Specifying Timing Requirements in Domain Specific Languages for Modeling

Damjan Temelkovski
University of Westminster
London, United Kingdom
damjantemelkovski@gmail.com

Ljerka Beus-Dukic
University of Westminster
London, United Kingdom
l.beus-dukic@westminster.ac.uk

Abstract—**Complex Real-Time Embedded Systems (RTESs) can be developed using model-based engineering. The problem is choosing a modeling language that has capabilities to model the most important characteristic of RTESs: timing. This paper shows an analysis of the most popular modeling languages and their capabilities to model timing constraints in RTESs. It includes UML, SysML, AADL, MARTE and EAST-ADL. A brief comparison between MARTE and EAST-ADL, based on the case study from the automotive industry, is also included.**

Keywords-model-based engineering; modeling languages; MARTE; EAST-ADL; real-time embedded systems;

I. INTRODUCTION

A Real-Time Embedded System (RTES) is an embedded system where the correctness of the system depends on the time it produces the result as well as the logical correctness. Hard RTESs have timing constraints which they must satisfy or the result would be a system failure. RTESs are called safety-critical if a failure of the system leads to the loss of human lives. Examples of such systems include a braking system in a car, an oxygen ventilation machine in a hospital, air traffic control systems, etc.

As RTESs become larger and more complex, the need for a more structured development process such as Model-Based Engineering (MBE) has been recognized by the research community (INRIA, ITEA, CEA LIST) [1, 2, 3]. In MBE all the development revolves around models. This approach allows for performance analysis to be done in the early stages of the development, predicting the system's performance before any implementation. This includes timing analysis - the most important part of performance analysis for RTESs. Timing constraints can refer to either logical or chronometric (physical) time. Multiform timing constraints are constraints for which physical dimension does not play any role, but the simultaneity and precedence between events gives the notion of time.

II. REVIEW OF MODELLING LANGUAGES FOR RTESS

There are many different modeling languages that can be used in MBE, but there is a lack of a standard language and methodology that will be used for RTESs in all domains.

This research has been supported by the EUROWEB Project funded by the Erasmus Mundus Action II programme of the European Commission

A modeling language can be:

- General Purpose Language (GPL)
- Domain Specific Language (DSL)

GPLs are used across all domains providing the same semantics for different kinds of systems. DSLs are designed through a collaborative process between software engineers and domain experts and are used on a particular domain.

A. UML

UML [4] is the best example of a GPL. Since version 2.0, UML supports time related meta-classes such as TimeExpressions, TimeObservations, and Durations. UML provides a simple way to model time.

The UML timing diagram is used to display the change in state or value in elements over time and it can be used to specify time-related behavior. There are two kinds of timing diagrams: state timing diagrams and general value timing diagrams. The state timing diagram shows the changes between different states (y-axis) over time (x-axis). The general value timing diagram shows the change of a value (displayed inside a diamond) over time.

UML's simple time model is not sufficient for the development of complex RTES.

B. SPT

UML's shortcomings regarding time modeling have triggered the creation of the **SPT (UML profile for Schedulability, Performance & Time)** [5]. However, SPT is based on UML 1.4 and is now obsolete.

It has been noted [6] as very useful for dealing with time and resources, using concepts such as: instant, duration, time based event, and stimuli. It introduced a time domain model partitioned into several groups of concepts for modeling: time and time values (TimeModel), events in time and time-related stimuli (TimedEvents), timing mechanisms such as clocks and timers (TimingMechanisms), and timing services in real-time operating systems (TimingServices).

However, although SPT provided a framework for representing time and time-related mechanisms, it revealed deficiency of expressive power and flexibility.

