

# Harmonizing MARTE, EAST-ADL2, and AUTOSAR to Improve the Modelling of Automotive Systems

Huascar Espinoza<sup>1</sup>, Sébastien Gérard<sup>1</sup>, Henrik Lönn<sup>2</sup>, Ramin Tavakoli Kolagari<sup>2</sup>

<sup>1</sup>CEA LIST/LISE, Boîte courrier 65, Gif sur Yvette, CEDEX, F-911 91 France,

<sup>2</sup>Volvo Technology Corporation, Mechatronics & Software, SE-405 08 Gothenburg, Sweden,

huascar.espinoza@cea.fr, sebastien.gerard@cea.fr,

henrik.lonn@volvo.com, ramin.tavakoli@volvo.com

## Abstract

A wide adoption of MARTE (<http://www.omgmarte.org/>) in the automotive domain requires an efficient adaptation of embedded automotive software system development concepts, culture, and practices within MARTE. This paper reports current investigations of the ADAMS project (<http://www.adams-project.org/>) on the use of MARTE in the automotive domain and on identified actions to foster convergence between MARTE and the main automotive standards, especially EAST-ADL2 (<http://www.atesst.org/>) and AUTOSAR (<http://www.autosar.org/>). Adaptation and alignment needs are communicated to the standardization bodies of MARTE as well as of EAST-ADL2 and AUTOSAR. The main results of the investigation that are discussed in this paper are 1. an alignment of the EAST-ADL2 and MARTE profiles by making EAST-ADL2 stereotypes specializations of the respective MARTE stereotypes and 2. extensions/adaptations of MARTE in order to be applicable in automotive systems development.

## 1. Introduction

In the automotive industry, software-based systems are increasingly based on the AUTOSAR standard [1]. The AUTOSAR standard represents both a modelling approach for software architecture and an execution platform.

Software architectures are the result of a systems engineering effort that takes several concerns into account. It is therefore necessary to also represent the requirements and abstract system components that are realized by the AUTOSAR software architecture. For this reason, the architecture description language

EAST-ADL2 ([2] and [3]) is defined to provide additional information required for automotive embedded systems.

An AUTOSAR model represents:

- software architecture,
- software component internals,
- hardware topology, and
- allocation and communication.

The EAST-ADL2 model complements this model by:

- requirements,
- feature content,
- variability,
- functional structure of applications, middleware, plant (environment), and
- abstract hardware architecture and preliminary functional allocation.

The Object Management Group (OMG, [www.omg.org](http://www.omg.org)) is one of the principal international organizations promoting standards supporting the use of model-based software and systems development, usually called Model-Driven Architecture. The Unified Modelling Language standard (UML [6]) is probably the most well-known of these OMG standards. UML has had significant success in the software industry as well as in other domains such as IT and financial systems. It is indeed now the most widespread language used for modelling in both industry and academia. It was designed as a general-purpose modelling language as well as a foundation for generating different domain-specific languages, mainly through its profile mechanism. This latter capability is of great importance, because it allows the general concepts of UML to be specialized for a specific domain or application.

Because of the diverse nature of the disciplines needed for designing real-time and embedded systems, it is clear that a single modelling language will not be sufficient for covering all the various concerns in-

volved in this specific area. Therefore, a new UML profile for real-time and embedded systems was issued. This profile is named MARTE (an abbreviation of “Modelling and Analysis of Real-Time and Embedded systems,” [7]). In addition to support the modelling of real-time embedded systems, MARTE was also aimed at supporting modelling and analysis of component-based architectures as well as a variety of different computational paradigms (e.g., asynchronous, synchronous, or timed). The expected advantage of designing MARTE as a UML profile is that thanks to its underlying UML foundations, its dissemination in industry will benefit on one hand from the broad acceptance of UML in both industry and academia and on the other hand from the more and more efficient tooling.

Also the EAST-ADL2 and AUTOSAR modelling approach can be interpreted as a domain model, which can be implemented as a UML profile for use in UML tools. For EAST-ADL2 a UML profile implementation is already available [3].

AUTOSAR and EAST-ADL2 are defined in an automotive context and have therefore taken into account automotive needs regarding the product structure, the development processes in use, and the manufacturer-supplier relationships at hand.

