Timing Modeling and Analysis for AUTOSAR-Based Software Development - A Case Study

Kay Klobedanz, Christoph Kuznik, Andreas Thuy and Wolfgang Mueller
University of Paderborn/C-LAB
Faculty of Electrical Engineering
Computer Science and Mathematics
33102 Paderborn, Germany
{kay.klobedanz, christoph.kuznik, andreas.thuy, wolfgang.mueller}@c-lab.de

Abstract—Safety-critical automotive systems must fulfill hard real-time constraints for reliability and safety. This paper presents a case study for the application of an AUTOSAR-based language for timing modeling and analysis. We present and apply the Timing Augmented Description Language (TADL) and demonstrate a methodology for the development of a speed-adaptive steer-by-wire system. We examine the impact of TADL and the methodology on the development process and the suitability and interoperability of the applied tools with respect to the AUTOSAR-based tool chain in the context of our case study.

I. INTRODUCTION

Concepts and means for timing modeling and analysis are mandatory in the automotive software development process to allow verification and systematic reasoning about the timing behavior of a system under development. Elements influencing the timing behavior due to hardware or software can be scattered all over the system. Their timing properties, which may depend on the implementation and vary between manufacturers, shall be verified against the timing constraints derived from the safety analysis. Those timing properties could be task execution times, bus latencies and sensor sampling rates, for instance. A timing constraint can be a requirement of a controlling algorithm for an ECU that expects updated sensor data every 1ms, for example. When timing properties as well as timing constraints are not modeled explicitly, it is nearly impossible to apply an (automated) verification process.

The AUTOSAR standard [1] is introduced to define a standard layered software architecture and interfaces. This allows the integration of software modules from different vendors throughout the complete supply chain. However, AUTOSAR R3.1 does not sufficiently support timing modeling. For this, the TIMMO project [2] worked on integrating timing analysis for the overall development process and developed a methodology and an event (chain) based language (TADL) for time modeling.

This article presents a study, which examines the impact of event-based timing modeling through the application of TIMMO concepts in the context of an AUTOSAR-based tool chain. We demonstrate a speed-adaptive steer-by-wire system applying an AUTOSAR tool chain based on TIMMO concepts. Thereby, our design decisions are driven by an early model of the timing behavior of the system, which is used later on to verify the system’s timing behavior. We present some advantages that can be achieved by combining both approaches and motivate deeper integration of those concepts into the next AUTOSAR release.

II. FUNDAMENTALS

A. AUTomotive Open System ARchitecture (AUTOSAR)

AUTOSAR [1] is a standardized architecture for automotive software that is developed by an international consortium of automotive OEMs, Tier-1 suppliers and tool vendors. AUTOSAR offers a software component model and a three layered software architecture divided into application software, runtime environment (RTE), and basic software (e.g., drivers and communication system). The RTE realizes an intermediate layer between the hardware independent application software components and the hardware dependent basic software components. Complementary to this, AUTOSAR specifies a standardized XML data exchange format, which supports tool interoperability used in different steps of the development process. Timing aspects of automotive software are not yet adequately covered by the current AUTOSAR release R3.1. There is a lack of a clear timing specification, i.e., the lack of a precise definition of time, an adequate time-based event model, and timing constraints. This is mandatory since the interaction of software components causes a variety of timing dependencies due to scheduling, communication and synchronization effects that are not adequately addressed by AUTOSAR [3].

B. TIMing MOdel (TIMMO)

The ITEA2 TIMMO [2] project introduces a formal, standardized approach through TADL, the Timing Augmented Description Language, and the TIMMO methodology. Both support analysis, verification and traceability of timing constraints across different levels of abstraction and development phases via a standardized exchange format. TIMMO combines and
adapts the model-driven EAST-ADL2 architecture description language [4] and AUTOSAR to introduce timing constructs at meta-model level.

TADL is based on an event chain concept where each element of an event chain is stimulated by an incoming event (stimulus) and produces an outgoing event (response). The time period between stimulus and response is called delay and represents computation time of executed functions between both events. This covers several event models such as periodic events and sporadic events. An element of a TIMMO event chain is depicted in Fig. 1. The following attributes give an overview of the covered properties:

![Fig. 1. Model of an event chain element.](image)

- **Period** $T$ of the element, e.g., period of a task execution.
- **Sampling Period** $T_s$, at which the element reads data.
- **Writing Period** $T_w$, at which the element writes data.
- **Delay** $d$, represents the time between stimulus and response, e.g., task execution time.

Through this it is possible to describe and analyze the system for different end-to-end timing constraints, such as:

- **Reaction**: Delay from a specific (sensor) input value or signal to a corresponding (actuator) output value specifying how long a value or signal needs from one end to the other.
- **Age**: Delay until a specific output (actuator) value is available from a corresponding (sensor) input value or signal (equals the validity of a specific value or signal before a new value arrives).

