Multiform Time in UML for Real-time Embedded Applications

Charles André, Frédéric Mallet, Marie-Agnès Peraldi-Frati

HAL Id: inria-00204503
https://hal.inria.fr/inria-00204503

Multiform Time in UML for Real-time Embedded Applications

C. André, F. Mallet, M-A. Peraldi-Frati

Aoste project (INRIA/I3S)

I3S, Université de Nice Sophia-Antipolis, CNRS; FRANCE

andre,fmallet,map@unice.fr

Abstract

Each domain has its own interpretation of time. We propose to extend UML, which is more and more used in the domain of real-time embedded applications, with a concept of time inherited from reactive system modeling : multiform time.

After a brief review of some UML profiles, we present our extensions and we illustrate—on an example from the automotive industry—how to represent and to constraint behaviors depending on multiform time. We advocate that this model of time offers wider possibilities than restricting models only to the physical time.

The original publication is available at ieee.org[1].

1. Introduction

Modeling and analysis of real-time embedded (RTE) application passes by the expression of time constraints. Even though in the common life we are used to express time using units of the international system, we are not bound to do the same for RTE applications. Actually, each domain may have its own interpretation and modeling of time. F. Schreiber [15] has described several aspects of time and defines ontologies for time in different domains of computers and their applications. Each domain defines its own language and there is a strong demand for a standard description of systems in general and time properties in particular. In this context, visual modeling languages like UML and SysML are good candidates.

RTE systems have specific requirements. Real-time systems, on the one hand, require constructs to model time-dependent events and behaviors, as well as constraints on event occurrences and execution durations. On the other hand, embedded systems are subject to additional constraints due to their limited resource availability. UML 2.0 offers constructs to represent events and behaviors, and

---

[1] http://dx.doi.org/10.1109/RTCSA.2007.51

to express constraints. However, given that the default operational semantics of the UML is inherently untimed, and rightfully so since there is no Time information in the ground metamodel, the same model can be interpreted differently depending on whether it is considered from the UML causality model or the intended timed analysis viewpoint.

Our notion of time covers both physical and logical times. Multiform time, originating from reactive system modeling, is our time model. Most of the ideas presented here have been proposed to the OMG for inclusion in the forthcoming UML profile for Modeling and Analysis of Real-Time Embedded (Marte [10]) systems. The extensions we propose to support multiform time modeling are applied to an automotive control application.

We start with a survey of some existing UML profiles. We continue by introducing the logical time and the extension proposed in Marte to take account of the multiform time. Section 4 presents the automotive application used in section 4.3 to illustrate the usage of our UML extensions. We insist on time constraints that can be used by tools to validate time properties.

2. Time modeling in UML and some of its extensions

2.1. UML

In UML [13] Time is seldom part of the behavioral modeling, which is essentially untimed (by default, events are handled in the same order as they arrive in event handlers). UML describes two kinds of behaviors: the intra-object behavior—the behavior occurring within structural entities—and the inter-object behavior, which deals with how structural entities communicate with each other [17]. The CommonBehaviors package defines the relationship between structure and behavior and the general properties of the behavior concept. A subpackage called SimpleTime adds metaclasses to represent *time* and *duration*, as well as actions to observe the passing of time. This is a very simple

time model, not taking account of problems induced by distribution or by clock imperfections. In particular the UML *causality model*, which prescribes the dynamic evaluation mechanisms, does never refer to time (stamps). Instead, the UML specification document explicitly states that *"It is assumed that applications for which such characteristics are relevant will use a more sophisticated model of time provided by an appropriate profile"*. Our contribution can be seen as providing the means for building such sophisticated time models.

2.2. SPT

The UML Profile for Schedulability, Performance, and Time (SPT) [11] aimed at filling the lacks of UML 1.4 in some key areas that are of particular concern to real-time system designers and developers. SPT introduces a *quantifiable* notion of time and resources. It annotates model elements with quantitative information related to time, information used for timeliness, performance, and schedulability analyses.

SPT only considers *metric* time, which makes implicit reference to physical time. It provides time-related concepts: concepts of instant and duration, concepts for modeling events in time and time-related stimuli. SPT also addresses modeling of timing mechanisms (clocks, timers), and timing services. But "time" here is only introduced through dedicated stereotype annotations that are *not* interpreted and given meaning as part of UML semantics. Instead, their purpose is to be understood by external analysis tools to perform schedulability or performance evaluation and after automatic translation from the UML model into such tool input format.

