# Translating Timing Constraints during Vehicular Distributed Embedded Systems Development

Saad Mubeen[*†], Mikael Sjödin[*], and Jukka Mäki-Turja[*†]

[*]Mälardalen Real-Time Research Centre (MRTC), Mälardalen University, Sweden
[†]Arcticus Systems AB, Järfälla, Sweden
{saad.mubeen, mikael.sjodin, jukka.maki-turja}@mdh.se

**Abstract.** The end-to-end response-time and delay analysis can verify timing requirements specified on vehicular distributed embedded systems without performing exhaustive testing. For this purpose, the timing requirements and constraints should be unambiguously translated among several models, methodologies and tools that are used at various abstraction levels and phases during the industrial development of these systems. Within this context, we translate timing constraints that are specified at higher abstraction levels using the Timing Augmented Description Language (TADL2) to an industrial model the Rubus Component Model (RCM). We also discuss corresponding extensions in RCM and perform a case study to validate our approach.

**Keywords:** Distributed embedded systems; component-based development; end-to-end timing analysis.

## 1 Introduction

With the recent advancement in computer controlled functionality, the software has immensely increased in size and complexity in vehicular embedded systems. For example, the embedded software in a modern premium car may consist of as many as 100 million lines of code that may occupy up to 1 GB of memory. This software may be realized by more than 2000 software functions. Moreover, it may be distributed over 100 Electronic Control Units (ECUs) [1, 2]. The model- and component-based development appears as a promising approach to deal with the complexity of these systems [3, 4]. This approach uses the principles of Model-Based Software Engineering (MBSE) and Component-Based Software Engineering (CBSE). It supports the use of models to describe functions, structures and other design artifacts. It also supports the development of large software systems by reuse and integration of software components and their architectures. Hence, it raises the level of abstraction for software development. The model- and component-based development of software architectures for vehicular embedded real-time systems has had a surge the last few years [5–8].

### 1.1 Objective and Paper Contribution

There are a number of models, methodologies and tools that are used at various abstraction levels[1] and phases during the development of vehicular distributed

[1] An abstraction level provides a complete definition of the system for a given purpose.

embedded systems. Intuitively, timing requirements and constraints can be specified using one modeling technology, whereas the detailed end-to-end timing analysis can be performed using the tools accompanying another. The end-to-end response-time and delay analyses [9–11] is one of the predominant techniques used in the industry to provide guarantees that the distributed embedded system is going to meet its deadlines when executed. In order to support the analysis, the timing information should be unambiguously translated among several modeling approaches, languages and tools.

Within this context, we consider TIMMO methodology [12] at higher abstraction levels and an existing industrial model the Rubus Component Model (RCM) [13] at the lower level. The TIMMO methodology makes use of EAST-ADL [14] for modeling of software architecture and Timing Augmented Description Language (TADL2) [15] for annotation of timing constraints. TADL2 is intended to provide AUTOSAR with a timing model. At the lower abstraction level, we consider modeling and timing analysis support of RCM and accompanying tool suite Rubus-ICE [16]. We provide translation of those timing constraints, from TADL2 to RCM, that impose restrictions on end-to-end delays. We discuss corresponding extensions in RCM to carry out these translations. We also discuss the semantics and viewpoint of analysis engines about these constraints. In order to provide a proof of concept, we conduct a case study.

We choose RCM instead of AUTOSAR at the lower abstraction level because AUTOSAR lacks a complete timing model, e.g., control flow is not specified unambiguously. This work is first step towards a bigger goal, i.e., development of a seamless tool-chain for model-based development of vehicular real-time systems; and support for inter-operating various modeling and analysis tools [8].

## 2 Background and Related Works

There are several frameworks that support timing modeling such as AADL [17], SCADE [18], MARTE [19], MAST [20], SysML, CHESS [21, 22]. However, the focus in the vehicle industry today is on EAST-ADL and AUTOSAR; RCM is also being used. In this work, we focus only on the vehicular domain.

