not be designed with a reasonable performance.

Following the assertions, we also present a set of guarantees, which describe changes in the timing properties due to the internal behavior of the components. Each guarantee implies that the corresponding output value is correct according to a chosen norm. It also relates an output port to an input port.

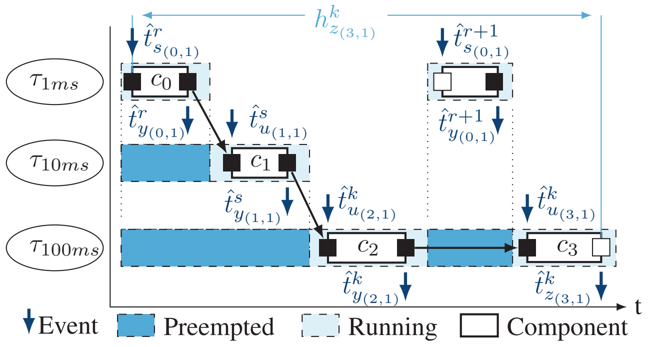


Figure 5. Runtime behavior of the example system. Also the end-to-end latency $h^k_{z_{(3,1)}}$ of the event $z^k_{(3,1)}$ is highlighted.

1) Delay: The guarantee $G^{Lat}_{(i,j,l)}$:

$$t^k_{y_{(i,j)}} = t^k_{u_{(i,k)}} + d_{y_{(i,j)}} \quad \forall k \in \{1, \ldots, n\} \tag{11}$$

specifies that the logical timestamp of each event of the output signal $t_{y_{(i,j)}}$ depends on the logical timestamps in the input signal $t_{u_{(i,k)}}$ and the algorithmic delay $d_{y_{(i,j)}}$. Delays can be either positive or negative depending on the algorithm. Algorithms with negative delays predict future states and can be implemented as models computed by Euler forward solvers.

2) Resampling: The guarantee $G^{Res}_{(i,j,l)}$:

$$l^k_{y_{(i,j)}} = \frac{1}{2 f^{\max}_{y_{(i,j)}}} \quad \forall k \in \{1, \ldots, n\} \tag{12}$$

describes a modification of the logical band limit $l_{y_{(i,j)}}$ of the output signal $y_{(i,j)}$ with the band limiting frequency $f^{\max}_{y_{(i,j)}}$. Various methods are known that can alter the sampling rate of signals. In general these methods can be described as a filter [17]. Assuming that the filter is ideal, the band limit of a signal is equal or less than the cutoff frequency f_c such that $f_s \leq f_c$ or in case of a non-ideal filter $f_s \lesssim f_c$.

V. System-Level Analysis

During integration the signal properties are obtained by analyzing the composed system, to verify the correctness of the documented assertions. The analysis is build onto existing real-time analysis methodology and evaluates the end-to-end behavior of signal paths. Since the signals paths are not specified in the contracts, they must be obtained during analysis. We assume the existence of a description of the components, their interfaces and the communication between these interfaces. Also, the mapping of the components, the scheduling parameters and the execution times of executable units on their respective resources are known.

The signal paths are determined at the system-level using a graph-based analysis methodology, similar to the method proposed by [11]. Thereby the signal flow is represented as a directed graph. At construction of this graph starts with the definition of all interfaces as nodes. Then for each communication between two interfaces an edge is added. Additionally edges are added for intra-component dependencies. These are derived from the linking information that is included in the guarantees of the timing contracts. We implemented this process based on the NetworkX package [18].

To determine the signal properties with existing methodology, we transform these first into functions of the so called *end-to-end latency* [19], tags and delays. The latter are obtained by evaluating the guarantees in the corresponding signal path. The latency is the difference between the tags of the corresponding events of the first interface and the last interface in the signal path as shown in Fig. 5. It can be written as

$$h_x^k = \hat{t}_x^k - \hat{t}_s^r \quad \forall (k,r) \in P_x \tag{13}$$

where P_x is the set of pairs which relate the causal events of x and s. A framework for the computation of these latencies, based on event-chains, is described in [19].

The defintion of the logical data age (5) is reshaped by inserting (2) and (13).

$$a_x^k = \hat{t}_x^k - \hat{t}_s^j + d_x^k = h_x^k + d_x^k \tag{14}$$

The reshaped expression of the logical data age is then inserted into (5) to reformulate the logical time stamp.

$$t_x^k = \hat{t}_x^k - a_x^k = \hat{t}_x^k - h_x^k - d_x^k \tag{15}$$

The description of the logical sampling rate is transformed by inserting (15) into (3). Assuming that the algorithmic delay between two events is constant, the logical sampling rate can then be expressed as a function of the difference of the tags of the events and the difference of the latencies.

$$\Delta t_x^k = t_x^k - t_x^{k-1} = \left(\hat{t}_x^k - \hat{t}_x^{k-1}\right) - \left(h_x^k - h_x^{k-1}\right) \tag{16}$$

Using the equations (14),(16) the timing properties can be determined from the results of a real time analysis.