Taking EAST-ADL2 and AUTOSAR as the overall modelling pattern, MARTE-compliant UML profiles for the two modelling approaches can be introduced. This way, the modelling pattern that serves automotive needs is preserved and at the same time, analysis methods provided by the MARTE concepts are made accessible.

The MARTE profile provides constructs for representing resources, software, and communication along with a semantic foundation. Analysis tools and algorithms based on MARTE concepts can be applied on models using the MARTE profile, or specializations of it.

This paper reports current investigations of the ADAMS project (<http://www.adams-project.org/>) on the use of MARTE in the automotive domain (see [4] and [5] for another investigation of the relation between MARTE and EAST-ADL2) and on identified actions to foster convergence between MARTE and the main automotive standards EAST-ADL2 and AUTOSAR (see Section “2. Introduction to automotive systems modelling with EAST-ADL2 and AUTOSAR”). One important step to achieve the needed alignment is that the UML profile for EAST-ADL2 needs to be a specialization of the MARTE profile in order to be aligned with the MARTE constructs (see Section “3. Relation between EAST-ADL2 and MARTE”). This way, MARTE-compliant tools would be able to ma-

nipulate and analyze the automotive (EAST-ADL2) models.

The way to introduce MARTE to the automotive community is furthermore to show how UML models according to EAST-ADL2 and AUTOSAR can be defined using MARTE-compliant UML profiles, and subsequently benefit from the stringent semantics and analysis capabilities provided by MARTE. In order to be broadly applicable in the automotive domain, it is needed to adapt MARTE to specific automotive needs and challenges, an exemplary adaptation need is discussed in Section “4. Adaptation of MARTE to reflect automotive needs”.

Section “5. Conclusion” provides a concluding discussion about further research and alignment needs with respect to MARTE, EAST-ADL2, and AUTOSAR.

## 2. Introduction to automotive systems modelling with EAST-ADL2 and AUTOSAR

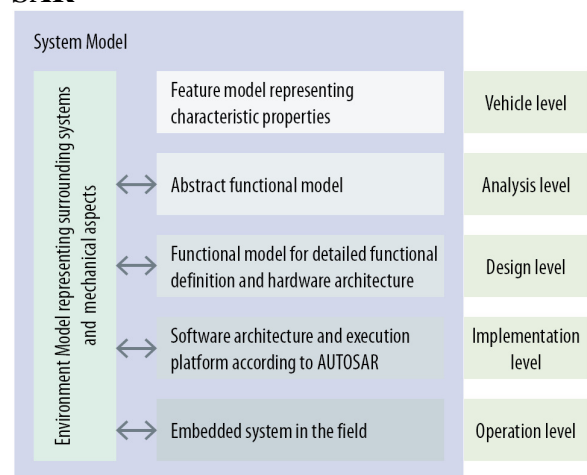


Figure 1. The model organization in EAST-ADL2

The purpose of EAST-ADL2 is to provide an information structure for the engineering information related to the development of automotive electronics and software.

The organization of the user model is based on (sub-)models on different abstraction levels (see Figure 1). On a high abstraction level, only the externally perceivable aspects of the embedded system is handled, while on a low abstraction level the implementation-specific solution in form of an AUTOSAR software architecture is represented. This ensures separation of concerns and provides means to manage the solution and problem domains in a common structure.

The abstraction levels implicitly represent different stages of an engineering process. However, the order in which a system is modelled could be top-down, middle-out, or bottom-up, in relation to the abstraction levels. Below, we will shortly describe the purpose and contents of models on each abstraction level.

## 2.1. Vehicle features: Vehicle Level

To document what the embedded system provides to the user and other external stakeholders, a feature model is used. It is a structural representation based on feature entities and may also contain requirements and use cases to complement the plain features. The feature model can be used to expose what the system provides and how a product line is organized in terms of available options and dependencies between options.

## 2.2. Abstract functional description: Analysis Level

The vehicle features are realized at the Analysis Level by abstract functions (“ADLFunction”) and devices that interact with the vehicle environment (“FunctionalDevice”). The Analysis Level captures the principal interfaces and behaviour of the embedded system without design details or decisions on implementation technology.

