End-to-end Timing Challenges in Seamless Tool Chain Development for Vehicular Embedded Real-Time Systems

Saad Mubeen*†, Jukka Mäki-Turja*† and Mikael Sjödin*
* *Mälardalen Real-Time Research Centre (MRTC), Mälardalen University, Västerås, Sweden*
† *Arcticus Systems AB, Järfälla, Sweden*
saad.mubeen@mdh.se

***Abstract*—Often, there exists mismatch among tools that are used for structural, functional, and execution modeling of vehicular embedded real-time systems in the industry. Building a seamless tool chain to support model- and component-based development of these systems with different, and sometimes independent, tools is challenging. Within this context, we investigate the challenges related to modeling, analyzing, and exchanging end-to-end timing information. We target domain specific models like EAST-ADL supplemented by the Timing Augmented Description Language; and component and execution models that are already used in the industry such as the Rubus Component Model.**

***Keywords*-Automotive embedded real-time systems; timing model; component-based real-time systems; model- and component-based development; timing analysis.**

I. Introduction

The industrial requirements on embedded real-time systems are constantly evolving. With the flexibility offered by software, the complexity of system designs and the amount of advanced computer controlled functionality in products is increasing. Historically, developers of embedded real-time systems have used low level programming languages to guarantee full control of the system behavior. Hence, many embedded real-time systems have become overly complex and hard to manage during functionality or technology shifts. The variety of functionality in today's embedded real-time systems requires development methods and tools that support flexible and efficient development.

Within the business segment of construction-equipment vehicles (and similar segments for heavy special-purpose vehicles), model-based development of software architectures for embedded real-time systems has had a surge the last few years. The idea is to use models to describe functions, structures and other design artifacts. This is in contrast to the previously used text documents. Benefits that are sought by this transition in design technology include simplified communication amongst engineers and other stake holders, use of precise and unambiguous notations to describe complex features, faster turn-around times in early design phases, possibilities to automatically perform timing analysis and derive test cases, and possibilities to automatically generate code.

In practice, existing tools and languages for model-based software design for embedded real-time systems imposes many hinders with respect to information flow between different abstraction levels and project phases. In industry, productivity is hampered by incompatible tools and file formats, in conjunction with the need for non-trivial, manual and tedious translations between different model formats. Moreover, these translations are done in ad hoc fashion making the result of the translation unpredictable and potentially altered in terms of semantics. Thus, there is a strong need to investigate how to work with existing modeling languages and tools in an effective and efficient way. A solution must entail possibilities to make tools inter-operable to allow automated (and semi-automated) translations between modeling languages and tools with preserved model semantics.

A. Motivation and contribution

The motivation for this work comes from the industrial needs at the partner companies. Different tools are used for *structural*, *functional*, and *execution* modeling of vehicular embedded real-time systems. The translations between different models of the systems are done manually and in ad hoc fashion. Hence, there is a need to develop a seamless tool chain that should support *structural*, *functional*, and *execution* modeling of vehicular embedded real-time systems.

In this paper we identify and discuss the challenges related to modeling, analyzing, and exchanging the end-to-end timing information during the development of a seamless tool chain for these systems. We focus on the models and related tools that support model- and component-based development of these systems in the segment of construction-equipment vehicles such as EAST-ADL [1], Timing Augmented Description Language (TADL2) [2], Rubus Component Model [3] and Rubus-ICE [4].

B. Outline

The rest of the paper is organized as follows. In Section II, we discuss the background and related work. In Section III, we discuss the research challenges. Section IV discusses the current work.

II. Background and related work

A. Structural and functional modeling of vehicular real-time systems

The *structural modeling* is concerned with the structure definition of requirements and high-level architectural objects. Whereas, the *functional modeling* refers to the structured way of representing software functions for the system to be modeled. In this work, we consider the structural and functional modeling support of EAST-ADL

which supports domain-specific modeling concepts for modeling product lines of automotive control systems. Basically, it is an architecture description language that tends to describe, capture and model the engineering information of the automotive electronic systems in a standardized way. It describes the functionality of the vehicle at four vertical levels of abstraction starting from requirements capturing to the system implementation as shown in Figure 1.

