Schedulability analysis support for automotive systems

Kang, Eun-Young; Schobbens, Pierre Yves

Published in:
Proceedings of the ACM Symposium on Applied Computing

DOI:
10.1145/2554850.2554929

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

Link to publication

Citation for pulished version (HARVARD):
Kang, E-Y & Schobbens, PY 2014, Schedulability analysis support for automotive systems: From requirement to implementation. in Proceedings of the ACM Symposium on Applied Computing. ACM Press, New York, NY, USA, pp. 1080-1085, 29th Annual ACM Symposium on Applied Computing, SAC 2014, Gyeongju, Korea, Republic of, 24/03/14. https://doi.org/10.1145/2554850.2554929

Schedulability Analysis Support for Automotive Systems: From Requirement to Implementation

Eun-Young Kang and Pierre-Yves Schobbens
PReCISE Research Centre,
Dept. of Computer Science, Namur Univ., Belgium
{eykang,pyschobb}@fundp.ac.be

ABSTRACT

Modeling and analysis of precise non-functional properties, such as energy and timing constraints, is key to the correct development of automotive systems. Automotive applications development cost, in particular, is impacted by incorrect design made at the early development phases but only detected later, often after implementation. This late detection of design errors leads to additional cost. In this paper, we propose a model driven approach to perform non-functional properties verification and to enable scheduling analysis of automotive systems at the very early design level. The different phases of a design range from the requirements to a model allocated on a specific execution platform: EAST-ADL and MARTE are used together to specify the structure and energy/timing constraints of the software, as well as the hardware parts of the system. To prove the correctness of specification and perform the scheduling analysis, the semantics of the constraints is given as mapping to a formal interchange format XFG (eXtended Function-block Graphs) language. The XFG models are then automatically translated into priced timed automata for model checking. This later transformation is supported by a tool chain called A-BeTA. We demonstrate the applicability of our approach on the Brake-By-Wire case study.

Keywords

Embedded real-time systems, MBD, EAST-ADL, MARTE, Model-checking

1 Introduction

In the automotive domain, Model-based development (MBD) is used extensively for the development of safety-critical and resource-constrained embedded control systems, where models are designed and analyzed to establish their (non)functional properties and refined to concrete and executable systems. The use of formal methods is the cornerstone to ensuring model correctness and preservation of properties through refinements.

EAST-ADL (Electronics Architecture and Software Technology-Architecture Description Language) [6, 16], aligned with AUTOSAR (Automotive Open System Architecture) standard [3], is a concrete example of the MBD approach for the architectural modeling of safety-critical automotive embedded systems. The recent EAST-ADL has adopted the timing model proposed in the Timing Augmented Description Language (v2) (TADL2) [22]. TADL2 expresses and composes the basic timing constraints, i.e., repetition rates, end-to-end delays, and synchronization constraints. The time model of TADL2 specializes the time model of MARTE, the UML profile for Modeling and Analysis of Real-Time and Embedded systems [19]. MARTE provides CCSL, a time model and a Clock Constraint Specification Language, that supports specification of both logical and dense timing constraints for MARTE models, as well as functional causality constraints [17].

In current practice, automotive applications development starts with high level models which capture the software (SW) and hardware (HW) models separately. Models are refined in several steps by adding details of functionalities and HW characteristics. The HW platform can be introduced and refined like other parts of the system, or developed independently and allocated by the SW elements. At the end of these refinements, schedulability techniques can be applied on models by extracting the information needed by scheduling analysis [8]. Schedulability techniques and energy/timing constraints validation of the system are still considered late in the development process. However, applying preliminary analysis on the system at the very beginning of the design phase should also be possible, i.e., before including any details of the HW platforms. Such analysis allows designers to detect unfeasible SW architectures, prevents costly design mistakes, and provides an analytical basis to assess design tradeoffs associated with resource optimization. Once the system is refined enough, it must be possible to extract sufficient information, which can be used as an input of classical scheduling analysis.

In this paper, we propose a model driven approach to support schedulability analysis for automotive systems as well as perform (non)functional properties verification during the early design phases. In fact, the methodology presented here has the particular objective of completing EAST-ADL models to enable scheduling analysis at the design level. Our approach is based on the combination of the aforementioned modeling languages, EAST-ADL augmented with timing constraints specified in TADL2, and UML MARTE including CCSL as a means to express: 1. the timing requirements and functional causality constraints; 2. the energy-aware real-time (ERT) behaviors of the system (e.g., resources – CPU load or memory usage – for each component of the system during its execution); 3. the constraints introduced by the execution platform.

Furthermore, to explicitly annotate and reason about the ERT behaviors (for step 2), we adopt UML diagrams and extend them with a UML profile, called *XFG profile* [10, 11, 12], which integrates relevant concepts from EAST-ADL and MARTE profiles. Such profiled models are translated into a formal interchange format, eXtended Function-block Graphs (XFG) language [13] that allows a formal capture of ERT semantics. With knowledge of the HW platform and the allocation of functionalities onto ECUs (Electronic Control Units), other constraints can be incrementally added to model ECU clock and allocation/preemption of ECU w.r.t functionalities execution (for step 3). The formal semantics of EAST-ADL/TADL2 constraints defined in CCSL during steps 1 and 3 is also given as mapping to the XFG language.

The resulting XFG models are then translated again into analyzable Priced-Timed Automata (PTA) [4] models. PTA is an extension of Timed Automata (TA) [2] augmented with prices on loca-

http://dx.doi.org/10.1145/2554850.2554929

tions and transitions, where the accumulation of resources is represented by continuous price variables. It is then possible to check the validity of the constraints against the requirements at any step of the refinement and formally verify the correctness of the system design at the model level to detect possible deadlock or requirement violation by means of the model checker UPPAAL-CORA[1]. The translation processes are supported in fully automatic by using the tool, A-BeTA (Aβ: EAST-**A**DL **Be**havioral Modeling and **T**ranslation into **A**nalyzable Model), developed by Kang et al [12, 14]. We show the proposed methodology at work on an industrial prototype, the Brake-By-Wire system (BBW).