## 2.3. Concrete functional description: Design Level

On the Design Level, the abstract functional model on Analysis Level is refined into a detailed, solution-oriented functional model. The level of detail and concreteness is such that it can act as a detailed specification for the software architecture. It is thus a functional representation of the software architecture, disregarding the system implementation details such as software structuring, tasks and APIs.

The abstract interface elements on Analysis Level (“FunctionalDevices”) are here realized by hardware elements such as sensors or actuators, and the software parts for signal transformation (“LocalDeviceManager”). Middleware abstraction projects the platform services and functionality (OS, AUTOSAR Basic software, etc.) to a functional representation. The hardware architecture is introduced in parallel to capture the hardware entities as abstract elements (e.g., I/O, sensor, actuator, power, Electronic Control Units (ECU), electrical wiring including communication buses) to describe the topology of the electronic architecture of the systems. Design Level allows preliminary allocation of

functional entities to ECUs and provides the basis for verification either by simulation or analysis techniques such as timing and dependability analysis.

The Design Architecture contains three parts, as shown in Figure 2, representing the application software, execution platform, and hardware respectively.

The Hardware Design Architecture can be seen as a circuit diagram of the system. In addition, the transfer function of hardware devices can be specified to capture sensor and hardware characteristics as well as other behaviour of the hardware architecture.

The Middleware Abstraction contains a representation of the software platform that the applications rely on. The Functional Design Architecture contains the functionality that is subsequently realized by application software.

## 2.4. Software architecture: Implementation Level

System implementation in software is not represented by EAST-ADL2 entities, since this is the scope of AUTOSAR. However, the AUTOSAR entities are part of the system model to support traceability. Figure 2 shows the AUTOSAR model and its relation to the EAST-ADL2 functions.

## 2.5. Cross-cutting constructs

In addition to the structural entities on each abstraction level, EAST-ADL2 also supports cross-cutting aspects. For example, behavioural representations can be defined for the functional entities on analysis and design level. Safety information and error propagation can be modelled. Timing can be modelled according to concepts integrated from the TIMMO project [8] which are aligned with AUTOSAR timing constructs.

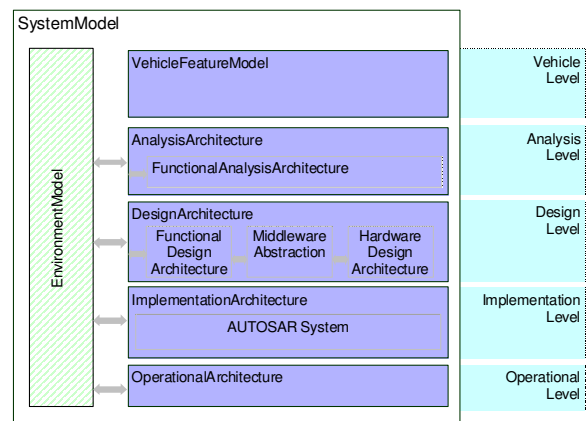


Figure 2. EAST-ADL2 System Model

### 3. Relation between EAST-ADL2 and MARTE

In this section, we discuss a comparison basis between EAST-ADL2 and MARTE. One problem occurs when comparing EAST-ADL2 and MARTE: Since both modelling approaches are constructed for specific domains and follow different design rationales, they tend to define different syntaxes (or architectural styles) for common modelling aspects. This means that some minimum alignment is highly required to provide the same kind of modelling approach for similar semantics aspects. Therefore, one objective of this paper is to identify a convergence agenda of what needs to be aligned.

#### 3.1. Modelling capabilities

The range of applications and areas of knowledge that are in the scope of MARTE (software-intensive embedded systems in general) is different than the current EAST-ADL2 scope (automotive domain). Although all of them are related with respect to the time- and resource-constrained perspective, different specialized actors, tools, and even computation paradigms are commonly involved. Consequently, it is natural that a number of modelling concepts and properties in MARTE are not useful in the automotive domain. Hence, it is important to conduct a systematic work to compare the UML profiles of both approaches to identify syntactical and semantic mismatches.

It is important to understand that EAST-ADL2 and MARTE were built in two stages. The first stage defined the conceptual constructs describing the problem that is understandable to the widest possible range of stakeholders. The product of this stage is called the domain model of the profile. The result of the next stage is the UML profile, see the next paragraph. These domain models have been created as free as possible from considerations related to specific solution technologies so as to not embody any premature decisions that may hamper later language use. Although these domain models are specified in the form of a meta-model with a textual semantic description (like AUTOSAR), it represents only conceptualization entities synthesizing the “universe of discourse”.

