MAENAD

Grant Agreement 224442

Model-based Analysis & Engineering of Novel Architectures for Dependable Electric Vehicles

Report type Deliverable D4.2.1

Report name EAST-ADL Profile for MARTE

Dissemination level	**PU**
Status	**Intermediate**
Version number	**1.0**
Date of preparation	**2011-08-31**

Authors

Editor	E-mail
David Servat	David.Servat@cea.fr

Authors	E-mail

The Consortium

Volvo Technology Corporation (S) Centro Ricerche Fiat (I)

Continental Automotive (D) Delphi/Mecel (S) 4S Group (I)

MetaCase (Fi) Pulse-AR (Fr) Systemite (SE) CEA LIST (F)

Kungliga Tekniska Högskolan (S) Technische Universität Berlin (D) University of Hull (GB)

Revision chart and history log

Version	Date	Reason
0.1	2010-12-06	Outline
1.0	2011-08-31	Intermediate

Table of contents

1 Introduction

In the previous series of projects ATESST1&2, the EAST-ADL language was implemented as a UML profile – see [1]. During these projects a study of the convergence between EAST-ADL and OMG language for modeling real time and embedded systems, MARTE, was conducted. It resulted in an annex to the MARTE specification describing how MARTE could be used to define an EAST-ADL model – see [3]. This study focused mainly on the structural description of an EAST-ADL system, in terms of hierarchical components, connectors and ports.

In the current project, MAENAD, the WT4.2 work task continues the work done to achieve a better integration of EAST-ADL and MARTE. For this it was decided to design a new version of the UML profile, which implements the EAST-ADL language. This implementation will explicitly connect the stereotypes of the EAST-ADL profile to MARTE, making EAST-ADL profile de facto a sub-profile of MARTE.

The foreseen advantages are clarity, usability of analysis tools made for MARTE models – especially those dedicated to timing analysis – further enhancement of the standardization effort for the EAST-ADL language.

The present deliverable will first explain the overall strategy adopted for conducting this re-engineering of the existing profile, then will consist in a specification of the new profile based on MARTE.

As of M12, in its first released version, the deliverable reviews a mapping of the core elements of the language. The engineering of the profile itself is under development: a release is planned for the end of 2011. Later, the profile will have to take into account the potential extensions proposed by other work tasks in the MAENAD project. New issues might appear (e.g. behavior, optimization, etc.) whose mapping to MARTE will have to be studied. It will be summarized either in intermediate releases or in the final version, planned for M36.

2 Design strategy

In this chapter the overall strategy for conducting the design of an EAST-ADL profile for MARTE will be explained. The starting point is the study done in previous projects, which resulted in the EAST-ADL annex to MARTE (see [3]) in June 2009. However several changes in the EAST-ADL language have occurred in the mean time, which makes the annex slightly outdated, although still valid in essence. The subsequent sections will provide with an update of this mapping study.

2.1 Functional modeling concepts

In this section we present an updated mapping table for the main elements of the EAST-ADL language which concern functional modeling. These are the definition of function types and prototypes, connectors and ports in various flavors. It can be noted that the overall structure of this part is shared in several modeling areas of the EAST-ADL language. For instance, one finds the same composite structuring and usage of ports and connectors, for hardware architecture description, error modeling, etc. As a result the same kind of mapping can be adapted to deal with these specific areas.

EAST-ADL concept	Description	UML concept	MARTE stereotype
FunctionType	It is an abstract concept, with concrete subtypes appearing in various levels (e.g. AnalysisFunctionType, DesignFunctionType etc.) It is the functionality provided by a car on that level.	Class	None: The stereotype FunctionType is introduced.
FunctionPrototype	It is an abstract concept, with concrete subtypes appearing in various levels (e.g. AnalysisFunctionPrototype, etc.) Appear as parts of FunctionTypes and are typed by a FunctionType. This allows for a reference to the occurrence of a FunctionType when it acts as a part.	Part	None: uses the plain UML2 part concept. A FunctionPrototype will be represented as a property typed by a FunctionType.
FunctionPort	The FunctionPort is an abstract port for data-flow or client-server interaction, which has several concrete subtypes	Port	See concrete mappings for FunctionFlowPort, FunctionClientServerport, FunctionPowerPort
FunctionFlowPort	The FunctionFlowPort represents a port that exchanges data. An EADirectionKind attribute specifies the direction of the flow (in, inout, out). The associated EADatatype specifies the type of data. FunctionFlowPorts are single buffer overwrite and non-consumable.	Port	FlowPort with direction in,inout,out. FlowPort defines also an RtFeature refering to an RtSpecification to denote the possible inter arrival time between two occurrences of the data conveyed via the port.
FunctionPowerPort	The FunctionPowerPort is a concrete port for denoting the physical interactions between environment and sensing/actuation functions, it essentially features CompositeDatatype as type in which two variables (across and through)	Port	Either a FlowPort with dedicated FlowSpecification, or specific stereotype here