The paper is organized as follows: The BBW running example is introduced in Sec.2. Sec.3 gives a brief overview of XFG and MARTE with CCSL. Sec.4 describes the mapping between EAST-ADL/TADL2 with MARTE to XFG and shows how our modeling approach provides support for scheduling analysis at the design level. We demonstrate the applicability of our method by performing resource-aware analysis and verification on the BBW case study in Sec.5. Sec.6 presents related work. The conclusion and future work are presented in Sec.7.

2 Running Example: Brake-By-Wire

A BBW application is provided by Volvo in the TIMMO-2-USE, MAENAD-FP-7 projects together with EAST-ADL Association [23, 16]. Fig.1 shows the structure of the BBW functionality that is viewed as the Functional Design Architecture (FDA) in EAST-ADL: the Function Prototypes f_ps denote subfunctions and the *connectors* represent data flows and dependencies.

The functionality of BBW consists of the following f_ps (components): The Brake Pedal Sensor (BPS) reads the pedal position on port *EISignal*; The Brake Torque Calculator (BTC) receives the pedal position from BPS and computes the desired global torque; The Wheel Sensors (WS) read the speed values of each wheel, *FrontRightWheelSensor* (FRWS), *FrontLeftWheelSensor* (FLWS), *RearRightWheelSensor* (RRWS), *RearLeftWheelSensor* (RLWS); The Global Brake Controller (GBC) receives the speed values from WS and the global torque from BTC. GBC computes the desired brake torque required for each wheel; The ABS, consisting of *ABSonFrontRightWheel* (AFRW), *ABSonFrontLeftWheel* (AFLW), *ABSonRearRightWheel* (ARRW), and *ABSonRearLeftWheel* (ARLW), adapt the brake force on each wheel if the speed of one wheel is significantly less than the estimated vehicle speed and controls the wheel braking to prevent locking of the wheels; The Actuators (ACT) for *FrontRightBrake* (FRBA), *FrontLeftBrake* (FLBA), *RearRightBrake* (RRBA), *RearLeftBrake* (RLBA) apply brake force on each wheel.

We consider the following Timing constraints on top of the BBW EAST-ADL model, `Delay`, `Synchronization`, and `Repeat` constraints, which are sufficient to capture the constraints described in Fig.1. For further details of TADL2 specification and complete EAST-ADL timing constraints of the BBW example, refer to [21, 22, 23, 7]:

TC1: Four **delays** $X1$ are measured from BPS to ACT. They are bounded with a minimum value of 70 ms and a maximum value of 120 ms. TC1 specifies bounds for event chains such as stimulus (*EISignal* inport on BPS) and the corresponding response (*EISignal* outport on four ACTs).

TC3: A **periodic** acquisition of WS must be carried out with a **repetition** constraint of $X3 = 10$ ms.

TC4: The **delay** $X4$ applied on GBC (*Rpm* ports and *Torq* ports on GBC) is 40 percent of the initial time budget $X1$.

TC7: The first and last wheel brake actuators (*EISignal* ports on ACT) must occur within a given time window, i.e., the tolerated maximum constraint $X7 = 5$.

9[1]The model checker UPPAAL-CORA is a branch of the UPPAAL tool for cost optimal reachability analysis. It accepts PTA as its input modeling language. www.uppaal.org

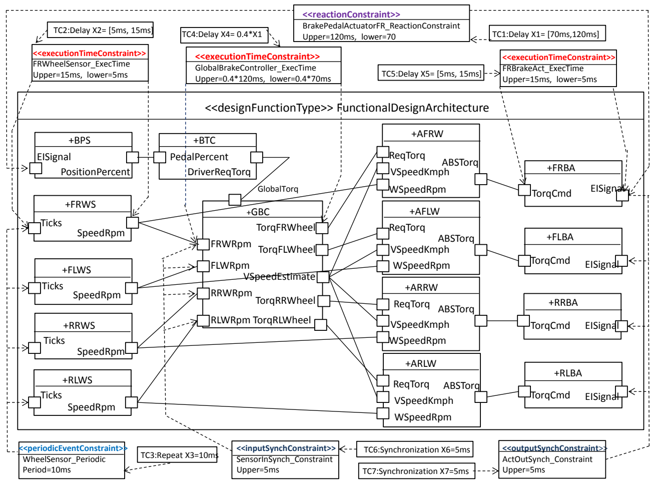


Figure 1: BBW in EAST-ADL augmented with TADL2 timing constraints

According to the EAST-ADL meta-model, the timing constraint describes a design constraint, but has the role of a property, requirement or validation result, based on its `Context` [6]. The TADL2 meta-model is integrated with the EAST-ADL meta-model and is supplemented with structural concepts from EAST-ADL. The TADL2 constraints contain the identifiable state changes as *Events*. The causality related events are contained as a pair by *EventChains*. Based on *Event* and *EventChains*, data dependencies, control flows, and critical execution paths are represented as additional constraints for the EAST-ADL functional architectural model, and apply timing constraints on these paths.

`DelayConstraint` gives duration bounds (minimum and maximum) between two events *source* and *target*, i.e., period, end-to-end delays. This is specified using *lower, upper* values given as either `ExecutionTimeConstraint` (see TC2, TC4, TC5) or `ReactionConstraint` (TC1) in EAST-ADL.

`SynchronizationConstraint` describes how tightly the occurrences of a group of events follow each other. All events must occur within a sliding window, specified by the *tolerance* attribute, i.e., a maximum time interval allowed between events, given as `Input/Output SynchronizationConstraint` in EAST-ADL (TC6, TC7).