SPT, which relies on UML 1.4, had to be aligned with UML 2.1. This is one of the objectives of the Marte profile.

2.3. SysML

SysML (Systems Modeling Language) [12] is a general-purpose modeling language for systems engineering applications. Though SysML offers no specific support for Time, it extends UML in several ways. Our time model has taken up two of these extensions: value property with units, and constraint block. A SysML value property defines a value with units, dimensions, and probability distribution. A SysML constraint block contains equations expressing constraints between value properties. The usages of the constraints in an analysis context are represented in a parametric diagram (kind of diagram new to UML).

2.4. Non OMG profiles

Several UML profiles, which are not responses to an OMG RFP, are dealing with time. None of them supports multiform time.

EAST-EEA, an ITEA project on Embedded Electronic Architecture [14], provides a development process and automotive-specific constructs for the design of embedded electronic applications. Temporal aspects in EAST are handled by requirement entities. In theory, concepts of Triggers, Period, Events, End to End Delay, physical Unit, Timing restriction, can be applied to any behavioral elements. In practice, some of these concepts, such as the event triggering, make the timing analysis very complex. In the EAST-ADL (Architecture Description Language) document, it is recommended to use event triggering carrefully or even to avoid it.

The temporal semantics of AADL [18] concepts is defined using "hybrid automata". These automata are hierarchical finite state machines with real-valued variables that denote the time. Temporal constraints, expressed as state invariants and guards over transitions, define when the discrete transitions occur. Concurrent executions are modeled using threads managed by a scheduler. The dispatch protocol (periodic, aperiodic, sporadic and background) determines when an active thread executes its computation. AADL supports multiform time models. However, it lacks model elements to describe the application itself, independently of the resources. UML activities allow for a description of the application, actions executed sequentially or concurrently, without knowing, at first, whether actions are executed by a periodic thread or a subprogram. This important information is brought by an orthogonal process: the allocation. After several analysis iterations, the threads are eventually deployed to the execution platform.

The UML profile Omega-RT [6] focuses on analysis and verification of time and scheduling related properties. It is a refinement of the SPT profile. The profile is based on a specific concept of event making it easy to express duration constraints between occurrences of events. The concept of *observer*, which is a stereotype of state machine, is a convenient way for expressing complex time constraints. Note that the Omega Event is different from the UML Event, which poses a compliance issue.

TURTLE-P [2] is a UML profile for the formal validation of critical and distributed systems. This profile introduces temporal operators and composition (parallel, sequence, synchronization, and preemption). It deals with temporal non determinism, usual in distributed systems. Properties of a TURTLE-P model can be evaluated and/or validated thanks to the formal semantics given in RT-LOTOS.

3. Multiform time and Marte

A first form of time is the one used in physical laws, and especially in mechanics. In computer science this time is often referred to as "physical time", but its nature is above

all mathematical.

In digital systems, this ideal time is approximated by circuits, called clocks, generating well defined "periodical" signals. This leads to a discrete model of time. Unfortunately, a digital system often needs several clocks. This raises the problem of clock synchronization [8]. Distributed systems, because of their spatial extension, experience the same problem to agree on a unique time reading. To address this issue, L. Lamport [7] has introduced the concept of logical clock. With logical clocks, partial ordering of events can be obtained without recourse to any physical "real" time. Improvements in logical clocks permit to characterize the causal relationship among events [16]. For performance evaluation or hard real-time property verification, a time model restricted to partial ordering of events is not enough. Synchronization with physical time becomes necessary (see the Enhanced View of Time Specification [9] for proposed standard and service definitions).

The synchronous languages [3, 4], used in reactive system programming, also make use of logical time. In synchronous programming, (physical) time passing is represented by event occurrences; for instance a signal generated by an external device. However, these events do not have any specific status that distinguishes them from other events. Hence, a synchronous program may have statements such as *"a task must complete before 10 ms"*, and *"a car must stop within 50 m"*. Both statements express a deadline: *"10 ms"* for the former, and *"50 m"* for the latter. This is known as Multiform Time.