**UML Profiling Capabilities.** Beyond conceptual considerations, it is important to discuss some key capabilities when implementing a modelling language as a UML profile. Although MARTE was designed with pure conceptual considerations first, some UML profil-

ing capabilities were kept in mind when the domain model of MARTE was developed in the first stage.

Indeed, some of the MARTE constructs have been designed to define domain-specific viewpoints. Such viewpoints, when applied to a standard UML model, cast that model in a domain-specific way and may also add supplementary information to the model relevant to the viewpoint. For example, for an analysis based on queuing theory, a particular role in a class of a given UML model may be stereotyped as a specialized resource with specific queuing policies and service rate characteristics. Thus, the UML model is viewed through a queuing theory filter. These can then be used by an automated performance analysis tool to determine the fundamental performance properties of a software design. At the same time (and independently of the performance modeller) a reliability engineer might overlay a reliability-specific view on the same model to determine its overall reliability characteristics, and so on. This feature allows the same model to be viewed from different viewpoints (e.g., schedulability, performance, security, availability or timing).

The mechanisms that allow MARTE users to do so is the stereotype and the ability to dynamically apply and un-apply a UML profile without affecting the underlying model. Particularly in the MARTE analysis-specific sub-profiles, there is not a one-to-one mapping between analysis view-point concepts and UML concepts. The same analysis concept may be manifested in a number of different ways in a particular model. From the analysis viewpoint, all of these manifestations represent the same. Consequently, it must be possible to apply the same analysis stereotype to different base UML concepts, and conversely, different stereotypes (possibly from different analysis viewpoints) may be applied in the same model element.

#### 3.2. Comparison principles

Now we outline a set of principles to compare EAST-ADL2 and MARTE. The goal of this is to guide potential stakeholders in finding the best strategies to compare, combine or align both languages.

In general, when comparing or integrating two or more languages, we may identify four scenarios:

1. Each language is used for a different partition of the system, in which case they are practically mutually exclusive and conflicts are small or even negligible. For example, a language is used for software design and another for hardware design.
2. Each language is used for a different level of abstraction. Again, there is not much conflict

here. For instance, EAST-ADL2 for system architecture analysis and design and AUTOSAR for implementation in software.

3. The languages are used in combination in the same parts of a model (e.g., into the same modelling view) and for the same purpose or application domain. This could lead to two mapping cases: defining a composite of EAST-ADL2/MARTE concepts, and making EAST-ADL2 concepts specializations of MARTE constructs to keep them aligned.
4. The languages are used in combination but for different purposes. For example, using MARTE annotations to do performance analysis on an EAST-ADL2 model. The UML profiling capabilities to apply many stereotypes in a single model entity are crucial for this kind of usage.

Each of these scenarios could imply different comparison and integration degrees of difficulty. In order to drive our study, we propose the following factors and principles, which are orthogonal to the proposed scenarios:

- *Conceptual domain coverage.* Beyond syntactical aspects, it is important to begin by assessing both languages from a conceptual viewpoint. The intention is to reach an overall understanding of EAST-ADL2 and MARTE and determine what application domains are best covered by each. A good starting point is using the conceptual domain models underlying these two languages. The performed comparison discussed here is based on this criterion. Only conceptual entities are compared.
- *Semantic overlapping.* The evaluation of related points between both languages should be clearly identified by defining overlapping semantics (conceptual coverage). Here, the terminology needs to be carefully assessed (synonymy/homonymy). The intention is to determine which aspects of both languages can be consistently selected to compare them. Overlapping aspects must be assessed in the light of one of the language comparison principles 3. or 4. identified in Section “3.2. Comparison principles”. While principle 4. needs revisiting the notion of views and viewpoints in the context of UML profiles (see Section 3.1.1.), principle 3. requires a more careful treatment of semantic consistency.
- *Expressiveness limitations.* One fundamental requirement that should drive a useful comparison is completeness and lacking of model ex-

pressiveness. The evaluation of missing aspects needs to be objective by clearly identifying whether it implies a conceptual, semantic, or attributes insufficiency. This raises the problems of improvement and extensibility of both languages, which is an important goal of the ADAMS project.