### 2.1 Abstraction Levels Considered by Various Methodologies

Various models and methodologies used for the development of vehicular distributed embedded systems [14, 15, 5, 23] consider four abstraction levels shown in Fig. 1. At the vehicle level, features and requirements on the end-to-end functionality of the vehicle are captured in an informal and solution-independent way. At the analysis level, the requirements are formally captured in allocation-independent way. Functionality of the system is defined based on the requirements and features without implementation details. A high-level analysis may also be performed for functional verification. The artifacts at this level are developed in an implementation-independent way. These artifacts also contain middleware abstraction, hardware architecture and software functions to hardware allocation. The implementation level contains software-based implementation of the system functionality. The EAST-ADL methodology defines a system at this level in terms of AUTOSAR elements. However, in this work, our focus is on using RCM. Hence, the artifact at this level consists of software architecture of the system defined in terms of Rubus components and their interactions. In this work, we focus on the design and implementation levels.

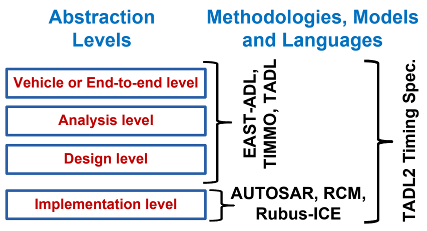

**Fig. 1.** Abstraction levels considered during the development

## 2.2 Models and Methodologies in the Vehicular Domain

**Rubus Component Model (RCM) and Rubus-ICE** Rubus [24] is a collection of methods and tools for model- and component-based development of dependable embedded real-time systems. It is developed by Arcticus Systems [2] in close collaboration with several industrial partners. Rubus is today mainly used for the development of control functionality in vehicles by several international companies, e.g., BAE Systems, Volvo Construction Equipment, Knorr-bremse, Mecel and Hoerbiger. The Rubus concept is based around RCM and its development environment Rubus-ICE which includes modeling tools, code generators, analysis tools and run-time infrastructure. The overall goal of Rubus is to be aggressively resource efficient and to provide means for developing predictable, timing analyzable and synthesizable control functions in resource-constrained embedded systems. The timing analysis supported by Rubus-ICE includes distributed end-to-end response-time and delay analyses [10]. Rubus methods and tools mostly focus at the implementation level in Fig. 1. The lowest-level hierarchical component in RCM is called Software Circuit (SWC). Its purpose is to encapsulate basic functions. Fig. 2 shows an example of a software architecture in RCM composed of SWCs; interconnections between SWCs; and their interactions with external events and actuators with regard to data and triggering.

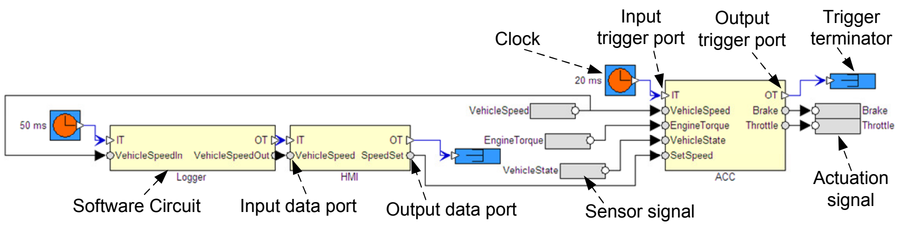

**Fig. 2.** Example of software architecture of a system modeled in RCM.

**AUTOSAR** AUTOSAR [25] is an industrial initiative to provide standardized software architecture for the development of embedded software. It is used at the implementation level in Fig. 1. It describes software development at a higher level of abstraction compared to RCM. The timing model in AUTOSAR is introduced fairly recently compared to RCM. AUTOSAR is more focussed on the functional and structural abstractions, hiding the implementation details about execution and communication. AUTOSAR hides the details that RCM highlights.

[2] http://www.arcticus-systems.com

**EAST-ADL, MARTE, TIMMO, TIMMO-2-USE, TADL and TADL2** TIMMO [6] is an initiative to provide AUTOSAR with a timing model [26]. It is based on a methodology and TADL [23] language which expresses timing requirements and constraints. It is inspired by MARTE [19] which is a UML profile for model-driven development of real-time and embedded systems. TIMMO methodology uses EAST-ADL language [14] for structural modeling and AUTOSAR for the implementation. TIMMO and EAST-ADL focus on the top three levels in Fig. 1. In TIMMO-2-USE project [7], a major redefinition of TADL is done and released in TADL2 specification [15]. TADL2 can specify timing information at all the abstraction levels. Most of these initiatives lack the support on expressing low-level details at the higher levels such as linking information in distributed chains. These details are necessary to extract the end-to-end timing model from the architecture. Furthermore, there is no support on how to extract this information from the model or perform timing analysis. In our view, the end-to-end timing model includes enough information from the systems to be able to perform certain type of timing analysis, e.g., end-to-end response-time analysis.