The logical band limit depends on the band limit of the signal at the predecessing interface in the signal path and band limiting effects between the two interfaces. We consider resampling by the components and limitations due to communication, which can be lower bounded by the logical sampling rate. For each pairing $\left(\underline{\mathbf{y}}, \underline{\mathbf{u}}\right)$ and $\left(\underline{\mathbf{u}}, \underline{\mathbf{y}}\right)$ in a signal path, the logical band limit of the respective interface is determined by

$$l_u^k = \max\{l_y^k, \Delta t_u^k\}, \qquad l_y^k = \max\left\{g_u^k, \Delta t_y^k\right\} \tag{17}$$

whereas the guaranteed bandwidth of an output signal g_u^k is obtained from evaluating the guarantees in the timing contracts. To evaluate (17) an iterative approach is required.

VI. Case Study

In order to test the applicability of the proposed method, a subset of an existing component-based engine control software is analyzed. This software has been used by IAV as a test platform for the development of series production software in multiple projects. It can be characterized as a software product line based on components that can be configured according to project demands. During the analysis we focus on aliasing caused by rate-transitions.

First we evaluate a subset of 55 out of 433 components for their behaviors which are then documented as guarantees. We define a no-aliasing assumption for each signal which represents a physical based state. As a result we obtain assumptions for 81 interfaces. To simplify the analysis we include only harmonic periodic tasks that are scheduled via rate monotonic scheduling into the analysis. Thous we can assume that the data ages are static and the system as uniformly sampled. The signal paths as well as the logical sampling rates and logical band limits are determined using the proposed methods.

For this setup the analysis detects 11 signals where the no-aliasing assumption is not satisfied. This result is unexpectedly high. It may be partially explained due to the conservative choice for the assumptions. Further studies of functional effects are required which may result a weakening of assumptions or functional changes. Still this study shows that the proposed method it is capable of finding integration errors that would go unnoticed otherwise. It can be used to reveal unrealistic domain level assumptions.

VII. Conclusion and Future Work

In this paper we present a contract-based approach to improve the functionality of component-based software by considering domain level requirements during integration. We extend existing event based component models so that timing constraints can be defined at the component level without full knowledge about the target platform. We demonstrate how these constraints can be tested using existing system-level timing analysis. Future work will extend the applicability of the approach for more complex use-cases. Also, best practices must be developed for contract-negotiations.

References

[1] R. A. Gupta and M.-Y. Chow, "Networked control system: overview and research trends", *IEEE Transactions on Industrial Electronics*, vol. 57, no. 7, pp. 2527–2535, Jul. 2010.

[2] L. Zhang *et al.*, "Network-induced constraints in networked control systems - a survey", *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 403–416, 2013.

[3] M.-A. Peraldi-Frati *et al.*, "The timmo-2-use project: time modeling and analysis to use", in *ERTS2012 International Congres on Embedded Real Time Software and Systems*, 2012.

[4] AUTOSAR, *Specification of timing extensions v2.1.1 (r4.1)*, Mar. 2014.

[5] C. Wolff *et al.*, "Amalthea tailoring tools to projects in automotive software development", in *IDAACS'2015*, Jul. 2015, pp. 515–520.

[6] M. Faugere *et al.*, "Marte: also an uml profile for modeling aadl applications", in *Engineering Complex Computer Systems, 2007. 12th IEEE International Conference on*, IEEE, 2007, pp. 359–364.

[7] L. Thiele *et al.*, "Real-time calculus for scheduling hard real-time systems", in *ISCAS 2000 Geneva*, IEEE, vol. 4, 2000, pp. 101–104.

[8] R. Henia *et al.*, "System level performance analysis - the symta/s approach", *Computers and Digital Techniques, IEE Proceedings -*, pp. 148–166, 2005.

[9] M. Deubzer, "Robust scheduling of real-time applications on efficient embedded multicore systems", PhD thesis, Technische Universität München, 2011.

[10] K. G. Larsen *et al.*, "Uppaal in a nutshell", *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 1, no. 1, pp. 134–152, 1997.

[11] S. Mubeen *et al.*, "End-to-end timing analysis of black-box models in legacy vehicular distributed embedded systems", in *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*, IEEE, 2015, pp. 149–158.

[12] P. Reinkemeier *et al.*, "Contracts for schedulability analysis", in *International Conference on Formal Modeling and Analysis of Timed Systems*, Springer, 2015, pp. 270–287.

[13] S. Lampke *et al.*, "Resource-aware control-model-based co-engineering of control algorithms and real-time systems", *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 8, no. 2015-01-0168, pp. 106–114, 2015.

[14] F. Marvasti, *Nonuniform sampling: theory and practice*. Springer Science & Business Media, 2012.

[15] A. Benveniste *et al.*, "Multiple viewpoint contract-based specification and design", in *International Symposium on Formal Methods for Components and Objects*, Springer, 2007, pp. 200–225.

[16] S. Challa *et al.*, "A bayesian solution and its approximations to out-of-sequence measurement problems", *Information Fusion*, vol. 4, no. 3, pp. 185–199, 2003.

[17] L. Milic, *Multirate Filtering for Digital Signal Processing: MATLAB Applications: MATLAB Applications*. IGI Global, 2009.

[18] A. A. Hagberg *et al.*, "Exploring network structure, dynamics, and function using NetworkX", in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA USA, Aug. 2008, pp. 11–15.

[19] N. Feiertag *et al.*, "A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics", in *Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS)*, 2008.