

On the Expressive Power of Modeling Languages for Enabling Scheduling Analysis of Automotive Applications

Saoussen Ansi^{1*}, Huascar Espinoza², Arnaud Albinet¹, Sébastien Gérard², François Terrier²

¹Continental Automotive France SAS, PowerTrain E IPP

1 Avenue Paul Ourliac - BP 83649, 31036 France

{saoussen.ansi, arnaud.albinet}@continental-corporation.com

²CEA LIST, Laboratory of model driven engineering for embedded systems,

Point Courrier 94, Gif-sur-Yvette, F-91191 France

{huascar.espinoza, sebastien.gerard, françois.terrier}@cea.fr

Abstract

Automotive software systems are characterized by increasing complexity and tight requirements on safety and timing. Recent industrial experience has indicated that model-based and component-based approaches, using architecture description languages can help improve the overall system quality, foster reuse and evolution, and increase the potential for automatic validation and verification. In this paper, we discuss some crucial specification capabilities that need to be satisfied by modeling languages to enable scheduling analysis. We evaluate the extent to which three major industry-based modeling languages, MARTE, EAST-ADL/TADL and AADL, satisfy those needs.

1. Introduction

The capability to develop highly dependable software solutions in a timely to market way has become one of the decisive competitive factors in the automotive industry. Automotive embedded systems often exhibit hard real-time constraints that need reliable guarantees for full system correctness [12].

For instance, power train and chassis applications include complex (multivariable) control laws, with different sampling periods, and conveying real-time information to distributed devices. Thus, a hard real-time constraint that has to be controlled in power train applications is the ignition timing according to the position of the engine, which is defined by a sporadic event characterizing the zero-position of the flywheel.

End-to-end response delays must also be bounded. An excessive end-to-end response delay of a control loop may induce not only performance degradation but also cause the instability of the vehicle. These constraints have to be assured whatever the situation. Hence, the “worst case” scenario must be taken into consideration, because of the dependable nature of automotive applications.

One mathematical approach to predict “worst-case” timing properties is scheduling analysis. Scheduling analysis can be used at different development stages. Early analysis of a design model aids developers to detect potentially unfeasible real-time architectures and prevents costly design mistakes. A later analysis of an implemented system allows designers to discover timing faults, or to evaluate the impact of possible platform migrations or modifications of scheduling parameters.

In order to master the system complexity and assess system-level trade-offs seeking higher quality and dependability, model-based engineering (MBE) is gaining momentum in the automotive domain. One of the advantages expected from this approach is the ability to exploit correct-by-construction, incremental design processes, which rely extensively on automated transformations and synthesis, and formalized computer-based correctness analysis.

A challenging problem in MBE is to integrate architecture models that are commonly used e.g. for software code generation with the information that is relevant to perform different kinds of mathematical analyses. These analyses (e.g., timing and safety analyses) help validate the system against several quality criteria and predict the system correctness under various conditions. In order to perform these analyses, the system architecture representation must be first

* S. Ansi’s work on this paper has been done in the context of a doctoral internship at CEA LIST

transformed into some form that admits mathematical evaluation. This form is denoted here as “analysis model”. Analysis tools accept as input these analysis models and evaluate them mathematically to produce results used for successive refinement of architecture models. An important goal in MBE is to ensure that the analysis model may be derived directly from a suitably annotated architecture model using automated or semi-automated support [9].

The design of modeling languages is at the core of this issue. While growing in number, they vary widely in terms of the abstractions they support and analysis capabilities they provide. In this paper, we particularly focus on three industry-based modeling languages that cover constructs and parameters required for scheduling analysis. First, OMG’s MARTE (Modeling and Analysis Real-Time and Embedded systems) [2] deals with time-related aspects and includes a detailed set of non-functional attributes to enable state-of-the-art scheduling analyses. Second, we consider EAST-ADL [4] and TADL (Timed-Augmented Description Language) [5], two AUTOSAR-related description languages [1] that cover, as a whole, most of required features to perform timing analysis. Last but not least, SAE’s AADL (Architecture Analysis and Design Language) [6] supports scheduling analysis, particularly for automotive and avionic software applications.

This paper characterizes the required modeling features in terms of (a) system-oriented aspects, i.e. information completeness and, (b) design-oriented aspects, i.e. features improving the designer’s ability for making decisions. Next, we highlight the capabilities and limitations of the covered modeling languages in those aspects. The question is how far these modeling solutions do map to the current challenges in automotive timing analysis. One main goal is to allow people involved in the design of these languages to identify capability areas that would have, to date, been passed over for useful scheduling analysis modeling.