B. Execution modeling of vehicular real-time systems

The *execution modeling* [5] is concerned with the modeling of run-time properties and/or requirements (e.g., end-to-end deadlines and jitter) of software functions. For example, in resource-constrained and safety-critical embedded systems, it is simply not enough to have a high-level view of the system in the models. Instead, the models need to capture what goes on at the execution level. The modeling of these systems should extend down to the execution level to allow precise control of resource utilization, avoid violation of timing requirements when the system is executed, and certify systems toward safety standards [6]. In this work, we focus on the component-based software development technologies that provide execution modeling support in vehicular domain and are actually used in the industry such as the Rubus Component Model.

C. Abstraction levels during the development of vehicular real-time systems

In this work, we consider four abstraction levels that are described by EAST-ADL. These levels are shown in the Figure 1.

1) Vehicle or end-to-end level: At the vehicle level, requirements, functionality and features of the vehicle are captured in an informal (often textual) and solution-independent way. Basically, this level captures the information regarding what the system should do [7]. In the segment of construction-equipment vehicles, this abstraction level is better known as end-to-end level because features and requirements on the end-to-end functionality of the machine or vehicle are captured in an informal way.

2) Analysis level: At the analysis level, the requirements are captured in a formal way. Functionality of the system is defined based on requirements and features without implementation details. A high-level analysis may also be performed for functional verification.

3) Design level: The artifact developed at the analysis level is refined into design functions at the design level. The resulting artifact at this level also contains middleware abstraction and hardware architecture. In addition, software functions to hardware allocation may be present.

4) Implementation level: At the implementation level, the design-level artifact is refined to software-based implementation of the system functionality. The EAST-ADL methodology defines the system at this level in terms of AUTOSAR elements. However, in this work, our focus is on using the Rubus Component Model and its development environment Rubus-ICE at the implementation level. Hence, the artifact at this level consists of the software architecture of the system defined in terms of Rubus components and their interactions. We choose Rubus instead of AUTOSAR at the implementation level because of industrial needs at the partner companies.

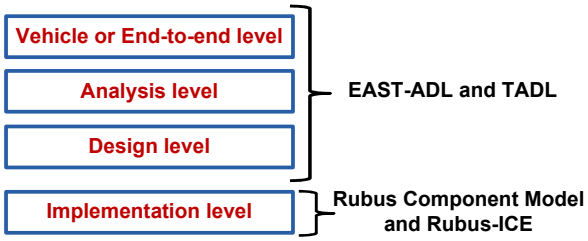

Figure 1. Abstraction levels considered during the development

D. The Rubus concept

Rubus is a collection of methods and tools for model- and component-based development of dependable embedded real-time systems. Rubus is developed by Arcticus Systems [4] in close collaboration with several academic and industrial partners. Rubus is today mainly used for the development of control functionality in vehicles by several international companies [8], [9], [10], [11]. The Rubus concept is based around the Rubus Component Model (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 analysis [12], [13].

E. AUTOSAR, TIMMO, and TADL

AUTOSAR (AUTomotive Open System ARchitecture) [14] is an industrial initiative to provide standardized software architecture for the development of software in the automotive domain. It can be viewed as a standardized distributed component model [15]. In AUTOSAR, the application software is defined in terms of Software Components (SWCs). The virtual function bus handles the distribution of SWCs, their virtual integration and communication at design time. Furthermore, it hides the low-level implementation and communication details at the design time. AUTOSAR provides same interfaces and services to the connected SWCs irrespective of the type of communication (intra- or inter-ECU).

TIMing MOdel (TIMMO) [16] is a large EU research project and serves as an initiative to provide AUTOSAR with a timing model. It describes a predictable methodology and a language Timing Augmented Description Language (TADL) [17] to express timing requirements and timing constraints in all design phases during the development of automotive embedded systems. TIMMO-2-USE [2] is the follow-up project to TIMMO. It defines TADL2 language that includes a major redefinition of TADL.

F. AUTOSAR vs RCM

When AUTOSAR was being developed, there was no focus placed on its ability to specify and handle timing-related information such as real-time requirements and properties. On the other hand, such requirements and capabilities were taken into account right from the beginning during the development of RCM. AUTOSAR describes embedded software development at a relatively higher level of abstraction compared to RCM. The software component in RCM more resembles to the runnable entity compared to AUTOSAR SWC. The runnable entity is schedulable part of AUTOSAR SWC.