**Previous Works** In [27], we presented a method to automatically extract the end-to-end timing models only at the implementation level. In [28], we extended our previous method to raise the models extraction at the design level[3]. In [29], we discussed the basic idea for translation of timing constraints.

## 3 Interpretation of TADL2 Timing Constraints in RCM

Timing requirements in TADL2 are modeled by means of timing constraints specified on events and event chains. We discuss those timing constraints that impose restrictions on the end-to-end delays, i.e., reaction and age constraints in Subsections 3.1 and 3.2 respectively. In each subsection, first we discuss semantics of the constraint according to TADL2 specification [15]. Then we provide its translation in RCM and propose corresponding extensions in RCM. Finally, we discuss it with the view point of analysis engines.

**Definitions and Notations** An event represents a distinct form of state change in a running system. It is used to trigger an analysis- or design-level function. When the function is triggered, input data is consumed followed by its processing, transformation and finally production at the output. An event takes place at distinct points in time which are called its occurrences. The occurrences of an event are denoted, e.g., by attributes $s$ and $r$. These attributes are basically time points showing when instances of the event occur. They can be added, subtracted and compared with each other. A constraint often puts limits on the occurrences of events. These limits can be specified in terms of time distances using *minimum* and *maximum* attributes. The occurrences of the events are required to lie within these limits. We use object-oriented notation to define the attributes of a constraint, e.g., TC.response refers to the response event on which the timing constraint TC is specified. In multi-rate systems (see Fig. 3 and 4), components in an event chain can be triggered with independent clocks. Hence, there can be multiple response occurrences to a single occurrence of stimulus

[3] This is an unpublished work and is provided as a technical report for referencing.

in an event chain. In these chains, multiple response occurrences due to each consecutive stimulus occurrence are differentiated by means of colors.

## 3.1 Reaction Constraint

**TADL2 Description** It constrains the occurrence of a response event after the occurrence of a corresponding stimulus event in an event chain. It specifies "how long after the occurrence of a stimulus a corresponding response must occur" [15]. Both reaction and age constraints differ from the delay constraint in a way that they can only be applied to event chains and not to individual events. In order to satisfy the reaction constraint, the earliest occurrence of the response with same color as that of stimulus must take place within the limits specified by this constraint as shown in Fig. 3(a).

**Semantics** A system behavior satisfies the specified Reaction constraint denoted by ReaC if and only if for each occurrence s of the event ReaC.stimulus, there is an occurrence r of the event ReaC.response such that

$$\text{(r.color = s.color) and (r is minimal in ReaC.response with that color)}$$
$$\text{and}$$
$$(\text{minimum} \leq (\text{r- s}) \leq \text{maximum})$$

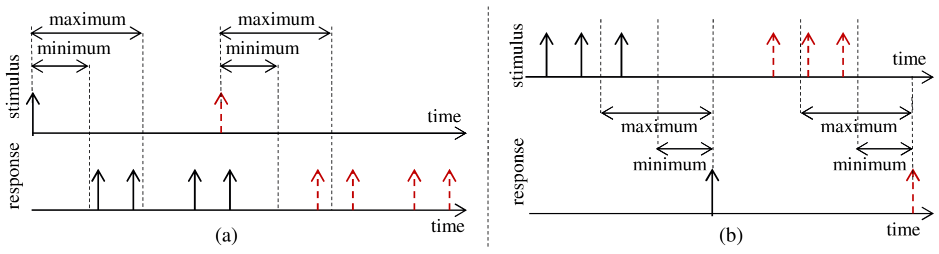

**Fig. 3.** Graphical illustration of (a) Reaction constraint, (b) Age constraint

**Interpretation in RCM** We introduce a new modeling entity in RCM denoted by DataReaction (DR for short) to specify the reaction constraint. This constraint can be specified on an event chain, event chain segment and distributed event chain (distributed over more than one node) by means of DR Start and DR End objects as shown in Fig. 4. The DR End object supports the specification of maximum" attribute by means of a deadline parameter associated to it. However, the minimum parameter is zero. In order to be consistent with TADL2 Reaction constraint, we propose to associate a parameter with DR End object to specify a non-zero minimum value of the constraint.