2. Automotive systems and timing analysis

In a typical development process for automotive embedded applications, the system is developed by composing pieces that, all or in part, have already been pre-designed or designed independently by different teams (OEM’s, software suppliers). Therefore, there is a need for supporting design/implementation artifacts by common and standard specification formalisms that will allow plug-and-play of subsystems. For instance, externally supplied software is integrated by car manufacturers into ECUs (Electronic Control Units). The functionality is then distributed over many ECUs into a

network that can even be built with multiple protocols such as CAN, LIN, or FlexRay. To enable modularity, scalability, transferability and reusability of automotive applications among projects, variants, and suppliers, the AUTOSAR [1] partnership has defined a component-based software infrastructure with standardized APIs. The AUTOSAR’s goal is to exchange parts of the embedded systems without the time-consuming and costly need to re-configure, port, and re-build the code.

In AUTOSAR, application models are organized in units called *software components*. Such components hide the implementation of the functionality and behavior they provide and simply expose well defined connection points called ports. In particular, atomic software components are entities that support an implementation and hold behavioral entities called *runnables*. A runnable is an entity that can be executed and scheduled independently from other runnable entities. A runnable may be cyclically triggered by a timing event or triggered by other events that could be posted asynchronously. Runnables can be then attached to one or more OS tasks. Identifying how runnables are allocated to OS tasks is considered as a core problem in ECU configuration. Runnables can be a target for code generation, but they are also the finest-grained application entity to model for scheduling analysis purposes.

AUTOSAR focuses on modeling the structure of software components. Although these models are important for the initial AUTOSAR scope, behavior models are arguably the most important for timing analysis.

Worst-case timing analysis can be employed at different levels of the software behavior description. At a single runnable/task level, well-chosen pieces of code are analyzed on specific hardware platforms. This leads to the calculation of worst-case *execution* times (WCET). At an ECU/system level, a set of runnables/tasks subject to various synchronization interferences, likely executing on distributed hardware platforms, and orchestrated by scheduling strategies, are evaluated for timing correctness. This leads to calculation of e.g., worst-case (end-to-end) *response* delays, jitters, and resource utilizations. This technique, so-called *scheduling analysis*, is our focus in the paper.

Scheduling analysis for automotive applications has received special attention in recent real-time systems literature. A comprehensive summary of analysis techniques for automotive applications is provided in [13]. Among relevant advances in this field, the holistic approach [15] extends the classical single-processor scheduling theory and applies this toward specific combinations of input event models, resource sharing and communication policies. The global view on the system allows taking global dependencies into account

(offsets between the activation instants of tasks), thus providing tightly calculated analysis bounds. A different approach for distributed architectures is the compositional scheduling analysis [7] which is rooted in similar approaches [14] based on Network Calculus. The basic idea of this approach is to break down the analysis calculation into separate local component analyses (e.g. mono-processor analysis with RMA) and to integrate them for system-level analysis by evaluating the propagation of event stream models.

3. Modeling Needs for Scheduling Analysis

In this section, we catalog some essential modeling features to support state-of-the-art scheduling analysis in automotive architectures. These features suffice for the purpose of the paper, which is to provide an informal, comparative review of the expressive power provided by the covered modeling languages. A more comprehensive study of analysis frameworks or design-analysis model transformations is beyond the scope of this paper. We organize the features considered into system-oriented and design-oriented modeling features.

3.1. System-oriented features

These modeling features are related to the attributes and constraints of the targeted system itself (i.e., information completeness). Figure 1 shows a simplified canonical model of the modeling features required for scheduling analysis, which is discussed in this section.

3.1.1. Timing constraints. Basic timing constraints include deadlines and maximum jitters. Both can be specified by relative/absolute durations (maximum time intervals) or instants (occurrence of a timeout event). These constraints can apply to the completion of control/data flows in functional chains or to arbitrary events within computation and communication chains. In particular, the time elapsed since a data is read by a sensor and an output processed with this data is passed to an actuator (known as “data age”) is of interest for the stability of control models.

Most of the scheduling analyses existing today use the notion of physical time as measured by a unique time base. However, distributed applications often experience problems for agreement on consistent time reading due to clock synchronization. This means that scheduling them depends on different time bases, and therefore constraints must refer to specific clocks. Another kind of timing constraint, as mentioned in Section 1, is related to timing assertions measured

according to other variables such as for example the motor cycles. This leads to timing constraints measured in different physical units or via conditional assertions. Modeling languages should account for this diversity to enable useful timing analysis.