As compared to AUTOSAR, RCM clearly distinguishes between the control flow and the data flow among the software components in a node or Electronic Control Unit (ECU). AUTOSAR hides the modeling of the execution environment. On the other hand, RCM explicitly allows the modeling of execution requirements, e.g., jitter and deadlines, at an abstraction level close to the functional specification while abstracting the implementation details. The Sender Receiver communication mechanism in AUTOSAR is very similar to the pipe-and-filter communication mechanism for component interconnection in RCM.

In conclusion, AUTOSAR is more focussed on the functional and structural abstractions, hiding the implementation details about execution and communication. Whereas, RCM is all about modeling, analysis and synthesis of the execution environment of software functions. Basically, AUTOSAR hides the details that RCM highlights.

III. Research Challenges

There are several different types of challenges that are faced during the development of a seamless tool chain to support model- and component-based development of embedded real-time systems in the segment of construction-equipment vehicles. In this section, we identify and discuss only those challenges that are concerned with the modeling, analyzing, and exchanging of the end-to-end timing information.

A. Mismatch between design and implementation levels

When RCM is used instead of AUTOSAR at the implementation level, there exists an incompatibility between the design and implementation levels. We believe, the main reason behind this incompatibility is the concept of virtual function bus in AUTOSAR. At the implementation level, EAST-ADL relies on virtual function bus for the distribution of software components, their virtual integration and communication. Further, virtual function bus hides the low-level implementation and communication details. At the design time, the components are considered at the same level irrespective of the communication they need, i.e., intra- or inter-ECU.

In RCM, there is no concept of virtual function bus. It differentiates between intra- and inter-ECU communication among its software components. It uses network interface components for inter-ECU communication; otherwise, the components communicate with each other via data and trigger ports. Hence, the communications should be explicitly modeled when RCM is used at the implementation level. Moreover, the timing related information on the communications should be explicitly specified in order to perform the end-to-end timing analysis at the implementation level [18].

The problem is that the design-level model does not differentiate between intra- and inter-ECU communications, whereas these communications are explicitly modeled at the implementation level when RCM is used. The timing-related information on communications is also explicitly available at the implementation level when RCM is used. One of the main challenges is to make the design and implementation levels compatible with respect to communications and the end-to-end timing information.

B. Refinement and translation of timing requirements and constraints

The timing requirements and constraints on vehicle features that are captured at the top level may be refined and broken down into more than one requirement and constraint at the lower levels. For example, a timing constraint specified on the braking system feature of the vehicle requires the brakes to be applied within three milliseconds from the time when the brake paddle is pressed. This timing constraint may be refined into more than one constraint at the lower levels. At the implementation level, these (sub) constraints may be specified on several event chains that may be distributed over several ECUs that may be connected to one or more networks. The timing requirements and constraints should be unambiguously refined and translated along all abstraction levels without any loss of timing information.

EAST-ADL supplemented by TADL2 supports the refinement and translation of timing requirements and constraints along all abstraction levels. However, the EAST-ADL methodology assumes that the implementation level is handled by AUTOSAR. When AUTOSAR is replaced by RCM at the implementation level, the refinement and translation of timing information between the design and implementation levels does not hold. Within this context, the challenge is to unambiguously refine and translate the timing requirements and constraints between the design and implementation levels with preserved semantics.

C. Tracing of timing requirements and their verification

Another challenge that we have identified is the need to support traceability of timing requirements from the implementation-level entities to the vehicle-level entities, i.e., following the bottom-up approach. The tracing of timing requirements is important to perform full coverage analysis of the requirements and their verification. Often, a timing requirement at vehicle level may be broken down into several timing requirements at the implementation level. If implementation-level timing requirements are satisfied, the corresponding timing requirement at the vehicle level is considered verified. Hence, the traceability of timing requirements among all abstraction levels should be supported by the tool chain.

The traceability of timing requirements is supported by EAST-ADL from the implementation-level entities to the vehicle level entities. The support for traceability does not hold when AUTOSAR is replaced by RCM at the implementation-level. When RCM and Rubus-ICE are used at the implementation level, the tracing of the timing requirements from Rubus components to the design-level entities arises as another challenge.

The support for traceability of timing requirements is also important for change management. For example, the user of the tool chain may be interested in finding out how do changes in timing requirements, constraints, or budgets at the higher abstraction levels impact on the entities at the lower abstraction levels. This type of support in the tool chain may also be useful to perform design-space exploration during the development of the systems.