`RepeatConstraint` states that the period of the successive occurrences of a single event must have a length of at least a *lower* and at most an *upper* time interval. The interval is given as `PeriodEvent Constraint` in EAST-ADL (TC3)

TimeBase, Dimension, Unit, TimeBaseRelation in TADL2 A discrete and totally ordered set of event occurrences (often an event occurrence is called *tick*) is represented as *TimeBase*. The type of *TimeBase* is a *Dimension* with a kind that represents the nature of *Timebase*. *Dimension* defines the set of units to express the duration measured on a given *TimeBase*. Each *Unit* is related to another *Unit* with *factor, offset,* and *reference* to enable conversions. Since *TimeBase* is a discrete set of instants, a discrete step is specified with *precisionFactor* and *precisionUnit* [21].

For example (see Listing 1), the *physicalTime* dimension has three units where $1second = 10^6 micros$ and $1ms = 10^3 micros$, and *universalTime* is declared based on *physicalTime*. The drifts between different *TimeBases* can be specified with *TimeBaseRelation*, i.e., $ecu1$ is an ECU *TimeBase*, and tbr states that $ecu1$ has a drift of 0.1 ms for each ms compared to *universal_time*.

```
Dimension physicalTime {
  Units {micros{factor 1.0 offset 0.0},
         ms{factor 1000.0 offset 0.0 reference micros},
         second{factor 1000000.0 offset 0.0 reference
             micros}}}
  TimeBase universal_time {dimension physicalTime
       precisionFactor 1 precisionUnit micros}

  TimeBase ecu1 {dimension physicalTime precisionFactor
        0.1 precisionUnit micros}

  TimeBaseRelation tbr {(1.0 ms on universal_time) =
       (1.1 ms on ecu1)}
```

Listing 1: Dimension, TimeBase, and TimeBaseRelation

3 XFG Language and MARTE/CCSL

This section presents the background, XFG and MARTE/CCSL, as a means to formally capture the semantics of the EAST-ADL/TADL2 constraints and ERT behaviors of the system, and define a time model based on the constraints, as used in our approach.

3.1 XFG Language

An interchange format XFG (eXtended Function-block Graphs) language is based on Hybrid Automata [1]. It provides support for formal modeling and analysis of ERT behaviors of f_ps in EAST-ADL [11, 13]. An XFG system consists of a finite number of XFG processes and the control part of any process is described as a finite state machine. The processes communicate by *channels* (e.g., $a!$ and $a?$) with *rendezvous* or broadcast semantics. Transitions of an XFG process can be marked as urgent with a small dot (see Fig.2) implying that they should be taken without any time elapse. To formally define the semantics of XFG, we introduce the following terms. A finite set of variables V. A set of clock variables $V_c \subseteq V$. *Expr* a set of value expressions (over the set V of variables) and *Bexpr* $\subseteq$ *Expr* a subset of Boolean expressions. A universe *Val* of values includes the set $\mathbb{R}^{\geq 0}$ of non-negative real numbers and the Boolean values. For a valuation $\rho : V \rightarrow Val$ and $\delta \in \mathbb{R}^{\geq 0}$, $\rho[+\delta]$ denotes the increment of each clock in V_c by δ: $\rho[+\delta](v) = \rho(v) + \delta$ if $v \in V_c$, otherwise $\rho[+\delta](v) = \rho(v)$. For simplicity, we define a core syntax for an XFG system and its semantics is considered a single global transition system in this paper.

DEFINITION 1. *An* XFG *is a tuple* $\langle Dtype, Init, L, l_0, I, h, E, U, Eng\rangle$ *where* $Dtype : V \rightarrow \{disc, cont, clock\}$ *assigns each variable a dynamic type: discrete, continuous, or clock. The sets* V_{disc}, V_{cont}, V_c *are defined as* $V_t = \{v \in V \mid Dtype(v) = t\}$ *for* $t \in \{disc, cont, clock\}$; $Init \in Bexpr$ *indicates the initial condition; A set of dotted variables* $\dot{V} \subseteq V_{disc}$ *represents different rates of increasing energy; L is a finite set of locations,* $l_0 \in L$ *is the initial location;* $I : L \rightarrow Bexpr$ *assigns an invariant to each location; H is a finite set of synchronizing action labels;* $E \subseteq L \times Bexpr \times 2^{V\times Expr} \times (H \cup \tau) \times L$ *is a set of transitions, represented as tuples* $\langle l, g, h, u, l'\rangle$ *where* $l \in L$ *is the source location,* $g \in Bexpr$ *is the guard,* $h \in H$ *is a synchronization label* $\{h!x, h?v | x \in Expr, v \in V\}$*, where x and v are sequences of expressions or variables,* $u \subseteq V \times Expr$ *is an update;* $U \subseteq E$ *identifies the subset of urgent transitions;* $Eng : L \cup E \rightarrow \mathbb{R}^{\geq 0}$ *assigns an energy consumption to each location and transition*

DEFINITION 2. *The operational semantics of an* XFG *is given as a transition system* $\langle S, s_0, T\rangle$*, where* $S = \langle l, \rho\rangle \in L \times (V \rightarrow Val)$*,* $s_0 = \langle l_0, \rho_0\rangle$*, where* ρ_0 *evaluates to zero for all clocks,* $T \subseteq S \times (E \cup \mathbb{R}^{\geq 0}) \times S$*. There are two kinds of transitions: 1. for any* $e = \langle l, g, h, u, l'\rangle \in E$, $\langle l, \rho\rangle \xrightarrow{e} \langle l', \rho[u]\rangle$*; 2. for any* $\delta \geq 0$, $\langle l, \rho\rangle \xrightarrow{\delta} \langle l, \rho[+\delta]\rangle$*, provided* $I(l)$ *is continuously true. To each such step, we associate an energy consumption defined by 1.* $Eng\big(\langle l, \rho\rangle \xrightarrow{e} \langle l', \rho[u]\rangle\big) = Eng(e)$*; 2.* $Eng\big(\langle l, \rho\rangle \xrightarrow{\delta} \langle l, \rho[+\delta]\rangle\big) = Eng(l) \cdot \delta$*.*

The energy consumption of π is the accumulated consumption of steps along the run.