3.1.2. End-to-end flows. Scheduling analysis methods are typically scenario-based. This leads to a behavior model supported on the notion of *end-to-end flows*². An end-to-end flow refers to a unique causal set of execution/communication functions triggered by an activation event (or logical combination of events). All possible end-to-end flows in a given system can be generated by starting from the activation events, and then forming event sequences by recursively considering the output event set for the functions producing the events.

The granularity of the functions involved in an end-to-end flow is often related to the choice of black or gray-box component modeling. For the first case, port to port delays should be considered, while for the second scenario runnables may be considered. Whatever the choice, the ordering of functions follows a predecessor-successor pattern, with the possibility of multiple concurrent successors and predecessors, stemming from concurrent functions joins and forks respectively.

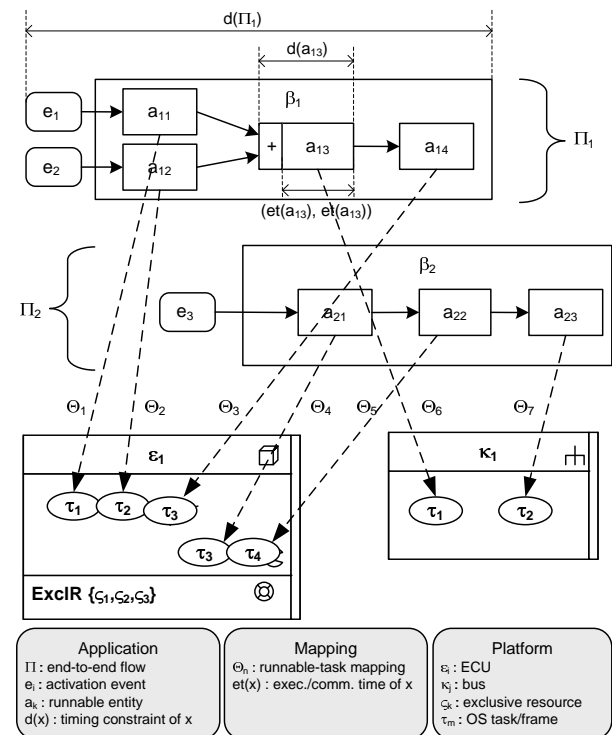


Figure 1. Simplified canonical model for scheduling analysis

² Similar terms are *end-to-end/timing path/chain, transaction*

3.1.3. Activation events. Both event-triggered and time-triggered paradigms are often involved in automotive applications. Event-triggered means that tasks are executed or messages are transmitted by the occurrence of significant events (e.g., a door has been opened). Time-triggered consists of task executed or messages transmitted at predetermined points in time (they communicate by means of asynchronous buffers).

The event-triggered paradigm provides a more efficient use of resources but for analysis purposes it needs a richer set of activation models. This may imply activation events modeled by known patterns and their parameters (e.g., periodic, sporadic, burst activations), activation tables, or by workload generator models (e.g., StateFlow models).

3.1.4. SW and HW resources. What is needed for scheduling analyses is to take into account the impact of the OS and hardware resources on applications (e.g., overheads due to the OS and the stack of communication layers, throughputs, and bandwidths). Among these aspects, protocols for access to mutual exclusive resources are of paramount importance in scheduling analysis of modern multiprocessor architectures [17]. It must be noted that these access protocols are not explicit in AUTOSAR models.

The system model shown in Figure 1 thereby captures information about the applications and the available resource platform of the system, and it also defines the mapping of application functions to OS resources, ECUs and buses.

3.2. Design-oriented features

These features are related to the modeling constructs and styles that serve to organize models and to improve the designer's ability for making decisions.

3.2.1. Application vs. platform. In a typical automotive development flow, application and platform evolve separately. Application artifacts center on functionality and control logic, while platform artifacts focus on ECU/bus selection, middleware layers, and OS services. When creating a modeling approach for specifying this kind of systems, the well-known Y-Chart scheme [8] provides a strong foundation. This approach specifies how the different models of the application and the HW/SW resource platform are put together to build the global system model. This scheme can be regarded as an abstraction of a *layering* architectural pattern, which may include distributed applications, middleware layers, OS services, and hardware.

3.2.2. Allocation and refinement. In order to integrate global models, e.g., for performing system-level analysis, we must recombine application and platform models. This is achieved by means of a third model, often called the allocation model. This model allows exploring different architecture options with respect to a set of functionalities and thereby reusing an architecture platform with different functions. This model also may include the associated timing attributes resulting from the allocation. For example, when allocating a runnable to a given OS task and ECU, one needs to specify its execution time (after e.g. calculation or measurement).

