The Need of a Timing Model for the AUTOSAR software standard

Razvan Racu, Rolf Ernst
Institute of Computer and Communication Network Engineering
Technical University of Braunschweig
Hans-Sommer Str. 66
D-38106 Braunschweig, Germany

Kai Richter
Symtavision GmbH
Frankfurter Str. 3b
D-38122 Braunschweig, Germany

Abstract— **Even though recognized as a major challenge in system integration, the recently published AUTOSAR standard lacks aspects of timing and performance. Reasons include mismatch between industry requirements and many –mostly academic– timing analysis approaches. The talk highlights key technical and non-technical challenges for defining a comprehensive timing model for AUTOSAR, and outlines requirements for possible solutions. Examples from practice and a look into the industrial process of designing –and the way of thinking– shall help structuring the discussions.**

I. Introduction

The increasing application complexity, together with a strong time-to-market pressure, requires a massively parallel design of systems, whether in automotive, avionics, multimedia, or telecommunications industries. The supply-chain often contains hundreds of companies that design their individual components based on requirement definitions from the OEMs or Tier-1 suppliers.

Systems integration is a major challenge. Dynamic component interactions result in a variety of non-functional timing and performance dependencies due to scheduling, arbitration, blocking, buffering etc., eventually generating hard-to-find timing problems, including transient overload, buffer under- and over-flows, and missed deadlines. Not having a systematic timing analysis procedure is currently challenging the automotive design process. At the same time, the cost of electronic systems has been rising. This cost increase is mainly due to a lack of understanding and control of integration effects, and a resulting conservative design style.

II. AUTOSAR and Timing

The AUTOSAR partnership [1, 4], an alliance of OEM manufacturers and Tier-1 automotive suppliers with many associates, has recognized integration as a major challenge several years ago. Since then, a number of de-facto open industry standards for automotive E/E architectures have been developed: first confidentially, now open to the public. The main goal of AUTOSAR is to define a standard software infrastructure for application and basic software, which allows exchanging parts of the system's software without rebuilding everything. This shall enable modularity, scalability, transferability and re-usability of software among projects, variants, suppliers, customers, etc. Figure 1 shows the software layers within a component, as specified in the AUTOSAR standard. Interestingly though, the current AUTOSAR standard still does not contain key aspects of timing and performance.

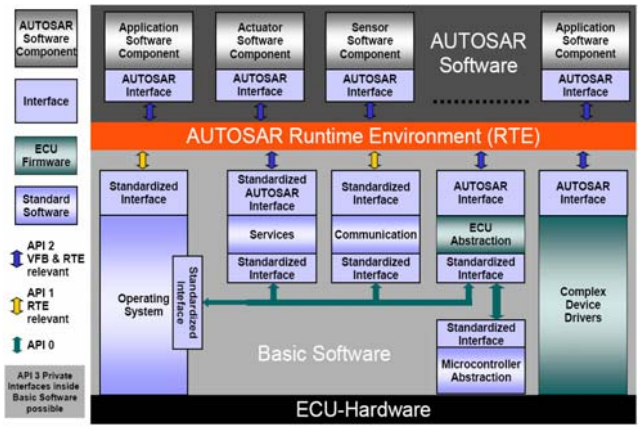

Figure 1 Standardized AUTOSAR software

It is important to understand that the primary objective of AUTOSAR is not solving timing problems in particular but supporting integration from a software-engineering perspective. Timing properties shall be added to the existing component models in a second step. However, the host of interaction and communication mechanisms defined by AUTOSAR –many of them borrowed from earlier standards such as OSEK/VDX [5, 6]– leads to numerous timing use-cases. These, in turn, have many possible interpretations of *timing* with no obvious solution. In the next section we present some examples that show why timing represents a major issue for the correct component integration, and should be considered from the beginning by the AUTOSAR standard. But why is defining a timing model so complex?

The reasons are manifold. Technically, the software-engineering view of AUTOSAR lacks clear execution semantics, on which the known approaches to timing analysis could build upon. The simultaneous use of heterogeneous interaction mechanisms complicates timing analysis further and does not match the clear, well-defined models of computation used in real-time systems. Introducing an effective timing model after the software architecture has been defined is a tough technical challenge and requires practical restrictions.

Practicability concerns and economical issues add to that dilemma. A successful technology must support designers in consequently taking decisions; directly and quickly. Therefore, a suitable methodology for using the model and applying the analysis must be in place. The model and methodology must further consider established design and supply-chain processes. IP protection, in particular, complicates modeling as important details are often not available in a particular design stage today.