III. RELATED WORK

A comprehensive overview about automotive design activities with respect to timing can be found in [5]. In [6] the authors introduce a framework for defining end-to-end delays in the presence of multirate, register-based systems. Additionally, different semantics of end-to-end delays that must be carefully distinguished from one another are highlighted. We make use of these defined semantics as they are very suitable for specifying our various (end-to-end) constraints. A related approach to apply seamless timing annotation and evaluation within the design methodology of automotive systems can be found in [7]. Here, the authors present an AUTOSAR conform methodology, which applies formal transformation rules to map the AUTOSAR Software Component template to SystemC semantics in order to approximate communication and application timings based on SystemC runtimes. In our approach, we apply the TIMMO timing chain concept which augments AUTOSAR modeling capabilities on higher abstraction levels like EAST-ADL2 [4]. In contrast, our execution times and paths are computed by a schedulability analysis in conjunction with consideration of adjustments of the underlying bus parameters. Another case study [3] considers flow preservation and time determinism in the communication of an AUTOSAR automatic transmission control. However, the work focuses on task-level timing. Despite the introduction of various methods providing a consistent implementation, the origin of given execution times is unexpressed. The authors of [8] propose a timing augmented design flow and discuss the concept of timing interfaces and contract-based timing specification. Further information on this topic can be found in [9].

IV. SPEED-ADAPTIVE STEER-BY-WIRE CASE STUDY

The TIMMO event chains, as described above, offer means for modeling timing requirements and system timing properties explicitly. The models can be used to verify the timing behavior of the system against the timing models. Due to the TIMMO methodology, a first timing model is already created in an early design phase before the software implementation has started. The results from an early timing analysis support software developers for early design decisions (e.g., architectural decisions). To obtain the full benefit of the results from the timing analysis, it is necessary that the results can be mapped to the AUTOSAR view. In AUTOSAR, there is no central place where timing information is stored. Rather than there are several places where timing information can be added to modeling elements. For example, entities that are scheduled in basic software components are defined by so-called Runnables. Runnables might be scheduled in a periodic manner (e.g. period given in ms) which can be described by a TimingEvent. This example shows that it is possible to specify timing information at certain points. In our case study, we investigated the interplay of TIMMO timing analysis and an AUTOSAR project during the development process.

![Fig. 2. Architecture of steer-by-wire system extended by speed-adaption.](image)

The starting point of the case study was an existing steer-by-wire system [10] that is realized by two ECUs connected by a FlexRay\(^1\) bus. ECU1 controls an active steering wheel and ECU2 controls the corresponding axle (see Fig. 3). We extended our existing system by a speed-adaption of the steering angle. For this, the question we had to answer was if we can migrate the additional tasks for the speed-adaptive behavior on one of the existing ECUs without a conflict with the specified timing behavior of the existing system or to integrate a third ECU for that. The question was to find a solution that satisfies timing requirements on the one hand -

\(^1\)www.flexray.com
e.g., end-to-end timing must be less than 10 ms - and that is cost efficient on the other hand.

In our setup, the limiting factor is the steering wheel sensor whose sampling rate is fixed to 420 Hz (\(T_r = T_w = 2,38\) ms). Given this constraint, we set the periods \(T, T_r\) and \(T_w\) of the FlexRay cycle to 2,38 ms as well. The task execution times which are modeled as a delay \(d\) of the ECUs have an additional impact on the end-to-end delays, which has to be considered. Here, the delay can be defined as the sum of the task execution times: \(d = d_{\text{control}} + d_{\text{adaption}} + d_{\text{COM}}\) for ECU1 and \(d = d_{\text{control}} + d_{\text{COM}}\) for ECU2. Note that, in Section V, we also examine the synchronization of the tasks to the FlexRay cycle to minimize the end-to-end delays.

Our software development process is aligned to a tool chain that includes a timing modeling/analysis tool as well as tools for developing AUTOSAR software systems and AUTOSAR compliant code generators for the basic software. After the requirements are specified, our design flow presumes the creation of an early timing model based on the requirements and an estimation of system properties. An example of a TIMMO model is given by the two ECU variant of our system in Figure 3. The given event chains are used as input for a timing analysis we executed with SymTA/S [11]. SymTA/S is a timing analysis tool based on formal scheduling analysis techniques and symbolic simulation. It supports heterogeneous architectures as well as complex task dependencies and delivers optimization algorithms for rapid design space exploration.

Table I provides the results of the timing analysis, based on the TIMMO event chains created for both scenarios. The values of the CPU load in Table I are based on estimations for the CPU load of the controlling and communication tasks. The results show that it is possible to execute an additional task on ECU1 like shown in Figure 2. In contrast, the worst case end-to-end delays of the 3-ECU solution are much bigger, due to the additional communication overhead via FlexRay. Based on these results, we decided to go for the variant with two ECUs.

The next step in our design flow is to build the AUTOSAR software architecture. Here, we get tool support from SystemDesk\(^2\), which is a system-level design tool from dSPACE that supports modeling of software architectures along the AUTOSAR standard: two software components representing the software running on the two ECUs and the corresponding atomic software components representing the sensors and the actuator. Further information that can be extracted from the timing model includes, for instance, the specification of triggers of TimingEvents specified in Runnables, which is defined by the designer in SystemDesk.