Refinement refers to different level of abstraction of the same real system. Timing modeling aspects and non-functional properties in general, must be allowed to be traced consistently among different level of abstraction. An essential modeling feature to this purpose is the ability of expressing constraint and properties by deriving other properties, likely from other refinement levels. This also enables to compose values and assertions from existing properties and constraints.

3.2.3. Semantics preservation. Automotive applications as described by component-based approaches do not match the models used in typical scheduling analyses. Beyond structural mismatches (which can be solved by model transformation techniques), semantics shall be preserved for full reliability of analysis results with other development activities such as code generation or simulation. In particular, the semantics of port communication and internal behavior of software components need to be aligned with the causal model supported by scheduling analysis techniques.

For instance, a common pattern in end-to-end flows that is particularly important in control systems is over/under sampling. Automotive applications often consist of periodic tasks possible dependent through data interchange via ports. In order to perform reliable timing analysis, the semantics of different synchronization patterns must be specified and aligned to predictions obtained by analysis (e.g., data age).

3.2.4. Composition. As mentioned in Section 2, integration of IP (Intellectual Property) components from different suppliers is frequent in automotive industry. Therefore, there is a need for specification means and composition rules, to which suppliers should comply with, that enable deduction of global timing properties from component properties in order to allow plug-and-play of subsystems in a correct-by-construction manner. To this purpose, component suppliers would need to provide only a set of parameters that characterize the

time behavior of their components without revealing internal designs.

3.2.5. Analysis scope. Due to the specific tools targeted by scheduling analysis, it is important to bound system model elements to a particular analysis or evaluation scope. As automotive applications become more complex, there is often a need to represent a system by multiple analysis models, corresponding to different application-platform allocations, abstraction levels, operational modes, or different quantitative values of non-functional parameters.

4. Modeling Languages Capabilities

We consider three industry-based modeling languages with strong timing analysis basis: EAST-ADL/TADL, MARTE and AADL³. In this paper, we do not provide details on the studied languages. Further information on publications, tutorials, tools, and links can be found via their respective websites: [4]/[5], [2], [6].

Table 1 contains a summary of the extent to which the surveyed modeling languages cover the features considered. Full explanations are given in the sequel.

Timing constraints. Modeling of time and timing constraints differ significantly in these languages.

AADL supports a physical notion of time based on its properties language. The predeclared set of timing properties defines different kind of deadlines, jitters, and communication latencies that can be attached to model elements such as end-to-end flows or subprograms.

TADL, which is the language for time modeling associated to EAST-ADL, allows expressing timing constraints such as maximum delays, repetitions rates, synchronizations, and data ages, by adding timing information to events and events chains. To sets these bounds, the parent concept is *timing constraint*. A timing constraint can specify an upper bound, a lower bound, a nominal value, and a jitter. These parameters are measured in time units and other physical units used in automotive systems such as for example angular degrees and cylinder segments.

MARTE extends the basic time model of UML with synchronous/asynchronous, physical/logical time models as wells as provisions on relativistic effects that can occur in distributed systems. We can distinguish at least three layers of time constructs:

- In a first layer, time is presented as a set of fundamental notions such as time instant, duration, time bases, or clocks.
- In a second layer, MARTE provides mechanisms to annotate timing requirements and constraints in models. One key modeling feature is the concept of observation. Observations provide marking points in models to specify assertions. Some typical assertions have been embedded in ready-to-use patterns, such as jitters or conditional time expressions.
- In the third layer, time concepts are defined as part of the behavior, not mere annotations. This set of constructs cover both physical and logical time.

Figure 2 shows an automotive example of logical clocked time constraints with MARTE. It consists in an ignition control and the knock correction for the case of a 4-stroke engine. Here, the position of the crankshaft is a “natural” reference frame for events and behaviors. An angle logical clock is created to deal with angular positions (crkClk) with degree crank units (°CRK). For instance, the occurrence of the timed event EC is constrained by $\{t_{OTDC} + [5..20]\}$, being $@t_{OTDC}$ an observation of OTDC. The constraint means that $t_{OTDC} + 5 < t_{EC} < t_{OTDC} + 20$, on crkClk and with the values given in °CRK. This can be extended for dealing with multiple clocks specifying ignitions in 4-cylinder engines [18].