III. Model Mismatch

In this section we present three key examples of model mismatches that emphasize the complex relations between the timing properties of the system components.

A. Runnables

At the ECU level, AUTOSAR defines so called *software components (SW-Cs)* as atomic entities from a software development view. However, when it comes to scheduling, each SW-C comprises several so called *runnables*. In the implementation, runnables belonging to different software components are then grouped into *tasks*, which are finally put under operating system control.

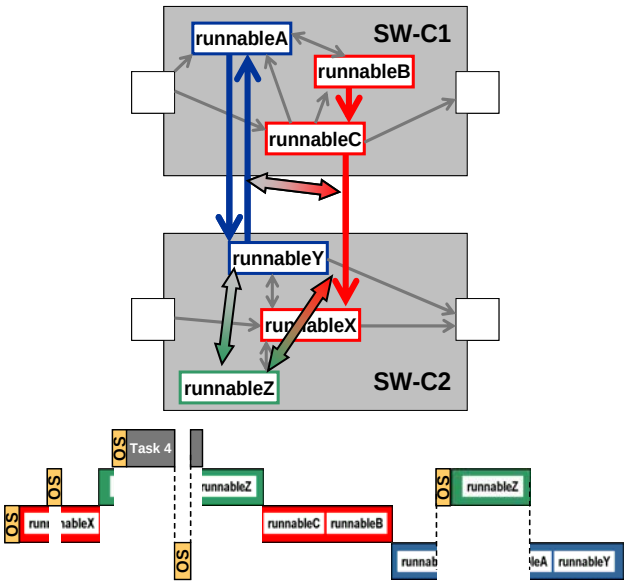

Figure 2 Hidden timing dependencies between SW-Cs

Figure 2 shows two software components containing more runnables building three distributed tasks: the first task contains *runnableX*, on *SW-C2* and *runnableC* and *runnableB* on *SW-C1*; the second task contains *runableA* on *SW-C1* and *runableY* on *SW-C2*; the third task contains *runnableZ* on *SW-C2*. The Gantt-diagram shown in Figure 2 shows the execution trace of the three tasks. As we can observe, implementation dependent timing behavior results from the low level dependencies crossing he component boundaries. This challenges the real-time behavior of the high level components, by introducing hidden, implementation and state dependent additional jitters and delays at the level of software components.

Clearly, a scheduling analysis can be performed on the low level, but the resulting task timing reveals hardly any direct and intuitive timing-relation with the high-level software components, to which timing information shall finally be attached.

B. End-to-End Timing

The second example illustrates another important type of model mismatch at a higher level of communication. With the increasing distribution of functions over several ECUs in a car, the importance of end-to-end timing (and deadlines) is also increasing. AUTOSAR has already defined models for capturing such *timing chains* composed of communicating *software components (SW-C)*. Figure 3 illustrates the software component view of the timing dependencies, mostly determined by the logical flow of data between the software components. Hand-over points (HOPs) shall enable easy composition and decomposition of such chains, thereby providing a framework for system-level timing considerations.

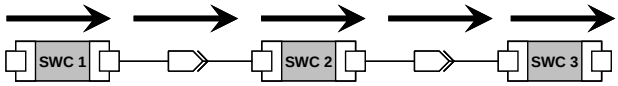

Figure 3 AUTOSAR View on "Timing Chains"

Similar models are also known from data-flow theory, where clear semantics relate the execution of nodes (here: software components) with timing behavior of the stream. However, AUTOSAR has not yet defined clear rules to describe the activation of the tasks within the software components. Hence, the actual timing of software components is undetermined. Moreover, there exist several valid communication semantics including client-server (remote procedure call), periodic sampling including under- and over-sampling, polling, and event-driven. This leads to a variety of indirect *causality chains* in the actual implementation. Figure 4 shows examples for these causality chains through the layered software defined by AUTOSAR.

So, what does end-to-end timing mean in absence of clear execution, communication, and HOP-buffering semantics? The timing is a result of implementation properties and will change with the implementation, uncontrolled by specification and untestable against test component model that lacks the necessary details. There are timing models available that could unambiguously capture and specify such timing properties.