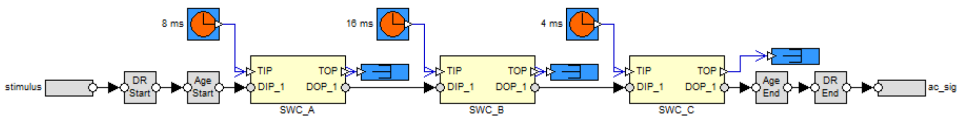

**Fig. 4.** Modeling objects in RCM to specify Reaction and Age constraint

**Interpretation by the Analysis Engines** The analysis engines provided by Rubus-ICE support calculations for the corresponding Reaction delay. Consider the example of an event chain in a multi-rate system in Fig. 4. Fig. 5 shows a time line depicting the execution of this chain (assuming each SWC corresponds to a task denoted by $\tau$ at run-time). $\tau_B$ is deliberately given an offset of 15 time units to maximize the delays. This delay is equal to the time elapsed between the previous non-overwritten release of task $\tau_A$ (input of the chain) and first response of task $\tau_C$ (output of the chain) corresponding to the current non-overwritten release of task $\tau_A$. Assume that a new value of the input is available in the input buffer of task $\tau_A$ "just after" the release of the second instance of task $\tau_A$ (at time $8ms$). Hence, the second instance of task $\tau_A$ "just misses" the read of the new value from its input buffer. This new value has to wait for the next instance of task $\tau_A$ to travel towards the output of the chain. Therefore, the new value is read by the third and forth instances of task $\tau_A$. The first output corresponding to the new value (arriving just after $8ms$) appears at the output of the chain at $34ms$. This results in the delay of $26ms$ as shown in Fig. 5. This phenomenon is more obvious in the case of distributed embedded systems where a task in the receiving node may just miss to read fresh signals from a message that is received from the network. The analysis engines calculate the Reaction delay as shown in Fig. 5 and compare it with the specified constraint parameters. We refer the reader to [10] for further details about the analysis.

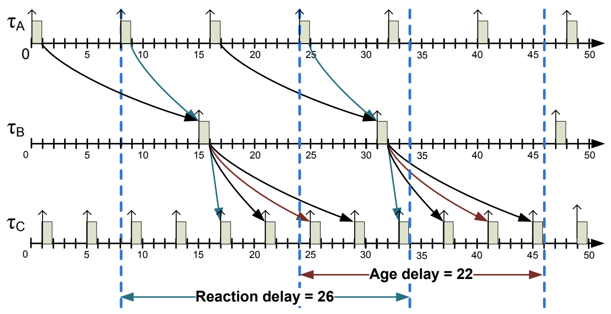

**Fig. 5.** Demonstration of Reaction and Age delay calculations by analysis engines

## 3.2 Age Constraint

**TADL2 Description** It constrains the occurrence of a stimulus from the occurrence of corresponding response looking back through the event chain. Basically it specifies "how long before each response a corresponding stimulus must have occurred" [15]. In order to satisfy this constraint, the latest occurrence of the stimulus with same color as that of the response must lie within the limits specified by this constraint as shown in Fig. 3(b).

**Semantics** A system behavior satisfies the specified Age constraint denoted by AgeC if and only if for each occurrence r of the event AgeC.response, there is an occurrence s of the event AgeC.stimulus such that

$$\text{(s.color = r.color) and (s is maximal in AgeC.stimulus with that color)}$$
$$\text{and}$$
$$(\text{minimum} \leq (\text{r- s}) \leq \text{maximum})$$

**Interpretation in RCM** We introduce a new modeling entity in RCM denoted by DataAge. This constraint can be specified on an event chain, event chain segment and distributed event chain by means of Age Start and Age End objects as shown in Fig. 4. The Age End object supports the specification of "maximum" attribute by means of a deadline parameter associated to it. In order to be consistent with TADL2 Age constraint, we propose to associate a parameter with Age End object to specify non-zero minimum value of the constraint.