Marte is a response to the OMG RFP to provide a UML profile for real-time and embedded systems [10]. Marte is a successor of SPT, aligned on UML 2, and with a wider scope. Marte introduces a number of new concepts, including time concepts, which are central to this paper. The main Marte extensions of UML for time-related concepts are informally described below. This presentation focuses on the concepts (domain view) rather than the formal UML specification of the profile (UML view).

The underlying model of time is a set of time bases. A time base is an ordered set of instants. Instants from different time bases can be bound by relationships (coincidence or precedence), so that time bases are not fully independent and instants are partially ordered. This partial ordering of instants characterizes the time structure of the application. This model of time is sufficient to check the logical correctness of the application. Quantitative information can be added to this structure when quantitative analyses become necessary. Note that a specification of a temporal behavior may refer to points of time (instants) or to segments of time (durations). In the Marte metamodel of time, Instant and Duration are two distinct concepts, specialization of the abstract concept of Time.

The users of Marte have access to the time structure through clocks. Here, clocks are not physical devices; they are model elements representing a general concept of time. While in SPT, clocks were implicitly bound to the physical time, in Marte, a clock can be bound to any recurrent event. Thus, Marte distinguishes two kinds of clocks: the chronometric clocks, which make reference to physical time, and the logical clocks, which focus on the ordering of instants, possibly ignoring the physical duration between instants.

Some classes are steroNtyped by the stereotype ClockType, they define the type of a clock. Such a class specifies the nature (dense or discrete) and the kind (chronometric or logical) of the represented time, a set of clock properties (e.g., resolution, maximal value, . . .) a set of accepted time units. For the chronometric clock types, time units are the usual time units: the second (s) and its derived units. Most logical clock types use a generic time unit called tick. In some cases, they may use more specific units: a processor cycle, for instance, or even units for physical quantities, like in the illustrative example of this paper, where time is measured in angular degree (°). A Clock (i.e., an instance of a ClockType) is characterized by its unit and the values (real numbers) given to its optional properties: resolution, maximalValue, offset. resolution gives the granularity of the clock; maximalValue is the value at which the clock roll over; offset specifies the origin instant. The resolution, the maximalValue and the offset are given with the unit of the clock. A predefined Clock is provided in the TimeLibrary of Marte: idealClk. This hypothetical clock reads the dense "physical time". It is used as a reference clock for the (imperfect) chronometric clocks defined by the users of the profile. A ClockConstraint sets dependencies up amongst clocks.

In Marte time model, time-related concepts (e.g., event occurrences and behavior executions) make explicit reference to one or several clocks, through the property on, on identifies the clock and then the unit used. In UML, an Event describes a set of possible occurrences; an occurrence may potentially trigger effects in the system. A TimeEvent is an Event that defines a point in time (instant) when the event occurs. The specification can be either absolute or relative to some other instant. A TimedEvent is a TimeEvent, where the instant specification explicitly refers to a clock. Note, that in the general case it is not possible to compare two events that refer to two different clocks. The comparison is only possible when specific constraints on the clocks (ClockConstraint) or the events (InstantConstraint) are given. A UML Behavior describes a set of possible executions; an execution is the performance of an algorithm according to a set of rules. Marte associates a duration, an instant of start, an instant of termination with an execution, these times being read on a clock. A TimedProcessing is a Behavior or an Action with explicit references to clocks.

4. Application to an Automotive system

This section presents an automotive application: an ignition control and the knock correction in the case of a 4-stroke engine.

4.1. Spark-ignition engine

In a 4-stroke engine, a cycle is characterized by four phases: Intake, Compression, Combustion and Exhaust. These phases are driven by the camshaft, the positions of which are measured in angle degree (°).

Since the spark plug needs a delay to produce a flame front in the combustion chamber, the electric spark must be generated before the theoretical ignition point (ITDC: Ignition Top Dead Center) and after the Maximal Ignition Advance Angle (MIAA).

During the compression phase, at the Ignition Decision Point (IDP), an electronic control system determines the best angular position to generate an electric spark for igniting the compressed air-fuel mixture. Starting from the Basic Ignition Advance Angle (between MIAA and ITDC), which is a function of the actual engine speed and the air/fuel rate, a correction is applied to determine the Actual Ignition Advance Angle (AIAA). These corrections depend on additional parameters. The angle from ITDC to AIAA is called the *advance*.