3.2 MARTE/CCSL

The UML MARTE profile provides CCSL that supports a specification of a generic timed interpretation for the *Time Model* through the notion of *clocks*. A clock (not to be confused with the XFG clocks) denotes particular UML events on which we impose a constraint. Clocks are ordered sets of event occurrences. CCSL, a non-normative annex of MARTE, specifies causal and timed constraints on clocks. Constraints are either a relation or an expression associated with modeling elements. We describe here only the clock relations pertinent to our running examples. For further details of CCSL, refer to related works [17, 18].

`discretizedBy`: A logical clock *micros* models a discrete clock of period 1 microsecond based on a dense clock, called *IdealCLK* (relative to the unit *second*) and pre-defined in MARTE. Each *Unit* given in a *TimeBase* in TADL2 is derived by discretizing *IdealCLK*. Eq.1 defines a discrete clock for the *micros* unit of *physicalTime* in *universal_time* in Listing 1.

$$micros \ = \ IdealClock \ \texttt{discretizedBy} \ 0.000001 \qquad (1)$$

`isPeriodicOn`: A slower clock, i.e., a subclock of a discrete clock is defined. Eq.2 gives an example of a clock relation that defines a subclock of the *micros* with period 1000 ($factor$ of the *ms* unit) for the *ms* unit of *universal_time*.

$$ms \ \texttt{isPeriodOn} \ micros \ period \ 1000 \qquad (2)$$

"a `precedes` b" specified that the event occurrent a must be observed before b (symbolically denoted by $a \prec b$ strictly or $a \preceq b$ non-strictly). CCSL also provides expressions to build new clocks from existing ones. For instance, "`inf`(a,b)" builds a new clock which is the slowest clock faster than a and b. Similarly, "`sup`(a,b)" builds the fastest clock which is slower than a and b. Finally, "$c = a$ `delayedFor` b" builds a delayed clock c whose ticks correspond to every n^{th} tick of b following a tick of a. The `exclusion` relation (denoted #) prevents two clocks from ticking simultaneously.

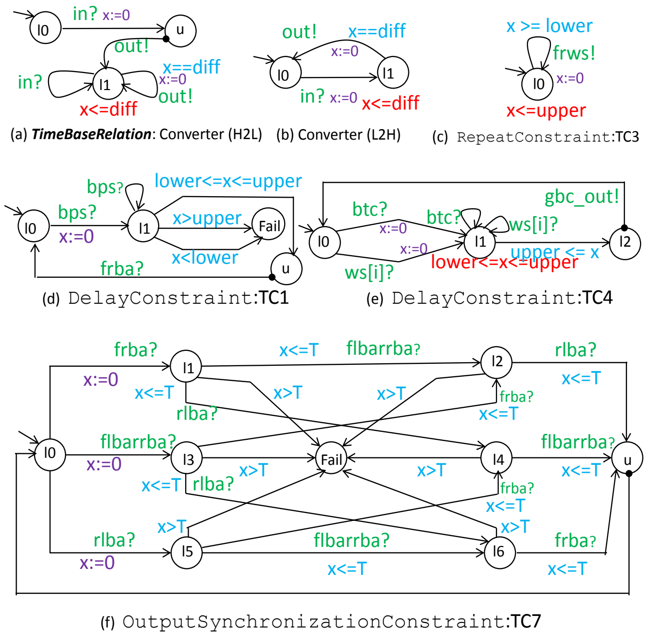
4 Mapping EAST-ADL/TADL2 with MARTE/CCSL To XFG

To perform the formal verification and scheduling analysis, the semantics of EAST-ADL/TADL2 constraints is captured in XFG. Prior to the semantic capturing, those constraints are defined in CCSL since it supports both kinds of constraints available in EAST-ADL/TADL2 such as causal (event chains) and temporal (delay, synchronization, repeat) constraints. We then map the constraints defined in CCSL to XFG to enable analysis of EAST-ADL/TADL2 specifications. The mapping follows three XFG modeling steps, *Timing requirement and functional causality modeling*, *(Non)-functionality modeling*, and *Resource allocation modeling*.

4.1 Requirement modeling: Timing and Functional Causality Constraints

We first describe XFG modeling of *TimeBase, Dimension*, and *Unit* in TADL2 (presented with CCSL clock in Sec.3.2). Then we show that timing and functional causality TADL2 constraints, defined in CCSL, can be modeled as XFG. The derived XFG using the Aβ tool chain can be either its textual or graphical format. The *Dimension* and *Unit* associated to the *TimeBase* are implicitly represented as a single *step* of time progress in XFG's *clock*, inspired by the work in [20]. The XFG model of a *TimeBase* (`universal_time` defined in Listing 1) consists of one location and one outgoing transition whereby the *Dimension*, i.e., *physical_time* and the duration of time unit *ms* (Eq.1 and 2) are implicitly represented by the *clock* variable 'x'. *clock* resets every time the transition is taken. The duration of a time unit is represented by the invariant $x \leq 1$ and the guard $x \geq 1$ at the location, i.e., a single step of the discrete time progress (tick) of *universal_time*.

A *TimeBaseRelation* is modeled as a signal reconstruction XFG, i.e., it describes the effect of converting a low-frequency signal to a high one (and vice versa) by holding each sample value for one sample interval (see Fig.2.(a) and (b)). In fact, two XFG processes with different periodicities synchronized using ports can theoreti-



(a) ***TimeBaseRelation***: Converter (H2L) (b) Converter (L2H) (c) `RepeatConstraint`:TC3

(d) `DelayConstraint`:TC1 (e) `DelayConstraint`:TC4

(f) `OutputSynchronizationConstraint`:TC7

Figure 2: Time Constraints as XFG

cally lead to a deadlock due to timing-frequency mismatch. To prevent such an issue, for example, the XFG$_{L2H}$ converter (Fig.2.(b)) is used when the sender is running at a lower-frequency (slow *TimeBase*) than the receiver. The *in?* (*out!*) corresponds to the input (output) from the sender (receiver). The frequency difference $diff$ parameter is calculated by dividing the *TimeBase* of the receiver with that of sender. In case the sender has a high-frequency, the XFG$_{H2L}$ converter (Fig.2.(a)) is used.