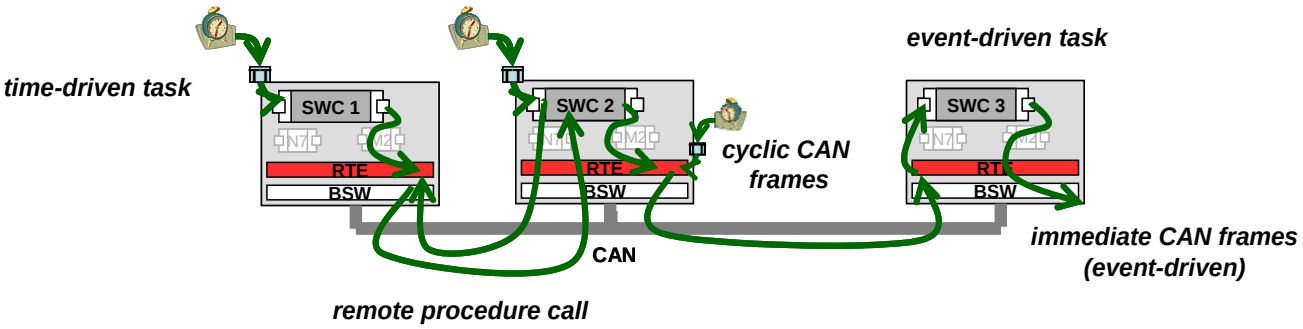

Figure 4 Causality Chains in Automotive Implementations

C. *Bus communication*

A look at bus communication reveals another type of mismatch. AUTOSAR defines a detailed API for the communication stack including several *frame transmission modes (direct, periodic, mixed, none)* and *signal transfer properties (triggered, pending)* with key influences on communication timing. Interestingly, the role of buffers and, in particular, buffer access strategies (FIFO, priority order, etc.) and the over-/underflow mechanisms are mostly left open, despite their enormous influence on signal timing. Figure 5 illustrates the communication mechanism between the software components of two ECUs connected via a CAN bus. The messages to be transmitted are translated into signals and sent into a waiting queue through *periodic, direct or mixed frames.* The waiting signals are buffered into the queue according to different buffering strategies (FIFO, priority ordered, hybrid). The signals waiting in the queue are dispatched by driver interrupts and send over the bus as message objects. Obviously, the frame generation modes and the buffering strategy complicate the timing behavior of the transmitted frames and introduce ambiguity and implementation dependency.

IV. Practicability Concerns

In addition to a clear and intuitive timing model, designers also need a methodology to determine and to utilize the timing model in the established design flow. Based on the feedback we have been receiving from a variety of designers, this in particular requires:

- Generating or obtaining the data needed for analysis (be it by definition, measurement, test, or simply asking the right people).
- Having a specific strategy when and how to apply the technology.
- Interpreting the results and consequently taking decisions.
- Being able to do all this in a reasonable amount of time, after a reasonable amount of training on that technology.

If a technology appears too complex, designers will avoid it. If input data is not readily available, they cannot use it, and if using the technology takes longer than finding a sub-optimal but acceptable manual solution, it will be considered equally useless. We highlight this, as researches (rightfully) tend to do work that is *elegant* or *systematic* in itself without paying too much attention to practical issues.