- *Abstraction/refinement levels.* One fundamental difference between EAST-ADL2 and MARTE relates to their organization to fit different levels of abstraction. For example, while EAST-ADL2 does consider specific abstraction levels with concrete viewpoints at each level, MARTE does not preclude particular language usages in a given architecture framework. In MARTE, using language concepts on abstract or concrete levels will depend on the involved development phases, and the kind of model processing (communication, simulation, verification, etc.) required at each level. While both approaches have a merit, aligning EAST-ADL2 and MARTE requires additional work around the latter in order to target the automotive domain.
- *Advanced support for views.* A typical scenario when using multiple languages is to use the power of separation of concerns. This means that a hypothetical EAST-ADL2/MARTE tool should allow for filtering appropriate information according to specific users. This aspect is more relevant when the UML profile approach is used and more than one stereotype is applied to a single UML model element. Thus, if concepts from different UML profiles are closely related, an inter-view consistency should be assured. A possible mechanism is model constraints. A constraint applied to a model is an assertion about a model construct, the violation of which will render the model invalid to one or more stakeholders. Constraints (e.g., implemented as OCL rules) ensure adherence to a particular choice of model element constructs and enforce usage boundaries.

### 3.3. Selected comparison discussions

In this section a short overview of selected comparison discussions is provided. Please refer to [3] and [7] to find detailed information about the mentioned modelling entities from EAST-ADL2 and MARTE (see Table 1).

**Table 1. Comparison of selected EAST-ADL2 and MARTE domain model entities.**

EAST-ADL2 domain model element	UML concept	MARTE domain model element
ADLClientPort	Port	MessagePort
ADLClientServerInterface	Interface	BFeatureSpecification
ADLClientServerPort	Port	MessagePort
ADLConnectorPrototype	Connector	AssemblyConnector
ADLFlowPort	Port	FlowPort
ADLFunctionPrototype	Property	AssemblyPart
ADLFunctionType	Class	StructuredComponent
ADLInFlowPort	Port	FlowPort
ADLInOutFlowPort	Port	FlowPort
ADLOutFlowPort	Port	FlowPort
ADLPortGroup	Port	Port
ADLServerPort	Port	MessagePort
FunctionalDevice	Class	Class
LocalDeviceManager	Class	Class
Trigger	Class	Trigger
Actuator	Class	HW_I/O
ADLHwConnector	Connector	Connector
ADLHwElement	Class	HW_Resource
ADLHwElementPrototype	Property	Property
ADLHwPort	Port	HW_Port

As illustrated in Table 1, most of the concepts defined in EAST-ADL2 for functional modelling have an almost direct counterpart in either UML or MARTE. Therefore, very few extensions of MARTE are required (see Section “4. Adaptation of MARTE to reflect automotive needs” for an example discussion).

The presented comparison table depicts an excerpt of the conceptual domain coverage comparison, not a comparison of the profile elements (the stereotypes) in the first place. Since the domain model of MARTE covers a lot more concepts than are covered in the EAST-ADL2 profile, not each mapping in the provided mapping table does in fact mean a specialization between the respective EAST-ADL2 and MARTE stereotypes.

Take as an example the mapping between the EAST-ADL2 domain model entity “ADLFunctionType” and the MARTE domain model entity “StructuredComponent”. The MARTE domain model entity “StructuredComponent” is not reflected in the EAST-ADL2 profile by an own stereotype but the original UML (Structured) Class modelling element is taken instead.

But the EAST-ADL2 domain model entities “ADLFlowPort”, “ADLInFlowPort”, “ADLInOutFlowPort”, and “ADLOutFlowPort” are all stereotypes in the EAST-ADL2 profile that specialize the stereotype «FlowPort» of the MARTE profile.

So the use of the mapping table is twofold: First it provides an overview of the matching domain modelling concepts of EAST-ADL2 and MARTE and second it reflects the specialization relationship between the respective EAST-ADL2 and MARTE stereotypes.

Comparing AUTOSAR and MARTE works in the same way and also for this comparison the use of table is the same as for the comparison of EAST-ADL2 and MARTE.