`DelayConstraint`: TC1, seen as 'reaction delay', given in Sec.2 is defined as in Eq.3 with corresponding CCSL clocks, i.e., the permissible delay between the source event activation (on BPS f_p inport), namely *bps*, and the target event actuation (on AFRB f_p outport), namely $frba$ (see Fig.1). The XFG modeling of Eq.3 is presented in Fig.2.(d). The input *rendezvous* actions, *bps*? and $frba$?, correspond to the events `ReceivingSignalEvent` and `SendingSignalAction` on the in- and outports of source and target, BPS and AFRB f_ps, respectively. When *bps* occurs, the transition is taken from l_0. Using clock x, the delay (permissible reaction time), specified as $lower \leq x \leq upper$ on the outgoing edge of l_1, is allowed before the target event $frba$. This XFG monitors the time distance between the two f_ps. If the distance is out of the reaction time bound [70ms,120ms], it enters the *Fail* location.

$$Lower \preceq frba \preceq Upper \tag{3}$$

TC4, another type of `DelayConstraint` seen as 'execution delay', is defined similar to Eq.3 replacing $frba$ by gbc_out, where gbc_out is the CCSL clock for the event `SendingSignalAction` of GBC f_p, $lower = 0.4*70ms$, and $upper = 0.4*120ms$, i.e., the permissible execution delay of GBC. The XFG modeling of TC4 is presented in Fig.2.(e). The 'input?' and 'output!' *rendezvous* actions correspond to 'receiving' and 'sending' events on the in- and outports of GBC respectively. When the input events of GBC (btc and $ws[i]$) occur, the edge is taken from l_0. btc is the event receiving the global torque value from BTC and $ws[i]$, $i = 0,1,2,3$, is the event receiving each speed value measured from the WS f_ps group. For simplicity, we use $ws[i]$ to state four events which read torque signals from the event group. To incorporate more events, additional edges corresponding to different combinations of occurrences can be added between l_0 and l_1. The execution delay, specified with the invariant $lower \leq x \leq upper$ and the guard $x \geq upper$, is allowed before any output event of GBC, $gbc\_out!$.

`RepeatConstraint`: The periodic interval between occurrences of an event is defined by the attributes *lower, upper* and *span* in TADL2 and is obtained from `PeriodicEventConstraints` in EAST-ADL. In the case of $span = 1$, the *lower* and *upper* attributes are equal, which means that the accepted behaviors must be strictly periodic. TC3 denotes the strict periodic nature of reading the sensor value, for one of the four wheels. Eq.4 gives the corresponding CCSL clock constraints for TC3, where $frws$ is the event clock of FRWS f_p. The XFG in Fig.2.(c) enforces the event $frws$ to tick between the corresponding ticks of *lower* and *upper*. Since $span = 1$ in TADL2, and $lower, upper$ have the same period, $frws$ ticks every 10 ticks of ms.

$$frws \ \mathtt{isPeriodicOn} \ ms \ period \ 10 \tag{4}$$

`SynchronizationConstraint`: TC7 describes the output synchronization constraint among the four brake actuators that must occur within the attribute *tolerance*, 5 ms. Eq.5 defines TC7 in CCSL and states that the slowest event must not tick later than 5 ms after the respective tick of the fastest event. Four events ($frba$, $flba$, $rrba$, $rlba$) occur on the outports of the corresponding f_ps (AFRB, AFLB, ARRB, ARLB) respectively (see Fig.1). `sup`/`inf` is the fastest/slowest clock of all clocks slower/faster than $frba, flba, rrba, rlba$. Thus, the constraint considers the interval from the earliest event to the latest event.

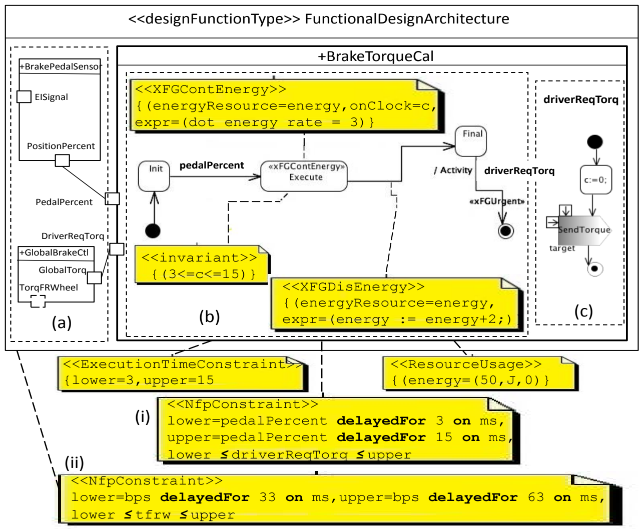
$$\mathtt{sup} \preceq (\mathtt{inf} \ \mathtt{delayedFor} \ 5 \ on \ ms) \tag{5}$$

The corresponding XFG in Fig.2.(f) specifies the time width within which a group of "response" events should occur: The actions $frba?, flba?, rrba?, rlba?$ receive each signal from the ABS f_ps group. The parameter T (represented as *tolerance* 5 ms) determines the maximum time allowed between the four responses. For simplicity, we assume $flba$ and $rrba$ occur successively in a strict order and that it is denoted as a single action $flbarrba?$. The input synchronization constraint is similar to the output synchronization except that instead of "response", "stimuli" are constrained to occur in a specified time width.