C. MARTE

MARTE (**UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems**) [7] was created as a replacement of SPT [8]. It is split into the following packages:

- the core MARTE Foundations extended by:
  - MARTE Design Model (used to model the features of an RTES)
  - MARTE Analysis Model, also called Real-Time & Embedded Analysis – RTEA (used to describe the analysis of system properties)
- MARTE Annexes containing profiles and model libraries

The MARTE Foundations package contains two packages for modeling time:

1. Time Package for time properties
2. NFP Package for Non-Functional Properties

The RTEA package contains the Performance Analysis Modeling (PAM) package which is useful for performance analysis. Note that MARTE provides facilities to annotate models with information required to perform specific analysis.

The MARTE Time Meta-model

MARTE defines a meta-model with concepts such as time structure, time access, and time usage. A TimeBase is the basic building block in a time structure. A time access is defined through the MARTE concept of a Clock. A clock has a defined Unit and an Event that happens on every clock tick. There are two predefined clocks in MARTE: a LogicalClock and a ChronometricClock. NFPs (skew, stability, etc.) are measured against a reference chronometric clock.

MARTE makes a clear distinction between a Duration Value, which is used to describe a span or an interval of time, and an InstantValue, used to reference a single time instant. A TimedElement in MARTE is any type of model element associated with a Clock. There are three main TimedElements in MARTE: TimedEvent, TimedProcessing, and TimedObservation. Most elements refer to either discrete or dense time specified by attributes such as TimeNatureKind.

The **MARTE Time Package** incorporates the concepts defined by the time meta-model. It provides new UML stereotypes, model libraries and expression languages such as Value Specification Language (VSL) for value expressions and Clock Constraint Specification Language (CCSL) for clock constraint expressions. MARTE ClockConstraints are specified as declarative statements and CCSL can be used for expressing the constraints. Most of the MARTE stereotypes extend a UML stereotype explicitly referring to a clock. The model libraries TimeTypesLibrary and TimeLibrary contain enumerations used to define various time properties.

Apart from the MARTE Time package, the Generic Resource Modeling (GRM) Package also defines time-related stereotypes, such as TimingResource. There are time-related NFP types defined in the NFP Package as well (e.g. NFP_Duration, NFP_Frequency).

D. SysML

Systems engineers have found UML too software-specific and they have created the **SysML (Systems Modeling Language)** [9]. SysML was created as a subset of UML 2.1.1 with several extensions. The extensions include two new diagrams (requirement diagram and parametric diagram), a new concept of a block replacing the UML class, neutral data formats, omission of software specific UML elements, etc. The lack of possibility to model non-functional requirements, such as timing requirements in UML, is addressed by the structured and standardized way of denoting requirements in the requirement diagram. SysML provides stereotypes such as «requirement», «deriveReqt», «satisfy», «verify», «refine», as well as a table notation for representing the requirements and their relationships.

The parametric diagram is used to define constraint relationships between the blocks and their internal properties including time.

E. AADL

The need for a language like **AADL (Architecture Analysis & Design Language)** has been recognized by the aerospace industry [10]. AADL has a component-based approach to system architecture and focuses on the components of the system, their interactions and their properties. It provides software models and mechanisms for analysis including early validation, safety analysis, etc. AADL has a textual, graphical and XML notation.

The runtime characteristics of software components can be declared in the AADL model. An AADL component is formed by a type and an implementation declaration. Properties such as the timing properties can be declared as part of the implementation declaration.

An AADL system specification consists of one or more property sets and one or more packages. A package has component types with implementation declarations and annex libraries. A system instance can be made based on the specification, and the AADL analysis mechanisms are performed on this instance. These analysis mechanisms are usually made by a specially designed tool (e.g. OSATE [11]).

Many AADL properties and property types have been pre-declared. They are divided into groups such as the Timing Properties group with properties for specifying timing.

F. AUTOSAR

In the past few decades the amount, size, and complexity of RTESs in the automotive industry has been growing exponentially. Today, an average automotive vehicle has as many RTESs (known as Electronic Control Units-ECUs) as an average aircraft had two decades ago [12].