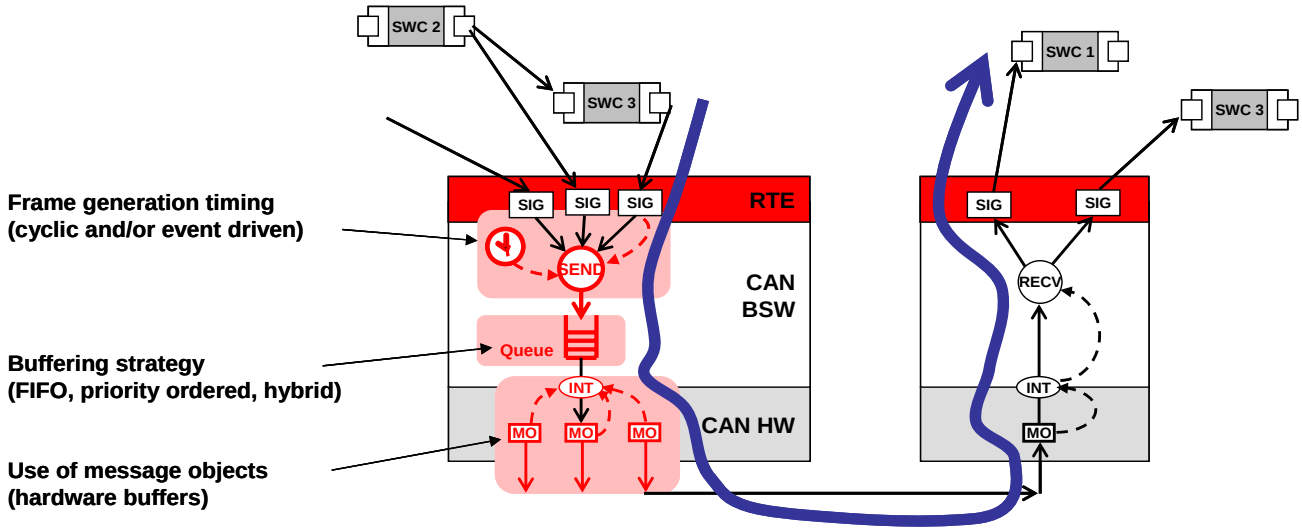

Figure 5 The communication mechanism defined in AUTOSAR

A. Supply-Chain Issues

Specifically car manufacturers nowadays have to cope with an increasing number of unprecedented real-time problems that are caused by the integration of networked applications. Even though OEMs do not develop large parts of the software, they are responsible for the network that is the main basis of integrations. The network timing, however, depends not only on the protocol but also on driver hardware and software (SW-Cs and COM stack), which is mostly out of the OEM's scope of responsibility and control. The supply-chain communication between OEMs and suppliers will have to evolve, most likely by establishing timing contracts between OEMs and suppliers. In order to be accepted

- Responsibilities and scope must be clearly defined, and must match the established roles of suppliers and OEMs.
- IP protection must be ensured, in particular on the supplier's side. Together with already existing standards like AUTOSAR, this will have a dominant impact on the abstraction of a timing model.
- A comprehensive and reliable timing verification methodology must be in place, since there is no point in modeling something that cannot be analyzed.
- It must be clarified what kind of analysis results and what level of accuracy can be obtained at a particular design stage, and the required effort.

V. Experience with formal timing models

We have applied the SymTA/S [7] scheduling analysis technology [8] in several projects with OEMs, Tier-1 and Tier-2 suppliers [9]. SymTA/S allows to capture timing information that is currently available in many stages of the automotive design flow. Experience shows that each particular partner is capable and willing to apply a certain amount of timing analysis, if only the scope is suitable, the analysis can be performed efficiently, and they see a real value for them.

It is clear that automotive platform design and planning can be much more systematic, if supported by a suitable global timing view, the enables reasoning about timing across company borders. On the one hand, academic expertise and support in defining such a model is extremely welcome but it must carefully consider established technological and business processes. On the other hand, industry design practice must also evolve to make designs much more transparent and analyzable, possibly imposing new roles and responsibilities (and liabilities?) for both OEMs and suppliers. As we have shown in the examples above, such a trend is not well supported by the current AUTOSAR standard. Rules and best practice examples are needed to avoid intransparent and inflexible designs that counter some of the key goals of AUTOSAR, platform and supplier independence.

VI. References

[1] AUTOSAR Partnership. www.autosar.org

[2] Kai Richter, Rolf Ernst. Real-Time Analysis as a Quality Feature: Automotive Use-Cases and Applications. In *Embedded World Conference*, Nuremberg, Germany, February 2006.

[3] Kai Richter, The AUTOSAR Timing Model—Status and Challenges, *ARTIST2 Workshop "Beyond AUTOSAR"*, Innsbruck, 2006

[4] AUTOSAR Partnership. AUTOSAR—Current results and preparations for exploitation. *7th EUROFORUM conference Software in the vehicle*. 3-4 May 2006, Stuttgart, Germany

[5] OSEK/VDX Communication. v.3.0.3, OSEK/VDX Consortium, July 2004

[6] OSEK/VDX Operating System. V.2.2.3, OSEK/VDX Consortium, February 2005

[7] SymTA/S Project. Institute of Computer and Communication Network Engineering, Technical University of Braunschweig, Germany, www.symta.org

[8] Rafik Henia, Arne Hamann, Marek Jersak, Razvan Racu, Kai Richter, Rolf Ernst. System Level Performance Analysis - the SymTA/S Approach. *IEE Proceedings Computers and Digital Techniques*, 2005.

[9] Kai Richter, Marek Jersak, Rolf Ernst. How OEMs and Suppliers can tackle the Network Integration Challenges. In *Proc. Embedded Real-Time Software Congress (ERTS)*, Toulouse, France, January 2006.