4.2 (Non)-functionality Modeling: ERT Behaviors of f_ps

We present the modeling of the structural and ERT behavioral view of the system. A part of the BBW units (depicted in Fig.3) is adopted as a running example: EAST-ADL FDA is composed of three connected f_ps, BPS, BTC, and GBC. The architecture of each f_p is presented as a composite structure diagram in Fig.3.(a). The behavior of each f_p is modeled by means of a UML state machine (SM) like the one depicted in Fig.3.(b). Each f_p's attributes constraints are expressed with three kinds of stereotypes: we use the EAST-ADL stereotype 'ExecutionTimingConstraint' and a part of the MARTE stereotype 'ResourceUsage' to define clock constraints and energy consumption. We also use the 'NfpConstraint' stereotype to associate CCSL constraints to an SM. Furthermore, we define a UML profile extending SM, *XFG profile*, based on the three stereotypes to model ERT and urgency semantics.

Thus, in Fig.3, we present the XFG-profiled ERT behavioral specification of BTC: Invariants (complying with timing and energy constraints), guards, and effects on transitions are expressed as XFG profiles according to the XFG language previously introduced. Each state of an SM is associated with a time constraint. This time constraint is modeled in terms of a clock-related XFG expression and attached to the state as its invariant. For a state where the system may consume energy continuously, the XFG profile offers the `XFGContEnergy` stereotype. It allows the specification of a consumption rate (`rate`), reference to the clock (`clock`) and energy attribute concerned, and an expression (`expr`) linking these elements as depicted by the comment attached to `Execute` state. Transitions model action behaviors ("effect") occurring between states and are controlled by guards. Choices are used to form complex paths between states. Receiving/sending signal from/to in-



<<designFunctionType>> FunctionalDesignArchitecture
+BrakeTorqueCal
<<XFGContEnergy>> {(energyResource=energy,onClock=c, expr=(dot energy rate = 3)}
<<invariant>> {(3<=c<=15)}
<<XFGDisEnergy>> {(energyResource=energy, expr=(energy := energy+2;)}
<<ExecutionTimeConstraint>> {lower=3,upper=15
<<ResourceUsage>> {(energy=(50,J,0)}
<<NfpConstraint>> lower=pedalPercent delayedFor 3 on ms, upper=pedalPercent delayedFor 15 on ms, lower ≤driverReqTorq ≤upper
<<NfpConstraint>> lower=bps delayedFor 33 on ms,upper=bps delayedFor 63 on ms, lower ≤tfrw ≤upper
(a)
(b)
(c)
(i)
(ii)

Figure 3: (a) Composite Structure Diagram, (b) CCSL-extended XFG-profiled State Machine, (c)Activity Diagram; (i) Execution Delay Stereotype, (ii) Reaction Delay Stereotype

/outports is modeled using `ReceiveSignalEvent`/ `SendSignalAction` respectively. For "signal sending" we use UML activities allowing detailed behaviors in Fig.3.(c). Energy consumption on transitions is discrete and is modeled by the `XFGDiscEnergy` stereotype. *Urgent* transitions can be decorated via `XFGUrgent`.

Based on the aforementioned mapping strategy as well as using the Aβ tool: 1. each f_p structure and its associated ERT behavior given as CCSL-extended XFG-profiled UML models are automatically translated into the corresponding XFG models; 2. FDA consisting of a set of f_ps is expressed in composite structural diagrams in Fig.3.(a). This FDA is converted to an XFG system, $\mathscr{X}_{SYS}$, which is seen as a network XFG consisting of a set of XFG, i.e., $\mathscr{X}_{SYS}$ = XFG$_{BPS}$ || XFG$_{BTC}$ || XFG$_{GBC}$ in our example. Thus, the ERT behavior of the entire system can be modeled in XFG. The XFG$_{BTC}$ in Fig.4.(b) is the ERT behavior of BTC f_p and is observed as an energy constrained model of the 'execution delay' XFG in Fig.2.(b) with one in/output event *pedalPercent*?, *driverReqTorq*!. The `Execute` location represents the ERT behavior execution of BTC. Continuous energy is consumed within that location until the execution period (complying with `NfpConstraint` in Fig.3.(i)) is finished. The outgoing edge from `Execute` deals with the special case of discrete energy updates. The associated CCSL stereotype for the 'reaction delay' between *BPS* and *GBC* f_ps (Fig.3.(ii)) is converted to an observer XFG, XFG$_{Obs}$. This XFG$_{Obs}$ is similar to XFG$_{TC1}$ (Fig.2.(c)) except that instead of the $frba$ event on AFRB f_p outport, the target event *TorqueFrontRightWheel* ($tfrw$) on GBC f_p outport is constrained to occur in a specified time width ($33 \leq x \leq 63$) after its source event *bps* activation. Such an XFG$_{Obs}$ is syntactically added to $\mathscr{X}_{SYS}$, we then verify if the observer *Fail* location can be reached in parallel with the actual main system $\mathscr{X}_{SYS}$ via the *rendezvous* communication *channels*.

4.3 Resource Allocation Modeling

We further refine the BBW system design toward a more precise description of the system and its execution platform. The BBW f_ps of FDA, allocated on the execution platform, is represented. We simply abstract the execution platform by a set of ECUs, however, more complex models with buses and their specific communication delays or with shared resources can also be considered. In our example, BPS, BTC, and GBC f_ps are allocated to a single ECU, this allows us to refine the system by considering preemption, delay, and precedence constraints. An ECU is associated to a time base

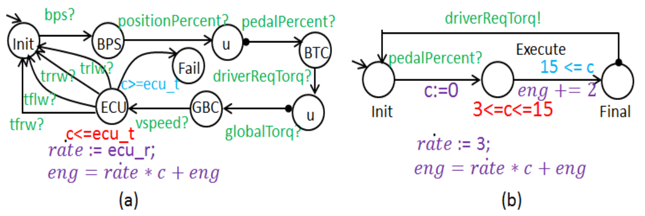


Figure 4: (a) ECU allocation (b) ERT behavior of BTC

ms, and the execution duration of each f_p is measured on this time base. Three f_ps allocations on a single ECU implies a sharing of the ECU clock by the f_ps. Thus, three clocks P_1, P_2, and P_3 are defined in CCSL and that models the actual execution of each f_p on an ECU, i.e., $\forall$ f_p i, j, k executing on ECU, and $i \neq j \neq k$: P_i # P_j # P_k, where only one f_p can be executed at a time on a specific ECU, the P_i, P_j, P_k clocks of single ECU are in exclusion of each other. These clocks are then time bases for the f_ps. As a result, the execution duration (delay) d of an f_{p_i}, is measured on the clock P_i and the P_i clock represents the ECU time allocated to f_{p_i}. The ECU clock is then the *union* over its allocated f_ps, of their P_i clock, i.e., for P_1, P_2, P_3 in the system: $ecu\_t = P_1$ `union` P_2 `union` P_3.