4.2. The example of the Knock

The *knock* is a physical phenomenon generated by an auto ignition during the combustion phase and leading to a lost of efficiency of the engine, a consumption increase, and may cause irreversible damage to the cylinder.

The knock control system detects and corrects this phenomenon. It consists of one or several noise sensors, and a controller, which performs the acquisition and computes the correction. Acquisition of the noise sensor signals is carried out during an observation window (KAW-Knock Acquisition Window). The starting point (KAWS-KAW Start) and the duration (KAWD-KAW Duration) of the window may vary. They depend on the knock intensity (KI) measured during the previous combustion phase. The knock controller is an adaptive system. A digital filtering is applied to the signal samples to deduce the knock intensity. This value is then used to adjust the advance (advance correction). More details about spark ignition engine management and knock control system can be found in an automotive handbook [5].

4.3. Modeling behavior with logical clocks

The ignition engine management is a typical example of real-time application gaining from a multiform time modeling.

In a UML state machine, a label on a transition specifies a trigger that must reference an event. Labels like *"after d"* or *"at i"* implicitly defines a TimeEvent. The former specifies a relative instant, the later an absolute instant; these instants implicitly reference physical time. In Marte, this convenient notation is extended to multiform time events, by applying the TimedEvent stereotype. In a 4-stroke engine, the succession of the phases is triggered by events associated with angular positions of the camshaft, not with physical time instants. In a multiform time approach, angular positions of the camshaft are considered as (logical) instants read on a logical clock: camClk. This clock represents a discrete logical time, its unit is defined as $°CAM$ (degree cam), its resolution is 0.5 (for instance), its offset is 0, and its maximalValue is 360. All the values are implicitly given in $°CAM$, the unit of the clock.

The events that trigger the transitions in the state machine shown in Figure 1 are stereotyped by TimedEvent, and the tag value on set to camClk. For instance, we call IC (Intake closes) the event associated with the transition from Intake to Compression; it is a TimedEvent occurring 90 $°CAM$ after entering state Intake.
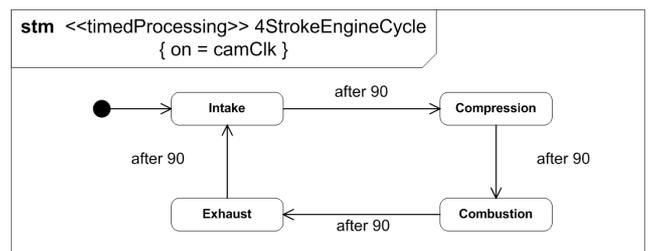

Figure 1. State machine of a 4-stroke engine cycle.

Note that, the whole state machine, which is a UML Behavior, has been stereotyped by TimedProcessing. The events involved in the state machine are, by default, bound to the same clock (i.e., camClk). This state machine is a simplified specification of the behavior. In actual engine, the Intake and the Exhaust phases are overlapping. This overlapping can be shown using UML interactions. Of course, in Marte interaction diagrams are extended to multiform time.

In Figure 1, we used the camClk. For some other aspects, it is easier to use another logical clock: crkClk (crankshaft clock), with $°CRK$ (degree crank) for unit. This clock is bound to the rotation of the crankshaft. Since a full 4-stroke cycle needs two revolutions of the crankshaft, the maximal value of this clock is 720 $°CRK$. Its resolution is assumed to be 0.5 (for instance).

The camshaft and the crankshaft are mechanically bound: the latter runs twice as fast as the former. Conse-

quently, the two clocks camClk and crkClk are tightly dependent; this dependency is expressed by a ClockConstraint (Eq. 1).

$$camClk = crkClk \text{ filteredBy } (1.0) \qquad (1)$$

This equation states that camClk is a subclock of crkClk, with instants of camClk being coincident with every second instants of crkClk. The coincident events are represented by a filtering operation according to an infinite periodic binary word 101010..., the shortcut of which is (1.0). Figure 2 presents the result of the application of Eq.1. This is a total order amongst instants of the same clock.
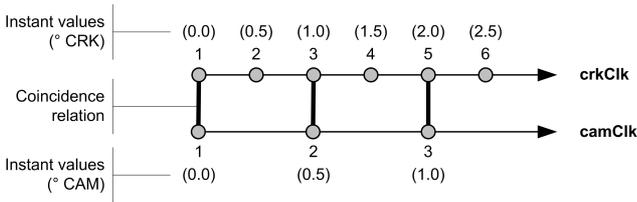

Figure 2. Clock constraint between camClk and crkClk.