**Interpretation by the Analysis Engines** The analysis engines support the calculations for the corresponding Age delay. Consider the example of an event chain in a multi-rate system shown in Fig. 4. Fig. 5 shows the time line when this chain is executed. The analysis engines calculate the Age delay as shown in Fig. 5 and compare it with the specified constraint parameters.

## 4 Automotive-application case study

In order to show the applicability of our approach, we conduct the Steer-By-Wire (SBW) system case study. It is an automotive feature that substitutes most of the mechanical and hydraulic components with electronic components in the steering system of a vehicle. A partial architecture of the SBW system is shown in Fig. 6(a). There are two ECUs (rest of the ECUs are not shown for simplicity) that are connected to a single CAN network. The Steering Control (SC) ECU receives inputs from steering angle, steering torque and vehicle speed sensors. It also receives a CAN message from the Wheel Control (WC) ECU. It sends two CAN messages: one caries steer angle and torque signals; while the other carries feedback signals. Based on all the inputs, it calculates the feedback steering torque and sends it to the feedback torque actuator which is responsible for producing the turning effect of the steering. Similarly, the WC ECU receives inputs from wheel angle and torque sensors. Depending upon these signals and CAN messages received from the SC ECU, it calculates the wheel torque and produces actuation signals for the wheel actuators. It also sends one CAN message carrying wheel torque signal.

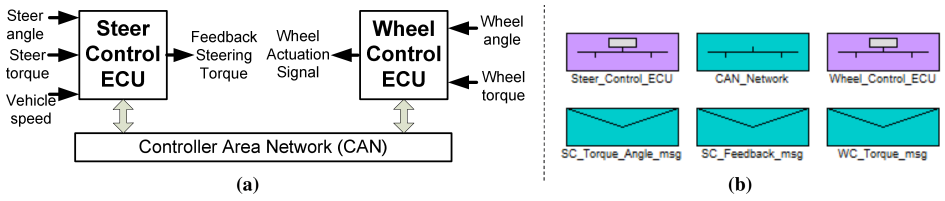

**Fig. 6.** Partial architecture of the steer-by-wire system

The internal software architecture of the two ECUs is modeled with EAST-ADL using EAST-ADLrubusDesigner[4] as shown in Fig. 7. There are two timing constraints namely age and reaction that are specified using TADL2. They put a restriction of 50 ms on the time between the production of steer angle and torque by Steer Controller component and their consumption by Wheel Controller component. These constraints are internally referenced to the components on which they are specified. For convenience, the start and end points for these constraints are identified using the solid-line arrow as shown in Fig. 7.

[4] http://www.arcticus-systems.com

The top level model of the SBW system consisting of models of the two ECUs, a CAN bus and CAN messages in RCM is shown in Fig. 6(b). Whereas, the internal software component architectures of SC and WC ECUs translated from the EAST-ADL model in Fig. 7 to RCM are shown in Fig. 8 and Fig. 9 respectively. The clocks corresponding to trigger flows are translated from the periodic constraints in TADL2 that are specified on the software components in Fig. 7. We use a one-to-one mapping between the software component (both functional as well as sensor and actuator components) in EAST-ADL and software circuit in RCM. The specified TADL2 timing constraints in Fig. 7 are also translated to RCM timing constraints as shown in Fig. 8 and Fig. 9.

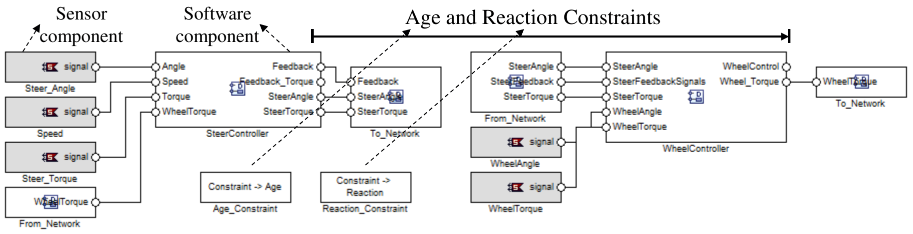

**Fig. 7.** Software architecture of SBW system modeled with EAST-ADL and TADL2

The run-time allocation of the SBW system results in three distributed chains (distributed over more than one node) and thirteen tasks. The analysis engines calculate the age and reaction delays for the distributed chains on which the timing constraints are specified. The calculated age and reaction delays are 30320 $\mu$s and 40320 $\mu$s respectively. A comparison between the specified constraints and calculated delays shows the system satisfies the specified timing constraints.