AUTOSAR (**AUTomotive Open System ARchitecture)** is a partnership between many key players in the automotive industry [13]. One of its goals is to make a common method for developing software, and have companies compete on implementation. A globally accepted standard for hardware independent software makes the integration of software from multiple suppliers much easier and helps make software a commodity.

AUTOSAR defines an Application Layer, a Runtime Environment (RTE), and a Basic Software Layer (BSL) consisting of services, hardware abstraction, and drivers. The Complex Drivers Layer (CDL) is used for special purpose functionalities for sensors and actuators, such as real-time functionalities. Drivers for devices with strict timing constraints are part of the CDL.

AUTOSAR Timing Extension

Since release 4, the AUTOSAR specification incorporates the AUTOSAR Timing Extensions. The timing extensions in AUTOSAR are using two basic concepts: event (TimingEvent) and event chain (TimingEventChain). There are several different types of timing constraints:

- SynchronizationConstraint
- AgeConstraint
- LatencyConstraint
- ExecutionTimeConstraint

G. EAST-ADL

EAST-ADL (Electronics Architecture and Software Technology – Architecture Description Language) was created specifically for complex and advanced RTESs in the automotive industry [14]. EAST-ADL complements AUTOSAR providing higher levels of abstraction as well as analysis and lifecycle management support. It follows the guidelines of the low-level implementation that AUTOSAR provides, but adds a higher, more abstract modeling level based on concepts from UML, SysML and AADL. It also allows for the AUTOSAR model to be extended with timing and safety requirements, traceability, and verification and validation models.

EAST-ADL is organized in 4 abstraction layers. The highest VehicleLevel defines what the vehicle should do. It presents the user-visible content of the vehicle as seen from the outside. The Analysis, Design, and Implementation levels explain how these features should be accomplished. The AnalysisLevel focuses on the functionality of the RTES, providing functional abstractions about the system's behavior and algorithms. The DesignLevel focuses on providing a detailed hardware-oriented view of the RTES. Finally, the ImplementationLevel provides an AUTOSAR-based implementation of this view. The RTES is modeled in each level, and various traceability relations provide links between the levels. This allows for a feature to be traced all the way to the hardware implementation.

The modeling of both timing requirements and properties in the functional abstraction levels is done by using TADL (Timing Augmented Description Language). The implementation level uses AUTOSAR Timing Extensions.

TADL was defined by ITEA [15] to provide the description of timing information at higher levels of abstraction and to allow timing analysis to be done in early stages of the development of systems. An updated version - TADL2 was defined in order to align with the AUTOSAR Timing Extensions.

Timing in EAST-ADL (TADL)

There are several fundamental concepts in TADL:

- Events
- Event Chains
- Constraints

An event either causes an execution (then it is referred to as stimulus), or it is caused by an execution (a response). A stimulus always happens before a response. TADL has several predefined events such as: EventFunctionFlowPort, EventClientServerPort, or EventFunction.

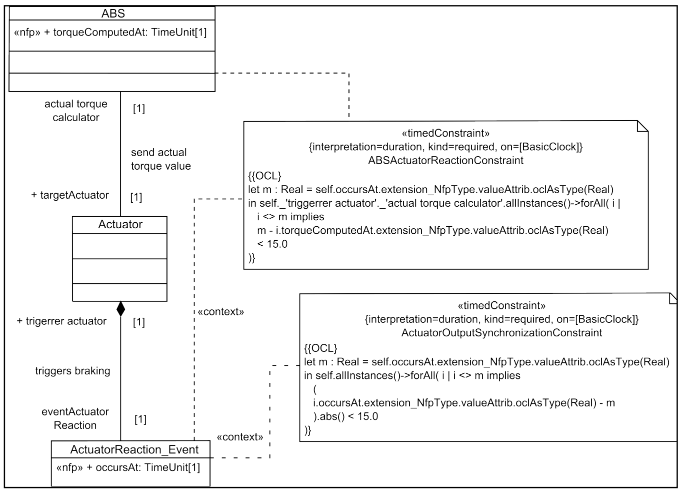
Event Chains are bindings of several events in order to show the relations between them. Events can be composed into an event chain, or an event chain can be decomposed into several events. The timing constraints can be set on an event or an event chain. The TADL timing constraints can be:

1. Event Constraints (set on an event)
    - Periodic
    - Sporadic
    - Pattern
    - Arbitrary
2. Offset Constraint (set on several events)
    - Age
    - Reaction
    - Input/output synchronization
3. Delay Constraints (set on an event chain)

III. COMPARISON OF MARTE AND EAST-ADL

A case-study of the development of a braking system in a car, done using EAST-ADL for modeling, was conducted to view EAST-ADL's capabilities in action [16]. The braking system was modeled in MARTE in order to compare the two languages. Our MARTE model used the EAST-ADL abstraction layers, producing use-case, class, and sequence diagrams on the vehicle, analysis, and design level.

In MARTE there is a lack of specific semantics for the types of timing constraints present in EAST-ADL, such as age, reaction, execution, etc. This has been overcome by adding new properties to the classes. These properties were referred to in the timing constraints using OCL (Object Constraints Language). Fig. 1 shows an excerpt of the class diagram on the design level where an OCL rule loops through all instances of the class *ABS* and makes sure that the value *torqueComputedAt* is at most 15 ms greater than the value *occursAt* of the *ActuatorReaction_Event* class. It also shows a synchronization constraint that makes sure that the *ActuatorReaction_Events* happen within 15 ms of each other, by looping through all instances and comparing the value of the property *occursAt*.


ABS
«nfp» + torqueComputedAt: TimeUnit[1]
actual torque calculator
[1]
send actual torque value
+ targetActuator
[1]
Actuator
+ trigerrer actuator
[1]
triggers braking
eventActuator Reaction
[1]
ActuatorReaction_Event
«nfp» + occursAt: TimeUnit[1]
«context»
«context»
«timedConstraint»
{interpretation=duration, kind=required, on=[BasicClock]}
ABSActuatorReactionConstraint
«timedConstraint»
{interpretation=duration, kind=required, on=[BasicClock]}
ActuatorOutputSynchronizationConstraint

```
{{OCL}
let m : Real = self.occursAt.extension_NfpType.valueAttrib.oclAsType(Real)
in self._'triggerrer actuator'._'actual torque calculator'.allInstances()->forAll( i |
   i <> m implies
   m - i.torqueComputedAt.extension_NfpType.valueAttrib.oclAsType(Real)
   < 15.0
)}
```

```
{{OCL}
let m : Real = self.occursAt.extension_NfpType.valueAttrib.oclAsType(Real)
in self.allInstances()->forAll( i | i <> m implies
   (
   i.occursAt.extension_NfpType.valueAttrib.oclAsType(Real) - m
   ).abs() < 15.0
)}
```

Figure 1. Excerpt of the MARTE class diagram on design level [17]

The constraints in our MARTE model used the MARTE TimedConstraint stereotype and were referring to a basic chronometric Clock or a multiform-time based logical Clock. The latter was used for constraints specified through distance such as "*The vehicle shall start to brake within 5 meters after the brake pedal is pressed*".

A comparison of the EAST-ADL timing constraints and the model in MARTE is shown in Table I.

TABLE I. TIMING CONSTRAINTS IN EAST-ADL AND MARTE

EAST-ADL Timing Constraint (value)	**MARTE Timing Constraint (value)**
Reaction (upper)	TimedConstraint («nfp» property:TimeUnit in the classes)
Execution (upper)	TimedConstraint (property:Real in the class)
Periodic (period)	TimedConstraint (property:Real in the class)
InputSynchronization (upper)	TimedConstraint («nfp» property:TimeUnit in the class)
OutputSynchronization (upper)	TimedConstraint («nfp» property:TimeUnit in the class)

IV. CONCLUSION

More and more modeling languages include concepts for handling timing. EAST-ADL, with the abstraction over the AUTOSAR timing concepts, offers an elegant way to model RTESs in the automotive industry. The UML-based MARTE is a very good alternative for RTESs in the automotive industry but can be used for any kind of RTESs.