By this comparison and the following alignment, future tools that perform analyses on a MARTE model can also be used for analysing EAST-ADL2 and AUTOSAR models and consequently the timing analysis will become more standardized and hence understandable for practitioners.

## 4. Adaptation of MARTE to reflect automotive needs

One important modelling concern for EAST-ADL2 is the ability to describe end-to-end flows of an application. UML proposes native concept to deal with this concern, especially through the interaction concept. However, as for other parts of the UML, user guidelines are required to drive the EAST-ADL2 practitioner to use these UML interactions to describe her/his end-to-end flows.

### 4.1. UML Interactions

UML interactions are defined in order to enable the description of how several instances (objects) of an application collaborate for performing one given activity/task. They may be shown in different kinds of diagram: sequence diagram, interaction overview diagram, communication diagram, and timing diagram. Please notice the existence of an additional possible form, a non-graphical one, where interactions may be shown as tables. The main artefacts involved within a UML interaction are lifelines and message:

- Each *lifeline* contained in one interaction denotes an individual entity that participates in realizing the interaction. In addition, a lifeline may refer to a structural element that has to be a connectable element as defined in the composite chapter of the UML specification (see section 9 in [5]). It means that lifelines refer to either parts or attributes of structured classifiers. One of the main features of connectable elements is that they may be linked by connectors that denote for the connected elements their ability to communicate together. Finally, lifelines involved in one interaction contribute to its realization by exchanging messages.
- A *message* defines different ways of communication between lifelines of one interaction, generally involving a pair of sender and receiver. In

this latter case, messages are said to be complete. In other cases, messages may be lost (one sender and no specified receiver), found (one receiver and no specified sender) or unknown. Messages may also refer to the connector that conveys them from sender to receiver. Messages may also be of following kinds: synchronous or asynchronous operation call, asynchronous signal post, creation or delete of an object, or a reply message.

UML interactions are then a good candidate for modelling EAST-ADL2 end-to-end flows. Nevertheless, as previously outlined, the main paradigm for communicating within interaction is the message that involved either operation call-based or signal-based communication. This is not sufficient for our purpose, because EAST-ADL2 enables also structural entities (also called “ADLFunction”) to communicate by data-passing, similarly to the general component model of the MARTE standard. The purpose of the next section is to describe the proposed extension to UML interactions in order to cope with data-based communications as found in EAST-ADL2.

#### 4.2. The UML sub-profile for EAST-ADL2 data-flow based interactions

The focus of this section is to describe the EAST-ADL2 sub-profile dedicated to define its extensions for modelling end-to-end flows on EAST-ADL2 functions (for both elementary and non-elementary functions).

First of all, Figure 3 denotes the stereotype «ADLE2EFlow» for marking an interaction to show that the latter is denoting an EAST-ADL2 end-to-end flow. Please notice its attribute called “kind” enabling the specification of an interaction as either internal or external. An external end-to-end flow focuses only on showing what are the inputs and the outputs of the element the interaction is attached to. Within an external end-to-end flow, one does not want to show the complete interactions. In other word, we consider the modelled element for which an interaction is described as a black box. An internal end-to-end flow will describe all the details of the flow, even what happens inside the described element.

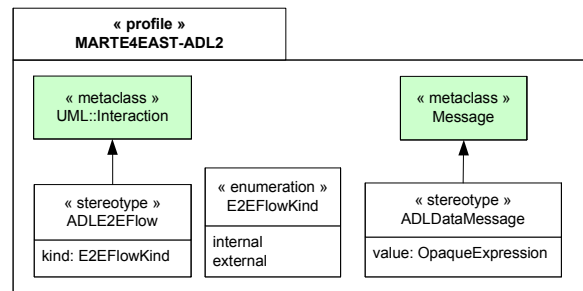
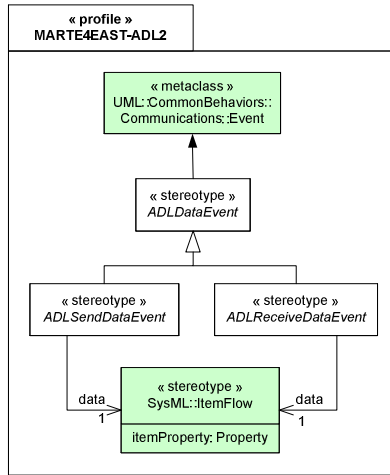


Figure 3. UML profile diagram for EAST-ADL2 interactions and data-based message.