On Figure 1, each phase is separated from the next one. On an actual engine, there is an overlap between phases. For example, the Compression phase starts before the actual end of the Intake phase. The start of the Compression phase is called FBDC (First Bottom Dead Center). The end of the Intake phase is IC. Using a clock constraint we can express the overlapping (Eq. 2).

$$(t_{FBDC} + 40 \le t_{IC} \le t_{FBDC} + 60) \text{ on } crkClk \quad (2)$$

In that case, the constant values, 40 and 60, are expressed according to the $°CRK$ clock unit. Using two different logical clocks provides more flexibility since it allows for using exact values found in handbooks and specifications. Of course, using several units in the same specification is possible only if there is a strong support from tools. Our profile avoids the burden of building a specific tool since graphical support and constraint editing are to be provided by UML tools that implement the Marte Profile.

4.4. Mixing logical and chronometric clocks

In real-time applications, the behavior executions are generally temporally constrained. A deadline imposed on the termination of an execution is a usual constraint that can be specified either by a duration (maximal execution time) or by an instant (occurrence of a timeout event). Here

again, Marte provides facilities to express such constraints on multiform time models.

The ignition control decomposes in three actions. First we get the basic ignition advance angle (BIAA) by a table lookup with two input parameters: the current air/fuel ratio and the current engine speed (i.e., the rotational speed of the crankshaft). The BIAA data is then used by a Corrections action that uses, among other factors, the knock intensity (KI). This action yields two results: the corrected ignition advance angle (CIAA) and information on the knock acquisition window, described below. The last action generates the actual advance (ADV). This sequence of actions is triggered at the ignition decision point (IDP). The IDP event occurs at a fixed angular position, therefore at a fixed instant on the crkClk. The new advance value must be available, under any circumstances, before another angular event (MIAA) that corresponds to the worst case (i.e., maximal engine speed and maximal allowed advance). Thus, the duration of the execution of the main ignition control activity must be less than $(MIAA - IDP) °CRK$.

The main ignition control (MICtrl) activity is just a part of the ignition control (ICtrl) activity (not represented here). The various corrections can be computed concurrently. For simplicity, we consider only the knock correction. The knock sensor is a vibration sensor that is sampled at 100 kHz (a classical frequency for vibration analysis). The use of a frequency unit makes *implicit* reference to physical time. In this application, as in many other control applications, logical clocks and chronometric clocks have to live together.

We can use another activity diagram (not shown here) to represent the behavior of the knock control (KCtrl). The behavior is triggered by the occurrence of the ITDC event (*i.e.*, when the crankshaft is at its top dead center at the end of the compression phase). This event occurs at a fixed instant ($360 °CRK$) on crkClk. The WaitWS action waits for the delay KAWS (Knock Acquisition Window Start, part of the KAW data). The knock signal acquisition action (KSA) is then carried out. Two events, KWB and KWE, denote the start and the end of this action. The acquisition fills in a buffer that is then read by the action Filtering. The knock intensity (KI) is the result of the filtering.

The acquisition terminates either when the buffer is full or when the knock acquisition window duration (KAWD, part of the KAW data) has elapsed. The first occurring event causes the effective termination. This is a *multiform time constraint*. The latter condition is measured in $°CRK$ while the former is bound to physical time through the imposed sampling period. The constraint is specified by: t_{KWE} - t_{KWB} = min (sampleNb * Tsampling on idealClk, KAWD on crkClk), where sampleNb is the maximal number of samples in the buffer, Tsampling is the period of sampling ($10\mu s$ since the sampling frequency is 100 kHz),

KAWD is the knock acquisition window duration, a value given in $°CRK$ and dynamically computed by the Corrections action of the main control system. The simplest way to make computation on these kinds of constraints is to relate them to actual chronometric clocks, for example to the idealClk defined in the Marte library. Such a constraint would depend on the engine speed. If we consider an engine with the maximal engine speed of 4500 rpm (revolutions per minute). A revolution is $360 °CRK$; The maximal engine speed is $27000 °CRK/s$, so that $1 °CRK \geq 37 \mu s$.