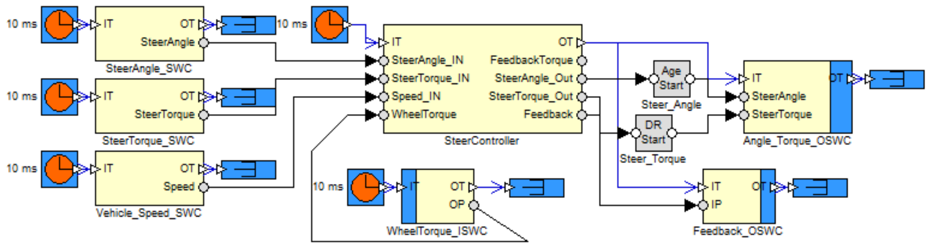

**Fig. 8.** Translated software architecture of SC sub-system in RCM

## 5 Conclusion

We discussed translation of timing constraints that are specified at higher abstraction levels using the TADL2 language to an existing industrial component model RCM. These translations along with our analysis engines enable the application of end-to-end response-time and delay analysis at a higher level of abstraction and early phases during the development of vehicular distributed embedded systems. In order to show the applicability of our approach, we conducted an automotive-application case study. We modeled the software architecture using

EAST-ADL and specified the age and reaction time constraints using TADL2. The software architecture and specified timing constraints were translated into the corresponding models in RCM. The analysis engines automatically analyzed end-to-end delays for the chains on which timing constraints were specified. Currently the translations of timing constraints and corresponding timing analysis is done automatically, however, the conversion of software architecture from the design model (using EAST-ADL) to the implementation model (using RCM) is done manually. In the future, we plan to support automatic conversion of the design-level models to the implementation-level models. Moreover, we plan to implement and validate automatic translations of other timing constraints from TADL2 to RCM including synchronization, repetition and pattern constraints.

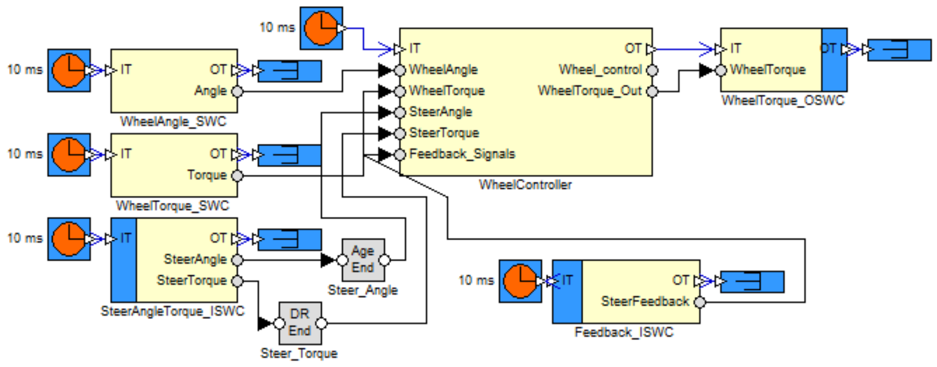

**Fig. 9.** Translated software architecture of WC sub-system in RCM

## Acknowledgement

This work is supported by the Swedish Knowledge Foundation (KKS) and Swedish Research Council (VR) within the project FEMMVA and SynthSoft. We thank our industrial partners Arcticus Systems and Volvo CE, Sweden.