For each operation of the end-to-end flows of f_ps, its execution time is specified ([$lower$, $upper$], $unit = ms$). We note how the end-to-end flow gives an immediate insight into the system behavior in terms of timing and energy constraints. The successor/predecessor relation and the activation event show that the deadline to be respected for each end-to-end flow is the cycle duration and that the sum of all execution times/energy consumption during the execution in each end-to-end flow needs to be lower than or equal to the cycle duration to have a schedulable system. Fig.4.(a) shows the XFG modeling of ECU allocation, XFG$_{ECU}$ with timing, energy, and precedent event constraints. In our example, we assume memory is a critical energy (resource) that needs to be checked. According to the aforementioned exclusion definition in CCSL, and the predecessor relations in EAST-ADL/TADL2 (Fig.1 & 3), the XFG$_{ECU}$ specifies precedent constraints on the order of execution of events where the allocation clock c in the *ECU* location is lower than or equal to the ECU cycle duration ecu_t, where $ecu\_t = \sum_{i=1}^{3} EndToEndDelay(f_{p_i})$. Similarly, the energy rate ecu_r constrains the total memory usage of eng, which needs to be lower than or equal to the sum of all memory consumptions of each f_p with respect to their execution times.

5 Resource-aware Analysis and Verification

We have presented how the ERT behavior of each f_p, described as a CCSL-extended XFG-profiled UML model is converted to a single XFG in terms of a particular `DelayConstraint`, i.e., `ExecutionTimeConstraint`. To enable tool-supported verification of the ERT behavior through well established model checkers such as UPPAAL-CORA, the XFG is transformed to an UPPAAL-CORA process (equivalent to a single PTA) using the Aβ tool. An $\mathscr{X}_{SYS}$, consisting of a set of XFGs is expressed as a parallel composition of the PTAs and is considered a network of PTA, namely $NPTA_{\mathscr{X}}$. Similarly, the XFG models of timing-, functional causality-, and resource allocation constraints, seen as observers XFG$_{Obs}$ (see Fig.2.(b) to (f), and Fig.4.(a)), are converted to PTAs, and composed to the $NPTA_{\mathscr{X}}$ in parallel, which enables us to verify all the constraints of the entire system using UPPAAL-CORA. If UPPAAL-CORA reports a counter-example this means the particular constraint does not hold for all possible system behaviors. The error traces generated by UPPAAL-CORA can be used to refine the generic constraints of the EAST-ADL/TADL2 model, or to modify the XFG-profiled UML model, or the XFG system $\mathscr{X}_{SYS}$ specification.

With the above transformed $NPTA_{\mathscr{X}} := NPTA_{\mathscr{X}} \parallel PTA_{Obs}$, we perform resource-aware analysis and safety-concerned verification through experiments. With regard to XFG$_{Obs}$, the verification becomes a reachability analysis in the two following forms: 1. *Property*1 below verifies that a system is free of any inconsistencies iff there is no deadlock and all the constraints are satisfied. In other words, it can also be used for consistency checking between the constraints specified for different parts of the ERT system; 2. $A[]\ (\neg\ P.Fail)$ verifies that a given constraint modeled with an UPPAAL-CORA process named P never reaches the *Fail* location, i.e., the constraint is satisfied iff for all initial conditions, the *Fail* location is never reached for all execution runs. The selective properties according to the TCs in Sec.2 are given and their verification results are established as valid:

P1: $A[]$ *not deadlock*

P2: $A[]\ BPS.bps \Rightarrow (\neg\ React_{Obs}.Fail \wedge Act.frba)$

P3: $BTC.Final \longrightarrow (GBC.exec \wedge (0.4{*}70 \leq c \leq 0.4{*}120))$

P4: $A[]\ GBC.(frba \vee flbarrba \vee rlba) \Rightarrow \neg\ Synch_{Obs}.Fail$

P5: $(Repeat_{Obs}.l_0 \wedge (c \leq 10)) \longrightarrow (FRWS.exec \wedge (5 \leq c \leq 15))$

P6: $A[]\ (BPS \vee BTC \vee GBC).exec \Rightarrow (\neg\ ECU.Fail \wedge (c \leq ecu\_t))$

P2, P3, and P4 correspond to TC1, TC4, and TC7 respectively. P5 specifies the combination of TC3 and TC2, i.e., the FRWS f_p is triggered every 10 ms periodically, and once the FRWS is activated, its corresponding execution event complies with the permissible execution delay [5ms, 15ms]. P6 shows that BPS, BTC, and GBC f_ps are allocated exclusively to a single ECU. The ECU resource, which can be either time or energy (memory) consumption during the allocations, is less than or equal to the ECU cycle duration *ecu_t* (or memory consumption *eng* during the cycle). The verification results tell us that the system remains schedulable.