We also need to extend the message concept as defined within the chapter Interactions of the UML specification (see Section 14 in [5]) in order to enable UML interaction to support data-based communication. As shown in Figure 3, we then define the stereotype «ADLDataMessage». This latter owns a property, value, which models the data value conveyed by the message. The type of this property is a UML OpaqueExpression. An opaque expression consists of a set of body description described in some given language defining how to interpret each text string contained in the bodies. For our purpose, it is expected that users will use VSL to describe the value conveyed by a message. But the user is also free to adopt any kind of other language, such Java or C++ for example, or even natural language.

In UML, a message owns generally two message ends: one refers to the event occurrence related to the posting of the message, whilst the other refers to the event occurrence related to the receipt of the message. Currently, due its initial intention, the UML interactions chapter defines only specific events dedicated to either operation-based message or signal-based message. For our concern, i.e., enabling data-based communication within UML interactions, we need to extend the concept of UML Event as defined in the package UML::CommonBehaviours::Communications. Figure 4 denotes the proposed extensions we define for that concern. It consists in extending the Event concept of UML by introducing an abstract stereotype, «ADLDataEvent» and to reify this latter in order to introduce both concepts of event related to posting and reception of data.



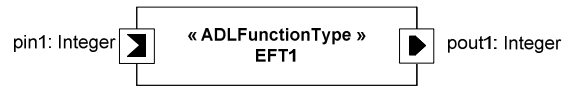
**Figure 4. UML profile diagram for EAST-ADL2 data event**

Firstly, we define an abstract stereotype «ADLDataEvent» and we generalize this latter into both stereotypes, «ADLSendDataEvent» and «ADLReceiveDataEvent». Both stereotype references a SysML ItemFlow which is used to denote the data that may be conveyed through the message.

Finally, from the notation point of view, we propose to introduce a new representation of the message concept in order to distinguish clearly data-based communication from signal-based and operation-based communications. A data-based message will then be represented as a line with a hollow triangle as an arrowhead (see examples in next section). The name of the message will consist of the text string contained in the body of data referenced by one of the end event (either «ADLSendDataEvent» or «ADLReceiveDataEvent»). In case of complete messages (i.e., messages with two message ends), it is the «ADLSendDataEvent» that is taken into account.

### 4.3. Example 1 – An elementary «ADLFunctionType»

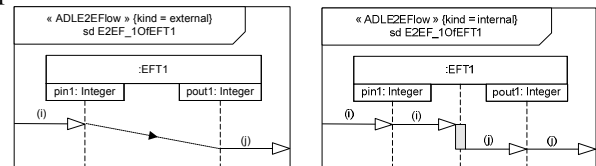
The model described in the Figure 5 denotes an elementary «ADLFunctionType», called EFT1, owning both following flow ports: on the left side of the figure, EFT1 specifies an input flow port typed as an Integer; on the right side of the figure, pout1 defines an output port also typed as Integer. Instances of EFT1 may then receive integer-typed values through pin1 and it may send integer-typed values via the port pout1.



**Figure 5. Example of an elementary «ADLFunctionType» with flow ports**

The left side of Figure 6 shows an example of an EAST-ADL2 external end-to-end flow for the previously defined elementary «ADLFunctionType», EFT1. This figure models the following: when an instance of EFT1 receives an integer value on its pin1 port, it sends some time later an output integer value (j) on its output port pout1. Please notice the specific line drawn between the head of the left message and the tail of the right message. This line denotes a general ordering (see UML specification [6] on p. 480, Section 14.3.12 for more details on this concept) that enables to specify a partial order between «OccurrenceSpecifications» (i.e., event linked to the ends of a message) that may otherwise not have a specified order. In this case, that means that the j value is sent from pout1 after the value i is received on pin1.

The right side of Figure 6 denotes an example of internal end-to-end flows. The interaction shown in this figure illustrates the fact that an instance of EFT1 will execute some actions when receiving the value i in order to generate the value j. This is described by the gray rectangle allocated to the life line shown in the middle of the sequence diagram. This rectangle specifies an execution of either a unit of behaviour, or action within the life line. Finally, let's note that in this example, there are no extra delays on both input and output ports.