When considering an engine with four cylinders instead of a single cylinder, constraints become stronger. The ignition control activity (ICtrl), and hence the activity MlCtrl that is part of it, must be executed for each cylinder. However, each cylinder has its own clock: crkClk1, crkClk2, crkClk3 and crkClk4. Making clocks a first class citizen of Marte allows for using clocks as parameters of activities. The commonalities amongst these clocks are defined by the ClockType AngleClock; each cylinder clock becoming an instance of AngleClock. The offset of each clock is different and determined by the engine firing sequence (order of combustion). For an engine whose firing sequence is 1,3,4,2, the offsets of the clocks are respectively 0,180,540 and 360 for crkClk1, crkClk2, crkClk3 and crkClk4.

The dependencies are not limited to a simple sequencing of activities, we must also take into account hardware resources involved. That is why Marte has also extended composite structure diagrams with multiform time.

5. Conclusion

We believe that multiform time, introduced by reactive languages, is of first importance to specify constraints in real time embedded systems. Additionally, UML is more and more present in the industry to bridge the gap between the domain experts, the customers and the developers. This paper introduced some efforts made in the context of the forthcoming UML Profile called Marte to take account of the multiform time in UML diagrams. The goal is to use UML visual editors to capture specifications and time constraints. Using logical clocks keeps the specification as close as possible from the domain expert handbooks. Then, time analysis tools should be able to extract Marte annotations to validate some constraints. On a simplify example, borrowed from the automotive domain, we have shown that with few time constraints we can capture enough information to perform multiform-time analysis. Some validations have been performed in [1], these validations concern performance and cost requirements (processor speed, number of buffers and their size), as well as variability requirements (number of cylinders).

References

[1] C. André, F. Mallet, and M.-A. Peraldi-Frati. A multiform time approach to real-time system modeling: application to In *Symposium on Industrial Embedded Systems 2007*. IEEE, july 2007.

[2] L. Apvrille, P. Saqui-Sannes, and F. Khendek. TURTLE-P: a uml profile for the formal validation of critical and distributed systems. *Software and Systems Modeling (SoSyM)*, 5(4):449–466, December 2006.

[3] A. Benveniste and G. Berry. The synchronous approach to reactive and real-time systems. *Proceedings of the IEEE*, 79(9):1270–1282, September 1991.

[4] A. Benveniste, P. Caspi, S. A. Edwards, N. Halbwachs, P. L. Guernic, and R. de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, 2003.

[5] Bosch. *Bosch Automotive handbook, 6th Edition*. Bentley Publishers, 2004.

[6] S. Graf, I. Ober, and I. Ober. A real-time profile for UML. *STTT, Software Tools for Technology Transfer*, 8(2):113–127, April 2006.

[7] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[8] D. G. Messerschmitt. Synchronization in digital system design. *IEEE J. Select. Areas Commun.*, 8(8):1404–1419, October 1990.

[9] OMG. *Enhanced View of Time Specification*, October 2004. OMG document number: formal/04-10-04 (v1.2).

[10] OMG. *UML profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE), Request for proposals*, February 2005. OMG document number: realtime/2005-02-06.

[11] OMG. *UML Profile for Schedulability, Performance, and Time Specification*, January 2005. OMG document number: formal/05-01-02 (v1.1).

[12] OMG. *Systems Modeling Language (SysML) Specification*, April 2006. OMG document number: ad/2006-03-01.

[13] OMG. *UML 2.1 Superstructure Specification*, April 2006. OMG document number: ptc/2006-04-02.

[14] I. project. *EAST-ADL: The EAST-EEA Architecture Description Language*, June 2004. ITEA Project Version 1.02.

[15] F. A. Schreiber. *Is Time a Real Time? - An Overview of Time Ontology in Informatics*, volume F127 of *NATO ASI*, pages 283–307. Springer Verlag, 1994.

[16] R. Schwarz and F. Mattern. Detecting causal relationships in distributed computations: In search of the holy grail. *Distributed Computing*, 7(3):149–174, 1994.

[17] B. Selic. On the semantic foundations of standard uml 2.0. In *SFM-RT 2004*, volume 3185 of *LNCS*, pages 181–199. Springer-Verlag, 2004.

[18] S. Standards. *SAE Architecture Analysis and Design Language (AADL)*, June 2006. document number: AS5506/1.