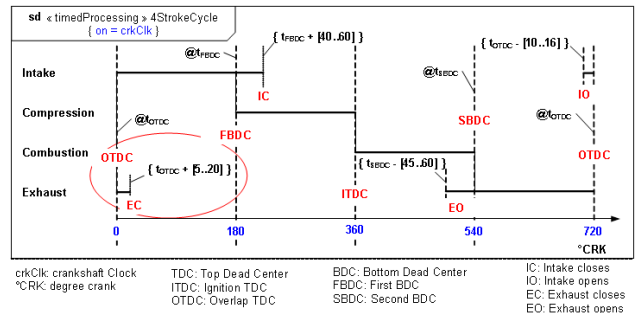


Figure 2. Timing diagram of a 4-stroke engine (source [18])

End-to-end flows. This aspect is supported by the three languages with more or less formality.

TADL calls this concept *event chain*. An event chain relates a set of stimuli to a set of response events. Such events are related to different kind of events handled by both AUTOSAR and EAST-ADL. Time chains can be composed of non-concurrent time chain segments. Figure 3 shows an example of TADL event chain with join and fork precedence relations. In this chain, end-to-end constraints such as reaction constraints and data age constraints can be applied. The semantics of these constraints is discussed later in this section.

³ As we focus on the design phases of system development, the combination AUTOSAR/TADL (related to implementation phases) is let out of the scope of this paper.

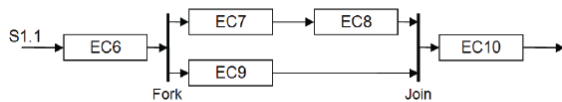


Figure 3. Example of a TADL event chain

In AADL and MARTE, the term used is actually end-to-end flow. In both cases, end-to-end flows describe logical units of processing work in the system, which contend for the use of processing resources (processors, buses, etc.). In both languages, data and control can be part of the processing. Like TADL, different kind of timing constraints can be attached to end-to-end flows (deadlines, output jitters, etc.).

One important feature in MARTE is that end-to-end flows can be represented in behavioral views complementing component models. This approach allows modelers to specify multiple end-to-end flow configurations that could be likely related to (a) specific operational modes, (b) alternative execution chains, or (c) different quantitative scenarios of activation parameters or other non-functional annotations. Thus, one behavior model that has been extended in MARTE with data communications is the sequence diagram. Sequence diagrams capture structural entities (including ports) in the horizontal axis, and interchanging of control and data messages in the vertical axis.

Activation events. Specification of activation events differ in the covered modeling languages.

The activation events are modeled via triggers in EAST-ADL and via events with repetition rate constraints in TADL. A TADL repetitive event describes whether the stimulus is periodic, sporadic, or pattern. The EAST-ADL trigger provides period and offset parameters.

AADL specifies activation by means of dispatch protocols attached to threads. A dispatch protocol can be periodic, sporadic, aperiodic, timed, hybrid, or background.

In MARTE, activation events are modeled by means of workload events. From a specification viewpoint, workload events can be modeled by known patterns (e.g., periodic, aperiodic, sporadic, burst), irregular time intervals, by trace files, or by workload generator models (e.g., state machine models). The workload event element contains additional parameters for periodic and aperiodic patterns such as jitters, burst parameters, and distribution probabilities.

SW and HW resources. The taxonomy of platform resources diverges between the studied languages.

AADL accounts for processors, virtual processors, memories, buses, devices. There are some missing parameters useful for scheduling analysis, which are not explicitly modeled. These include overheads (overheads by context switch times, or message transmission), and common parameters that are commonly calculated by analysis tools such as processor or bus utilizations.

The MARTE analysis model distinguishes two kinds of processing resources: execution hosts, which include for example processors and coprocessors, and communication hosts, which include networks and buses. Processing resources can be characterized by throughput properties as for example processing rate, efficiency properties such as utilization, and overhead properties such as blocking times and clock overheads.

EAST-ADL/TADL includes constructs for hardware and middleware modeling. For scheduling analysis in particular, ECU's relative speed and bus' data transmission rates are useful to obtain absolute execution times and communication delays. On the other hand, EAST-ADL does not explicitly model OS tasks. Thus, the kind of timing analysis supported by EAST-ADL/TADL is more a feasibility analysis that evaluates response times assuming ideal scheduling based on function execution times and triggering definitions.

None of these languages provide sufficient details to describe all the analyzable aspects in some specialized buses used in the automotive domain (e.g., FlexRay). There are, however, some approximation approaches that allow modeling time-triggered communication by means of static schedules (time tables), but they have not been fully formalized with concrete parameters in the surveyed languages.