D. Raising the end-to-end timing analysis at higher abstraction level

The safety-critical nature of many vehicular embedded real-time systems require evidence that each action by the systems is taken in timely manner. For this purpose, the end-to-end response-time and delay analysis [19], [13] should be supported by the tool chain. In order to perform the timing analysis, the end-to-end timing model[1] should be extracted from the architecture of the system under development. The Rubus-ICE tool suite supports the end-to-end response-time and delay analysis.

When RCM is used at the implementation level, another challenge is to raise the end-to-end timing analysis support provided by Rubus-ICE to the design level (i.e., lifting the analysis one level above). For this purpose, the end-to-end timing model should also be provided at the design level. The analysis framework of Rubus-ICE supports the extraction of end-to-end timing models at the implementation level. However, raising these timing models one level above at the design level is another challenge that we have identified.

IV. Current Work

Currently, we are conducting questionnaire and interviews at the partner companies to identify the patterns, styles of expression, and subsets of the full expressiveness of EAST-ADL that are used by the designers during the development of embedded real-time systems in the segment of construction-equipment vehicles. During the identification of these patterns, styles and subsets, we consider only first three abstraction levels which are vehicle, analysis, and design. After their identification, they will be integrated with the Rubus-ICE at the implementation level for the development of seamless tool chain. During the integration and implementation, we will attack the timing related challenges that we discussed above.

Currently, we are also identifying the most suitable use case at the partner companies for the verification and validation of the tool chain. We specified several requirements on the selection of the use case. That is, it should be a distributed real-time system and it should employ, at least, one CAN bus for communication among ECUs. Whereas, each ECU should have at least one mode and three software components (i.e., two components for network input and output interfaces and at least one component implementing the functionality).

[1] [18] should be referred for the details about end-to-end timing model

Acknowledgement

This work is supported by the Swedish Knowledge Foundation (KKS) within the projects FEMMVA and SythSoft. The authors would like to thank the industrial partners Arcticus Systems and Volvo Construction Equipment (VCE), Sweden.

References

[1] "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.

[2] "TIMMO-2-USE," http://www.timmo-2-use.org/.

[3] K. Hänninen et.al., "The Rubus Component Model for Resource Constrained Real-Time Systems," in *3rd IEEE International Symposium on Industrial Embedded Systems*, June 2008.

[4] "Arcticus Systems," http://www.arcticus-systems.com.

[5] J. Mäki-Turja, K. Hänninen, and M. Nolin, "Towards efficient development of embedded real-time systems, the component based approach," in *International Conference on Embedded Systems & Applications (ESA), 2006*, June 2006.

[6] "ISO 26262-1:2011: Road vehicles Functional safety. http://www.iso.org/."

[7] "Hans Blom et. al. EAST-ADL- An Architecture Description Language for Automotive Software-Intensive Systems. White paper, Version M2.1.10, 2012, http://www.maenad.eu."

[8] "BAE Systems Hägglunds," http://www.baesystems.com/hagglunds.

[9] "Volvo Construction Equipment," http://www.volvoce.com.

[10] "Mecel," web page, http://www.mecel.se.

[11] "Knorr-bremse," web page, http://www.knorr-bremse.com.

[12] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Support for Holistic Response-time Analysis in an Industrial Tool Suite: Implementation Issues, Experiences and a Case Study," in *19th IEEE Conference on Engineering of Computer Based Systems (ECBS)*, April 2012, pp. 210 –221.

[13] 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.

[14] "AUTOSAR Techincal Overview, Version 2.2.2. AUTOSAR – AUTomotive Open System ARchitecture, Release 3.1, The AUTOSAR Consortium, Aug., 2008," http://autosar.org.

[15] H. Heinecke et al., "AUTOSAR – Current results and preparations for exploitation," in *Proceedings of the 7th Euroforum Conference*, ser. EUROFORUM '06, May 2006.

[16] "TIMMO Methodology , Version 2," *TIMMO (TIMing MOdel), Deliverable 7*, October 2009, The TIMMO Consortium.

[17] "TADL: Timing Augmented Description Language, Version 2," *TIMMO (TIMing MOdel), Deliverable 6*, Oct. 2009, The TIMMO Consortium.

[18] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Extraction of end-to-end timing model from component-based distributed real-time embedded systems," in *Time Analysis and Model-Based Design, from Functional Models to Distributed Deployments (TiMoBD) Workshop*. Springer, October 2011, pp. 1–6.

[19] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocess. Microprogram.*, vol. 40, pp. 117–134, April 1994.