	represent the physical variables exchange		
FunctionClientServerInterface	The FunctionClientServerInterface is used to specify the operations in FunctionClientServerPorts.	Interface	ClientServerSpecification
Operation	Operation features a list of EADatatypePrototype for arguments and one optional additional return parameter.	Operation	None: uses the plain UML2 operation concept
FunctionClientServerPort	FunctionClientServerPort is a port for client-server interaction. An attribute clientServerType:ClientServerKind defines the type of exchange (client or server). The port is typed by a FunctionClientServerInterface, which provides the signature of the operations available or requested by the port.	Port	ClientServerPort, with kind set to provided or required (w.r.t to server/client type)
FunctionConnector	The FunctionConnector connects a pair of FunctionFlowPorts with matching types and opposite directions or a pair of FunctionClientSeverPorts, with matching FunctionClientServerInterfaces and opposite directions	Connector	None: uses the plain UML2 Connector.
AnalysisFunctionPrototype	A concrete FunctionPrototype to model the internal structure of a composite AnalysisFunctionType at Analysis level. It is typed by an AnalysisFunctionType.	Part	None: uses the plain UML2 part concept. An AnalysisFunctionPrototype will be represented as a property typed by an AnalysisFunctionType.
AnalysisFunctionType	A concrete FunctionType at Analysis level, which can be decomposed with several AnalysisFunctionPrototypes.	Class	None: The stereotype AnalysisFunctionType is introduced.
DesignFunctionPrototype	A concrete FunctionPrototype to model the internal structure of a composite DesignFunctionType at Design level. It is typed by a DesignFunctionType.	Part	None: uses the plain UML2 part concept. A DesignFunctionPrototype will be represented as a property typed by an DesignFunctionType.
DesignFunctionType	A concrete FunctionType at Design level, which can be decomposed with several DesignFunctionPrototypes.	Class	None: The stereotype DesignFunctionType is introduced.
BasicSoftwareFunctionType	A subtype of DesignFunctionType to represent a middleware functionality at Design level.	Class	None: The stereotype BasicSoftwareFunctionTyp e is introduced.
HardwareFunctionType	A subtype of DesignFunctionType to represent the transfer function for the identified HardwareComponentType or a specification of an intended transfer function.	Class	None: The stereotype HardwareFunctionType is introduced.
LocalDeviceManager	A subtype of DesignFunctionType to represent the functional interface to sensors.actuators and other devices.	Class	None: The stereotype LocalDeviceManager is introduced.
PortGroup	The PortGroup is used to collapse several ports to one. All ports that are part of a port group are graphically represented as a single graphically	None	None

	collapsed to a single line.		

2.2 Hardware modeling concepts

In this section we review the EAST-ADL constructs for hardware modeling and depict potential mappings with MARTE concepts.

MARTE introduces a set of constructs to depict resources, be they computing, device, or communication resources at a generic level with the GenericResourceModeling package (GRM) or in a much more detailed fashion in the HardwareResrouceModeling package (HRM).

The level of description in terms of parameter defined by EAST-ADL seems to better match the generic description (see the example of a Node and its potential counterparts ComputingResource and HW_Processor). On the other hand sometimes using such generic concepts does not allow a clear distinction between elements: for instance MARTE distinguishes between two DeviceResources, HW_Sensor and HW_Actuator, but the latter are from the HRM package.

Thus there are two alternative ways to map concepts here, see table below.

EAST-ADL concept	Description	UML concept	MARTE stereotype
HardwareComponentType	It is the equivalent of a FunctionType for the Hardware level. It can be decomposed using several HardwareComponentPrototype and feature a set of connectors, ports (called pins at hardware level) and buses. Concrete subtypes are Nodes, Sensors, Actuators, PowerSupplies	Class	HW_Component
HardwareComponentPrototype	It is the equivalent of a FunctionPrototype for the Hardware level. It is typed by a HardwareComponentType.	Class	None: in fact HW_Component can owned subcomponents which are then typed by a HW_Component
HardwarePin	It is the equivalent of a FunctionPort for the Hardware level. Concrete subtypes are IOHardwarepin, CommunicationHardwarepin, PowerHardwarePin. They feature a HardwarePinDirectionKind (in, inout, out) which is the equivalent of the EADirectionKind at functional level	Port	None
CommunicationHardwarePin	The CommunicationHardwarePin represents the hardware connection point of a communication bus.	Port	None: the stereotype CommunicationHardwarePin is introduced
IOHardwarePin	The IOHardwarePin represents an electrical pin or connection point. It features an IOHardwarePinKind (analog, digital, pwm – pulse width modulated, other)	Port	None: the stereotype IOHardwarePin is introduced
PowerHardwarePin	A PowerHardwarePin is primarily intended to be a power supply. The direction attribute of the pin defines whether it is providing or consuming energy.	Port	None: the stereotype PowerHardwarePin is introduced
HardwareConnector	It is the equivalent of a FunctionConnector for the	Connector	CommunicationMedia or HW_Media