EAST-ADL abstracts away many of these resource modeling aspects and relies on AUTOSAR. Thus, AUTOSAR includes a detailed task model with actual task/frame configuration and all the required parameters for FlexRay or CAN bus configuration.

Application vs. platform. All the covered modeling languages support separation of application from platform models. They differ in the way these models are created. For example, AADL and MARTE do not pre-define any model view, while EAST-ADL does.

In AADL, layering of system architectures is supported by different ways: hierarchical containment of components, layered use of threads for processing and services, and layered virtual machine abstraction of the execution platform. This is supported by the *abstract component* entity as well as the *software* components, the *hardware* components, and the *system* component. Execution platform modeling elements such as processor, virtual processor, memory, bus, and device can

also represent virtual machine abstractions. Timing analysis properties can be attached at multiple levels.

For its part, MARTE supports this separation of views at different abstraction levels. In particular, at scheduling analysis abstraction level the modeling concepts are organized into a “workload behavior” model dealing with application-specific annotations, and a “resources platform” model dealing with computing and communication annotations.

In contrast, EAST-ADL predefines a set of model views dealing with specific modeling concerns. For instance, three model views characterize the design level: functional design, middleware abstraction, and hardware design. EAST-ADL does not include OS task models. Instead, this level of detail is delegated to the AUTOSAR templates.

Allocation and refinement. The three modeling languages define the concept of allocation but with different terminology. EAST-ADL *allocation constraint* is a concept that can be used with arbitrary elements in a model. AADL “binding” and “allowed binding” model the actual binding and the binding constraints of e.g., application components to execution platform components. In MARTE, the allocation concepts allow for different kinds of spatial and temporal mappings.

An important difference that matters for scheduling analysis is the capability to represent allocation-specific information. While MARTE and AADL support this capability, EAST-ADL does not formalize how allocation-specific information is modeled. MARTE allows for specifying arbitrary non-functional properties attached to allocations. With a different approach, AADL allows to refer to a given binding relationship from property annotations.

Another timing analysis modeling capability is related to the abstraction of OS layers by means of allocation relationships. For instance, automotive applications are typically centered on functional design during early development phases. Functional models commonly do not account for scheduling parameters (priorities, scheduling policies, etc.), required for timing analysis. In order to perform scheduling analysis in early phases, an approach consists in making assumptions in the mapping of functional entities (e.g., runnables) into OS tasks, and in allowing for annotating estimate values directly into such entities. Figure 4 illustrates two allocation options, which can be used at different abstraction levels of the same architecture. In the more abstract level, alternative (b) assumes that Runnable1 and Runnable2 will be allocated to one OS task each, and that they will subsume scheduling-

specific parameters. This mechanism is available in AADL and MARTE.

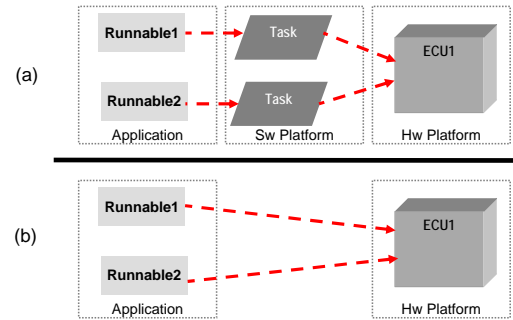


Figure 4. Allocation at different abstraction levels (a) and (b)

Semantics preservation. Due to its importance in automotive system design, the calculation of end-to-end delays, jitters and data age in the presence of multi-rate data communication has been recently considered in literature.

In [19], Feiler and Hansson propose an analysis framework based on AADL to calculate the end-to-end latency and age of signal stream data, as well as their jitter. For the data-driven processing case in particular, this work specifies tasks (so-called threads), likely of multi-rate activation (non-harmonic sampling), connected through immediate or delayed communications. The end-to-end worst-case and best-case delays are calculated for a set of semantic cases accounting for harmonic and non-harmonic up and down sampling.

In a further work, André et al [11] extend AADL calculations for arbitrary schedules by using MARTE. This work separates the logical semantics that determines the relative order of dispatched events and data (modeled as a sequence of execution steps), from the chronometric semantics that determines a total order (modeled as steps deployed in multi-rate tasks and, the latter in hardware resources). Note that while the MARTE notation requires certain specification effort, it may cover more semantic cases than AADL.

In [16] the authors distinguish four different semantics of end-to-end data communication, which are included in TADL. Among these, the “last-to-last” semantics (which derives in the maximum data age calculation) is needed for delay calculation in control engineering, while the “first-to-first” (which derives in the first reaction delay calculation) semantics is of interest e.g. for body electronics where “button-to-action” latency is critical.