6 Related Work

In the context of EAST-ADL, several approaches use specific annotations on a model to add the information required by scheduling analysis tools [8, 5]. Though these approaches provide temporal analysis on models, in contrast to our work, they lack precise temporal annotations at the early stage of the design, such analysis is performed at the very last step of the design. An effort on the integration of EAST-ADL and formal techniques based on timing constraints was investigated in [15, 9], which are however delimited to the executional aspects of system functions without addressing energy-aware behaviors. An earlier study [12, 14] performed towards the analysis of resource-aware EAST-ADL models based on informal semantics of the EAST-ADL architectural models, neither the resource allocation constraints nor stereotype of CCSL constraints were considered. Whereas, our current work is based on the explicit notion of the extended constraint, i.e., a combination of timing-, functional causality-, ERT behavior constraints, as well as the constraints introduced by the execution platform, which is considered for the formal analysis. Though Goknil et al. [20] presented both simulation and model checking approach of TADL2 specification, neither formal specification nor verification of different *TimeBases* and *TimeBaseRelation* in TADL2 were considered. F. Nallet et al. [18] proposed the use of MARTE to complement EAST-ADL to enable timing analysis, which, however, did not support a hybrid variable in particular regarding different energy consumption rates during execution.

7 Conclusion

We present an approach to perform (non)-functional properties verification and support schedulability analysis of automotive systems at the early design phase based on several modeling languages: 1. EAST-ADL/TADL2 for structural, timing- and f_p's causality constraints modeling; 2. XFG-profiled UML for ERT behavioral modeling; 3. the MARTE profile to enrich UML models with energy- and timing constraints; 4. CCSL to express causal- and temporal constraints between previously defined clocks. Since a CCSL is a conjunction of constraints, refinement is obtained by the addition of constraints related to different design steps, e.g., the constraints introduced by the execution platform. From the requirement formalization to specification of the implementation, an analyzable XFG model is obtained and translated into the UPPAAL-CORA model for model checking. We formally prove the validity of constraints against the requirements at any step of the refinement, and verify the correctness of the system design, as well as prove that the system is schedulable at the model level.

We discuss some open issues: 1. Although we have shown that Aβ preserves ERT behaviors and (non)-functional constraints of the BBW model and that the obtained UPPAAL-CORA manifests the same behaviors and constraints as both the SM and as the XFG models, there is no formal correctness proof for the derived procedures. As ongoing work, we use conformance checking to show that the translation procedure correctly preserves behaviors and constraints on a set of representative examples; 2. From tooling perspective, a dedicated plugin, which directly provides a translation of TADL2 to XFG, will be associated with the Aβ environment.

Acknowledgment

We wish to acknowledge Henrik Lönn, A. Goknil and M. A. Peraldi-Frati for their valuable feedback.

8 References

[1] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid Automata. In *Hybrid Systems*, LNCS(736), pp 209–229, 1993.

[2] R. Alur and D. L. Dill. A theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[3] Automotive open system architecture. www.autosar.org.

[4] G. Behrmann, A.Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata, In *Hybrid Systems*, LNCS(2034), pp 147–161, 2001.

[5] P. Cuenot, P. Frey, R. Johansson, H. Lönn, M. Reiser, D. Servat, R. Koligari, and D. Chen. Developing automotive products using EAST-ADL2 and AUTOSAR compliant architecture description language. Ingeniurs de l'Automobile 793, pp 58–64, 2008.

[6] EAST-ADL Specification v2.1.9. MAENAD Project 2011.

[7] EAST-ADL White Paper M2. http://www.maenad.eu

[8] H. Espinoza, H. Dubois, S. Gérard, J. Medina, D. Petriu, and M. Woodside. Annotating UML models with non-functional properties for quantitative analysis. In *MoDELS*, pp 79–90, LNCS(3844), 2006.

[9] E.Y. Kang, E. Enois, R. Marinescu, C. Seceleanu, P.Y. Schobbens, and P. Pettersson. A methodology for formal analysis and verification of EAST-ADL models. Reliability Engineering and System Safety Journal, pp 127–138, 2013.

[10] E.Y. Kang, G. Perrouin, and P.Y. Schobbens. Towards formal energy and time aware behaviors in EAST-ADL: An MDE approach. In *QSIC*, pp 124–127, IEEE, 2012.

[11] E.Y. Kang, G. Perrouin, and P.Y. Schobbens. EAST-ADL behavioral modeling and its translation into the analyzable formal model. TR, PReCISE Center, Belgium, 2013.

[12] E.Y. Kang, G. Perrouin, and P.Y. Schobbens. Model-based verification of energy-aware real-time automotive systems. In *ICECCS*, pp 135–144, IEEE, 2013.

[13] E.Y. Kang and P.Y. Schobbens. Advanced XFG language: Extending XFG with energy-aware timed requirement properties. TR, PReCISE Centre, Belgium, 2013.

[14] E.Y. Kang and P.Y. Schobbens. Extending EAST-ADL towards formal modeling and analysis of energy-aware real-time systems. In *ICCA*, pp 1890–1895, IEEE, 2013.

[15] E.Y. Kang, P.Y. Schobbens, and P. Pettersson. Verifying functional behaviors of automotive products in EAST-ADL2 using UPPAAL-PORT. In *SAFECOMP*, pp 243–256, LNCS(6894), 2011.

[16] MAENAD FP7 Project. http://www.maenad.eu/

[17] F. Mallet, C. Andre, and R. Simone. CCSL: Specifying clock constraints with UML/MARTE. In *ISSE*, pp 309–314, 2008.

[18] F. Mallet, M.-A. Peraldi-Frati, and C. Andre. MARTE CCSL to execute EAST-ADL timing requirements. In *ISORC*, pp 249–253, IEEE, 2009.

[19] MARTE: Modeling and analysis of real-time embedded systems v 1.1., OMG, 2011.

[20] A. Goknil, J. Suryadevara, M.A. Peraldi-Frati and F. Mallet. Analysis support for TADL2 timing constraints on EAST-ADL models, In *ECSA*, pp 89–105, LNCS(7957), 2013.

[21] M.A. Peraldi-Frati, A. Goknil, J. DeAntoni, and J. Nordlander. A timing model for specifying multi clock automotive systems: TADL2. In *ICECCS*, pp 230–239, 2012.

[22] Timing Augmented Description Language V2, 2010. http://www.timmo-2-use.org/timmo/publications.htm

[23] TIMing MOdel, http://www.timmo-2-use.org