## References

1. R. N. Charette, “This Car Runs on Code,” *Spectrum, IEEE*, vol. 46, no. 2, 2009, http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code.
2. M. Broy, I. Kruger, A. Pretschner, and C. Salzmann, “Engineering automotive software,” *Proceedings of the IEEE*, vol. 95, no. 2, pp. 356 –373, feb. 2007.
3. T. A. Henzinger and J. Sifakis, “The Embedded Systems Design Challenge,” in *Proceedings of the 14th International Symposium on Formal Methods (FM), Lecture Notes in Computer Science*. Springer, 2006, pp. 1–15.
4. I. Crnkovic and M. Larsson, *Building Reliable Component-Based Software Systems*. Norwood, MA, USA: Artech House, Inc., 2002.
5. “AUTOSAR Techincal Overview, Release 4.1, Rev. 2, Ver. 1.1.0., The AUTOSAR Consortium, Oct., 2013,” http://autosar.org.
6. “TIMMO Methodology, Ver. 2,” *TIMMO (TIMing MOdel), Deliverable 7*, Oct. 2009, The TIMMO Consortium.
7. “TIMMO-2-USE,” http://www.timmo-2-use.org/.

8. CRYSTAL - CRitical sYSTem engineering AcceLeration, http://www.crystal-artemis.eu, accessed May, 2014.
9. K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocess. Microprogram.*, vol. 40, pp. 117–134, Apr. 1994.
10. S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Support for end-to-end response-time and delay analysis in the industrial tool suite: Issues, experiences and a case study," *Computer Science and Information Systems, ISSN: 1361-1384*, vol. 10, no. 1, 2013.
11. R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," *Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, March 2005.
12. TIMMO-2-USE Methodology Description, Ver. 2, Del. 13, Jul., 2012.
13. K. Hänninen et.al., "The Rubus Component Model for Resource Constrained Real-Time Systems," in *3rd IEEE International Symposium on Industrial Embedded Systems*, Jun. 2008.
14. "EAST-ADL Domain Model Specification, Deliverable D4.1.1," http://www.atesst.org/home/liblocal/docs/ATESST2_D4.1.1_EAST-ADL2-Specification_2010-06-02.pdf.
15. Timing Augmented Description Language (TADL2) syntax, semantics, metamodel Ver. 2, Deliverable 11, Aug. 2012.
16. "Rubus ICE-Integrated Development Environment," http://www.arcticus-systems.com.
17. P. Feiler, B. Lewis, S. Vestal, and E. Colbert, "An Overview of the SAE Architecture Analysis & Design Language (AADL) Standard: A Basis for Model-Based Architecture-Driven Embedded Systems Engineering," in *Architecture Description Languages*, ser. The International Federation for Information Processing (IFIP). Springer US, 2005, vol. 176, pp. 3–15.
18. SCADE Suite, http://www.esterel-technologies.com/products/scade-suite, accessed May, 2014.
19. "The UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems," Jan. 2010. [Online]. Available: http://www.omgmarte.org/
20. "MAST–Modeling and Analysis Suite for Real-Time Applications," http://mast.unican.es/.
21. CHESS Project, CHESS consortium. Avialable at: http://www.chess-project.org, accessed May, 2014.
22. A. Cicchetti, F. Ciccozzi, S. Mazzini, S. Puri, M. Panunzio, T. Vardanega, and A. Zovi, "Chess: a model-driven engineering tool environment for aiding the development of complex industrial systems," in *27th International Conference on Automated Software Engineering (ASE 2012)*, Sep. 2012.
23. TADL: Timing Augmented Description Language, Ver. 2, Deliverable 6, Oct., 2009.
24. "Rubus models, methods and tools," http://www.arcticus-systems.com.
25. "AUTOSAR Techincal Overview, Version 2.2.2. AUTOSAR – AUTomotive Open System ARchitecture, Release 3.1, The AUTOSAR Consortium, Aug., 2008," http://autosar.org.
26. Mastering Timing Information for Advanced Automotive Systems Engineering. In the TIMMO-2-USE Brochure, 2012. Available at: http://www.timmo-2-use.org/pdf/T2UBrochure.pdf.
27. S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Communications-Oriented Development of Component- Based Vehicular Distributed Real-Time Embedded Systems," *Journal of Systems Architecture*, 2013.
28. S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Component-based vehicular distributed embedded systems: End-to-end timing models extraction at various abstraction levels," Tech. Rep., May 2014. [Online]. Available: http://www.es.mdh.se/publications/3545-
29. S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Towards Translation of Timing Constraints during Vehicular Embedded Systems Development," in *CompArch Work-in-progress Session*, July 2014.