Table 1. Modeling Support for Scheduling Analysis

Features - Languages		EAST-ADL/TADL	MARTE	AADL
System-oriented features	<i>Timing constraints</i>	Timing Constraint (Input/output synchronization, repetition rate, reaction, data age). Timing measurement units including time units, angular position, and other engine variables.	Physical and logical time constraints. Clock constraints. Time expressions for conditional assertions, indexed timed events, jitters, etc. based on time observations.	Physical time constraints by means of properties such as deadline, maximum jitters, communications latencies.
	<i>End-to-end flows</i>	Event chains. Event chains related to architecture events. Joins and forks precedence relations	End-to-end flows with join, forks, and conditional precedence relations. Modeled as separated behavior models (e.g., sequence diagram) complementing component models.	End-to-end flows attached to component connectors.
	<i>Activation events</i>	TADL Repetition rated events, EAST-ADL triggers (periodic, sporadic, pattern)	Patterns (e.g., periodic, aperiodic, sporadic, burst), irregular time intervals, by trace files, or by workload generator models.	Dispatch protocols attached to threads, with patterns: periodic, sporadic, aperiodic, timed, hybrid, or background.
	<i>SW and HW resources</i>	Hardware and middleware abstractions. ECUs, System, Bus, Connector. None scheduling parameters.	Execution and communication resources characterized by throughputs, utilization, overheads, scheduling parameters. OS (tasks, semaphores, etc.) and hardware	Processors, virtual processors, memories, buses, devices. No overheads, no ECU/bus utilizations.
Design-oriented features	<i>Application vs. platform</i>	Predefined separation of models at design level for: functional architecture, middleware, and hardware.	No predefined layers. Modeling support for different application and platform views	No predefined layers. hierarchical containment of components, layered use of threads, layered virtual machine abstraction
	<i>Allocation & refinement</i>	Allocation constraints	Allocations with non-functional properties. Refinement of non-functional properties through VSL variables and expressions.	Bindings and allowed bindings. Binding-specific property values.
	<i>Semantics preservation</i>	Four semantics for synchronization in end-to-end flows: first-to-first (first reaction), first-to-last, last-to-first, last-to-last (data age).	Logical relationships between read/write events	Multi-rate activation (non-harmonic sampling), connected through immediate or delayed communications
	<i>Composition</i>	Black-box composition rules for event chains	Required/offered/contract non-functional annotations in component interfaces	None
	<i>Analysis scope</i>	None	Analysis context, variables/parameters, analysis results	None

* see Section 4 for full explanations

Composition. AADL and MARTE do not formally account for the issue of composition of end-to-end flows for scheduling analysis. In MARTE, however, assumed/guaranteed non-functional properties can be annotated in component interfaces by using specialized constraint annotations. This is particularly useful to specify contract-based component applications. On the other hand, TADL is the sole language from these three that proposes calculation of end-to-end delays for black-box compositionality. Its vertical compositionality makes the approach applicable to different schedulers (SP, TDMA, RR, and EDF) and for realistic task models, e.g. periodic tasks with jitter or burst, as long as a response-time analysis is available [16]. Its horizontal compositionality makes it applicable to distributed systems with several ECUs and buses, be it synchronous (e.g. with FlexRay) or asynchronous (e.g. with CAN).

Analysis scope. Neither AADL nor EAST-ADL provide any construct to bound modeling elements and properties participating in analysis scenarios. MARTE, for its part, introduces the notion of *analysis context*. An analysis context is the root concept used to collect relevant quantitative information for performing a specific analysis scenario. Starting with the analysis con-

text and its parameters, a tool can follow the links of the model to extract the information that it needs to perform scheduling analysis. Analysis results can also be annotated back in MARTE models to take them into account for architecture refinement.

Analysis contexts, together with SysML parameters [3], can be also used to formalize the design space for architecture exploration. In [10], MARTE and SysML are used in system evaluation and optimization scenarios, by explicitly defining objective functions, costs, and the search space (multiple candidate architectures and design constraints).

5. Conclusions

In this paper, we discuss some crucial specification capabilities that need to be satisfied by modeling languages to enable scheduling-aware timing analysis in automotive applications. We evaluate the extent to which three major industry-based modeling languages, MARTE, EAST-ADL/TADL and AADL, satisfy those needs.