The next step is to configure the AUTOSAR basic software modules for our application. Modules where timing has a direct impact are the FlexRay modules of the AUTOSAR basic software. In our testbench \(TTX\) Universal Control Unit - Multi-Purpose ECU boards from TTTech\(^3\) are applied. Thus, we also apply \(TTX\) Plan and \(TTX\) Build. The first one can be used to create a FIBEX file describing the communication schedule. The designer needs to define, for example the number of messages and the length of the slots of the FlexRay cycle, where most of it can be derived from the timing model and the requirements. The second tool generates the AUTOSAR FlexRay stack along the configuration given in the FIBEX file.

After having generated the basic software modules, modules have to be connected to the application software through the RTE, which can be in turn generated by SystemDesk after having imported the FIBEX file. At this stage, the whole AUTOSAR software becomes available: application code generated by Real-Time Workshop/TargetLink, the RTE generated by SystemDesk and the basic software generated by \(TTX\) Build. This software can be used in a further verification step that checks the validity of earlier timing constraints.

As described in Section IV, we realized our speed-adaptive scenario based on an early timing analysis with SymTA/S where we mainly evaluated the CPU load of the ECUs and the worst case reaction/age end-to-end timings (cf. Section II) from the steering wheel sensor to the steering axle actuator. In addition, SymTA/S also supports the parameterization of the schedules on the ECUs and the communication to enable an improvement of the end-to-end timings. In our investigation we considered two approaches to improve the timing behavior. We first synchronized all tasks to the FlexRay communication cycle. Afterwards we used the design space exploration tool of SymTA/S. This feature computes applicable offsets between the different tasks and the communication to further optimize the reaction/age end-to-end delays. Table II summarizes the findings of the SymTA/S analyses for the properties of the existing system.

It shows that the CPU load is low and not changing when we synchronize the tasks or parameterize the offsets as this does not affect the task execution times. In contrast, the worst case values for the end-to-end delays are shrinking considerably from 8,76 ms (reaction)/13,92 ms (age) to 3,58 ms for both values. These results motivate our further research. Table III shows the results for the timing analysis of the speed-adaptive

---

\(2\)www.dspace.com/systemdesk

\(3\)www.ttech.com
system setup. The CPU load for ECU2 keeps constant since it only contains the control task for the steering axle in both cases. For ECU1 the load certainly grows because it has to execute the adaption task, additionally.

<table>
<thead>
<tr>
<th>Parameterization</th>
<th>cpu load [%]</th>
<th>reaction [ns]</th>
<th>age [ns]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECU1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard</td>
<td>12.61</td>
<td>8.4</td>
<td>13.92</td>
</tr>
<tr>
<td>Synchronized</td>
<td>12.61</td>
<td>7.34</td>
<td>7.34</td>
</tr>
<tr>
<td>Optimized</td>
<td>21.01</td>
<td>3.58</td>
<td>3.58</td>
</tr>
</tbody>
</table>

**TABLE II**

PROPERTIES OF EXISTING SYSTEM.

In contrast to the CPU loads, the end-to-end timings are clearly influenced by the synchronization and the offset parameterization of the tasks. Table III shows the worst case delays for both scenarios in the non-optimized case, which do not fulfill the predefined timing constraints. With the synchronization of the tasks to the FlexRay cycle, we can reduce the delay to values around the desired 10ms. Finally, Table III shows the best possible worst case end-to-end delays we determine with the help of the design space exploration based offset parameterization from SymTA/S. The analysis returns 4,96ms delay. These timing values show that our proposed solution meets the conditions to realize the speed-adaptive steer-by-wire system.

We found the integration of timing modeling in our development process in particular helpful at the very beginning and right after code generation. At the beginning the results of the timing analysis guided our architectural design decision. And after code generation, it is possible to apply a WCRT analysis of the code and check the results against the assumptions made in the timing model. Since the AUTOSAR meta-model does not yet cover the TIMMO properties, AUTOSAR tools do not yet support the TIMMO modeling flow. Thus, it is necessary that insights gathered from the analysis of the timing models have to be included in the AUTOSAR tools by hand.

However, AUTOSAR lacks a clear semantics of event models which makes it hard to apply a verification process. Therefore, it is necessary that the next release of AUTOSAR supports these semantics, e.g., for sporadic events. Furthermore, means to describe synchronization constraints and the traceability of timing properties on different levels of detail are needed.

**VI. CONCLUSION**

Timing specification and analysis is mandatory for complex automotive software. Therefore, extensions to the current AUTOSAR standards are required. In this article we have demonstrated how to close this gap by means of TADL and the TIMMO methodology. Our case study has shown how it smoothly integrates into an AUTOSAR based tool chain and how it combines with WCRT timing analysis.

**ACKNOWLEDGEMENTS**

This work was partly funded by the DFG Collaborative Research Centre 614 and by the German Ministry of Education and Research (BMBF) through the ITEA2 project TIMMO (01IS07002). We greatly appreciate the cooperation with the TIMMO project partners.

**REFERENCES**