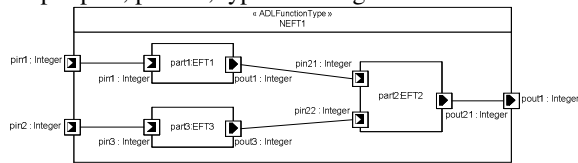


**Figure 6. Examples of end-to-end flow for an elementary «ADLFunctionType»**

### 4.4. Example 2 – An non-elementary «ADLFunctionType»

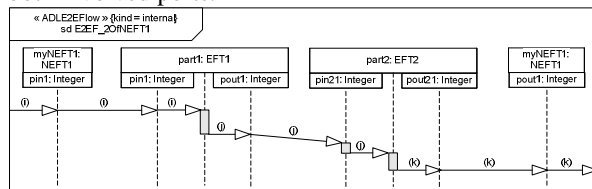
Figure 7 denotes a non-elementary function, NETF1. This latter consists of three elementary function prototypes that are denoted by parts, part1, part2 and part3. These elementary function prototypes are respectively typed by the elementary function types, EFT1, EFT2 and EFT3. EFT1 (resp. EFT3) owns an input port pin1 (resp. pin3) typed as Integer, and an output port pout1 (resp. pout3) also typed as Integer. EFT2 has two input

flow ports, pin21 and pin22, typed as Integer and one output port, pout21, typed as Integer.



**Figure 7. Example of a non-elementary function type owning two elementary function prototypes**

Figure 8 denotes an internal end-to-end flow for the previous described non-elementary functions NEFT1. On this figure, we may notice one particular point: the arrow head used to model the data-passing from pout11 to pin21 is tiled downwards showing that the data take sometime to pass along the assembly connector linking both involved ports.



**Figure 8. An internal end-to-end flow for one non-elementary ADL function**

## 5. Conclusion

This paper has discussed the use of UML and UML profiles for the modelling of automotive embedded systems. The domain-specific framework is provided by the EAST-ADL2 and AUTOSAR modelling approaches. Generic constructs for the modelling of real-time systems is provided by the MARTE standard. EAST-ADL2 provides the overall information structure and an ability to capture abstract system information for preliminary analyses and traceability to the fundamental system requirements. AUTOSAR constructs are used to represent the software architecture according to the de-facto automotive standard.

Because of the EAST-ADL2 model structure, it is possible to manage the complex dependencies between model elements, and to understand the refinement of abstract models and properties into concrete solutions.

The combination of the automotive specific UML profiles with the MARTE profile provides the benefit of preserved terminology for the automotive users while giving access to the MARTE standard for the generic aspects of real-time embedded systems.

This way, automotive users can have immediate access to theory, algorithms and tools in the MARTE context, as these evolve.

## 6. References

- [1] AUTOSAR Specification, Automotive Open System Architecture, <http://www.autosar.org/>
- [2] Cuenot, P., D. Chen, S. Gerard, H. Lönn, M.-O. Reiser, D. Servat, C.-J. Sjöstedt, R. Tavakoli Kolagari, M. Törngren, M. Weber, “Managing Complexity of Automotive Electronics Using the EAST-ADL”, 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS), July 2007. pp. 353–358.
- [3] EAST-ADL2 Specification, Architecture Description Language, <http://www.atesst.org/>
- [4] Espinoza, H., K. Richter, S. Gérard, Evaluating MARTE in an Industry-Driven Environment: TIMMO's Challenges for AUTOSAR Timing Modeling. Design Automation and Test in Europe (DATE), MARTE Workshop. Munich, Germany. March 2008.
- [5] Mallet, F., M.-A. Peraldi-Frati, and C. André, Marte CCSL to execute EAST-ADL Timing Requirements. in Proceedings of the 12th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2009), IEEE Computer Society, Tokyo, Japan, March 2009. pp. 249–253.
- [6] OMG, UML Version 2.2, formal/2009-02-02, <http://www.omg.org/spec/UML/2.2/>, February 2009.
- [7] OMG, UML Profile for Modelling and Analysis of Real-Time and Embedded Systems (MARTE) <http://www.omgmarte.org/>
- [8] TIMMO TADL specification, <http://www.timmo.org/>