The purpose of this paper is to raise questions and encourage discussion, rather than to argue that a particular modeling language is the answer. We think that a good modeling language should be general enough to

capture appropriate constructs to perform useful system analysis at different abstraction levels. As can be expected, different architectural styles, fidelity levels, and language formalities in the three surveyed languages lead to diverging capabilities. On the other hand, these constructs must not be too disconnected from the system execution semantics so that analysis results can be reliable. This is a wide-open area that needs more research in the three modeling languages discussed.

Finally, although it seems that MARTE has the richest expressive power, because of the range of applications and areas of knowledge that are in its scope, cares must be taken. The design of modeling languages typically involves a trade-off between expressive power and analyzability. The more a language can express, the harder it becomes to understand what instances of the formalism use. Thus, more expressive languages commonly require additional effort to define methodological frameworks and tools well suited, in this case, to the automotive industry.

6. Acknowledgements

This work has been supported by the EDONA project, which is partially funded by: "Direction générale des Entreprises" from the French Ministry of Industry, "Region Ile de France", "Conseil général des Yvelines", "Conseil général de l'Essonne", "Conseil général des Hauts-de-Seine", "Conseil général du Val d'Oise". The CEA LIST work on this paper was also partially funded by the INTERESTED (<http://www.interested-ip.eu/>) and TIMMO projects (<https://www.timmo.org/>).

7. References

- [1] AUTOSAR Partnership. www.autosar.org
- [2] MARTE website. www.omgmarTE.org
- [3] SysML website. www.omgSysML.org
- [4] EAST-ADL website. www.atesst.org
- [5] TIMMO website. www.timmo.org
- [6] AADL website www.aadl.info
- [7] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, R. Ernst. System Level Performance Analysis - the SymTA/S Approach. In IEE Proceedings Computers and Digital Techniques, Vol. 152, Is. 2, March 2005.
- [8] R. Chen, M. Sgroi, G. Martin, L. Lavagno, A. L. Sangiovanni-Vincentelli, J. Rabaey: UML for Real: Design of Embedded Real-Time Systems, pp. 189-270, Kluwer Academic Publishers, May 2003.
- [9] H. Espinoza, "An Integrated Model-Driven Framework for Specifying and Analyzing Non-Functional Properties of Real-Time Systems", PhD Thesis (English), University of Evry, FRANCE. Sept. 2007.
- [10] H. Espinoza, D. Servat, and S. Gérard, Leveraging Analysis-Aided Design Decision Knowledge in UML-Based Development of Embedded Systems, SHARK-ICSE 2008. Leipzig, Germany. May 2008.
- [11] C. André, F. Mallet, R. de Simone, "Modeling of Immediate vs. Delayed Data Communications: from AADL to UML MARTE", ECSI Forum (FDL), September 2007, Spain.
- [12] N. Navet, F. Simonot-Lion, editors, "The Automotive Embedded Systems Handbook", Industrial Information Technology series, CRC Press / Taylor and Francis, ISBN 978-0849380266, December 2008.
- [13] M. Di Natale, W. Zheng, C. Pinello, P. Giusto, A. L. Sangiovanni-Vincentelli: "Optimizing End-to-End Latencies by Adaptation of the Activation Events in Distributed Automotive Systems". IEEE RTAS Symposium 2007: 293-302
- [14] E. Wandeler, A. Maxiaguine, L. Thiele: "Quantitative Characterization of Event Streams in Analysis of Hard Real-Time Applications". IEEE RTAS Symposium 2004: 450-461
- [15] J. Gutierrez, J. Garcia, and M. Harbour, "On the Schedulability Analysis for Distributed Hard Real-Time Systems," Proceedings of the 9th Euromicro Workshop on Real-Time Systems, Toledo, Spain, pp. 136-143, 1997.
- [16] N. Feiertag, K. Richter, J. Nordlander, J. Jonsson, "A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics". In 1st Workshop on Compositional Theory and Technology for Real-Time Embedded Systems CRTS'08, 2008.
- [17] Simon Schliecker and Jonas Rox and Mircea Negrean and Kai Richter and Marek Jersak and Rolf Ernst. "System Level Performance Analysis for Real-Time Automotive Multi-Core and Network Architectures." In IEEE Transactions on Computer Aided Design (to appear), 2009.
- [18] C. André, F. Mallet, M-A. Peraldi, "A multiform time approach to real-time system modeling: Application to an automotive system", IEEE Int. Symp. on Industrial Embedded Systems (SIES'2007), July 2007, Portugal, pp. 234-241.
- [19] P. Feiler, J. Hansson, "Flow Latency Analysis with the Architecture Analysis and Design Language (AADL)", Carnegie Mellon University, Technical Note CMU/SEI-2007-TN-010, December 2007.