EAST-ADL's main benefit is that it is easier to use for a modeler experienced with modeling automotive RTESs, especially with AUTOSAR. Most importantly, EAST-ADL offers semantics for different types of timing constraints which is not the case with MARTE.

A case study of an approach using EAST-ADL was conducted and the same problem was modeled in MARTE as part of this research. Our MARTE model could be used directly by tools such as Acceleo (to generate code) and Cheddar (to perform timing analysis). This would be an advantage for MARTE because the case study of the EAST-ADL approach showed there isn't any uniform way to do this in EAST-ADL.

REFERENCES

[1] INRIA. INRIA AOSTE. [Online]. http://www-sop.inria.fr/aoste/

[2] ITEA. ITEA Ofiicial Web Site. [Online]. https://itea3.org/

[3] CEA LIST. CEA LIST Official Website. [Online]. http://www-list.cea.fr/

[4] Object Management Group. UML 2.4.1. Specification, 2011 [Online] http://www.omg.org/spec/UML/2.4.1/

[5] Object Management Group, "SPT 1.1 Specification," Object Management Group, 2005. [Online]. http://www.omg.org/spec/SPTP/1.1/

[6] Bran Selic, "Models, Software Models and UML," in UML for Real: Design of Embedded real-Time Systems, Luciano Lavagno, Grant Martin, and Bran Selic, Eds. Boston, Massachusetts: Kluwer Academic Publishers, 2003, pp. 1 - 16

[7] Object Management Group. (2012) The OMG MARTE Web Site. [Online]. http://www.omg.org/omgmarte/

[8] Charles André, Frédéric Mallet, and Robert De Simone, "Time modeling in MARTE," in ECSI Forum on specification & Design Languages (FDL), 2007, pp. 268-273. [Online] http://www-sop.inria.fr/members/Frederic.Mallet/publis/2007/fdl07b.pdf

[9] Object Management Group, "SysML 1.3 Specification," 2012. [Online]. http://www.omg.org/spec/SysML/1.3

[10] Peter H Feiler and David P Gluch, "Model-Based Engineering with AADL; An Introduction to the SAE Architecture Analysis & Design Language".: Addison-Wesley, 2013

[11] SAE International. OSATE AADL tool. [Online]. http://www.aadl.info/aadl/currentsite/tool/osate.html

[12] Hans Blom et al., "EAST-ADL – An Architecture Description Language for Automotive Software-Intensive Systems," White Paper, 2012 [Online] http://www.maenad.eu/public_pw/conceptpresentations/EAST-ADL_WhitePaper_M2.1.10.pdf

[13] AUTOSAR, "Specification of Timing Extensions," 411,. [Online]. http://www.autosar.org/download/R4.0/AUTOSAR_TPS_TimingExtensions.pdf

[14] EAST-ADL, "EAST-ADL Domain Model Specification," 2013. [Online]. http://www.east-adl.info/Specification/V2.1.11/EAST-ADL-Specification_V2.1.11.pdf

[15] Hans Blom, et al. "Annotation with timing constraints in the context of EAST-ADL2 and AUTOSAR–the timing augmented description language". In Proc. Workshop on the Definition, Evaluation, and Exploitation of Modelling and Computing Standards for Real-Time Embedded Systems, Dublin, Ireland, 2009, (pp. 2-5)

[16] Damjan Temelkovski, Ljerka Beus-Dukic, "Model-Based Engineering in Real-Time Embedded Systems: Specifying Timing Constraints", to be published, 6th ICT Innovations Conference, Ohrid, Macedonia, 2014

[17] Damjan Temelkovski, Ljerka Beus-Dukic, "Model-Based Engineering in Real-Time Embedded Systems: Specifying Timing Constraints" – research project report, University of Westminster, 2014