	Hardware level.		
Node	Node represents the computer nodes of the embedded electrical/electronic system. Nodes consist of processor(s) and may be connected to sensors, actuators and other ECUs via a BusConnector. Node denotes an electronic control unit that acts as a computing element executing Functions. In case a single CPU-single core ECU is represented, it is sufficient to have a single, non-hierarchical Node. They are characterized by an executionRate as float, which is the ratio compared to nominal execution (i.e. a 25% faster CPU would have an executionRate of 1.25), volatileMemory and nonVolatileMemory size in bytes	Class	HWComputingResource or HW_Processor from the HRM (HardwareResourceModeling, but both features a lot of parameters (such number of cores, etc.), which go beyond the EAST-ADL description. An alternative is to use **ComputingResource** from the GRM (GenericResourceModeling package) which only introduces a speedFactor, equivalent of executionRate
Sensor	A concrete HardwareComponentType representing a Sensor	Class	HW_Sensor or DeviceResource
Actuator	The Actuator is the element that represents electrical actuators, such as valves, motors, lamps, brake units, etc. Non electrical actuators fall outside the hardware modeling: they are part of the plant model	Class	HW_Actuator or DeviceResource
PowerSupply	PowerSupply denotes a power source that may be active (e.g., a battery) or passive (main relay). A boolean isActive indicates whether the source is active or passive.	Class	HW_PowerSupply or HW_Battery from the HRM (HardwareResourceModeling), providing a suppliedPower: NFP_Power and capacity:NFP_Ernergy Alternatively DeviceResource
LogicalBus	The LogicalBus represents logical communication channels. It serves as an allocation target for connectors, i.e. the data exchanged between functions in the FunctionalDesignArchitecture. It features a busSpeed as a float, which is in bits per second. Used to assess communication delay and schedulability on the bus. Note that scheduling details are not represented in the model. A LogicalBusKind describeds the type of bus scheduling assumed (EventTriggered, TimeTriggered, TimeandEventTriggered, other)	Class	HW_Bus or CommunicationResource
HardwarePinGroup	Equivalent of PortGroup for the Hardware level	None	None

2.3 Allocation modeling concepts

In this section we review the EAST-ADL constructs used to model allocation of functions to hardware. We briefly present MARTE constructs for this issue. It is worth noting that MARTE notion of allocation differs from the concept of deployment introduced by UML. The idea as stated in the specification is to be close to SysML approach, where allocation though relating a functional to execution platform mapping, allows that the execution platform is still in an abstract form. This essentially fits with EAST-ADL position on the matter.

EAST-ADL concept	Description	UML concept	MARTE stereotype
AllocateableElement	The AllocateableElement abstracts all elements that are allocateable. There is no way for an allocated element to have access to the other end of an allocation it is taking part in, contrary to MARTE, with its Allocated stereotype.	NamedElement	Yet MARTE defines an additional stereotype: Allocated which is applied to an allocated element, ie. An element part of an Allocate relationship (see below). It features derived property list of the sources and targets allocatedTo, allocatedFrom and an attribute AllocationEndKind (undef, application, executionPlatform, both) to distinguish the role played by the allocated element.
AllocationTarget	An abstract concept representing the potential target of an allocation. Concrete subtypes are LogicalBus and HardwareComponentPrototype, whose type can be Node, Sensor, Actuator, PowerSupply (all being concrete subtypes of HardwareComponentType)	None	In fact no restriction are made any NamedElement can serve either as source or target
FunctionAllocation	FunctionAllocation represents an allocation constraint binding an AllocateableElement (computation functions or communication connectors) on an AllocationTarget (computation or communication resource).	Abstraction	Allocate features an optional set of implied constraints as NfpConstraint, for instance to depict the cost of a particular allocation. Also an AllocationKind (structural, behavioral or hybridand AllocationNature (spatialDistribution to assign computations to execution resources or timeScheduling when describing a temporal/behavioral ordering – the order being that of targets) complement the description.

2.4 Summary

In its present form, the review of core concepts has focused on functional and hardware elements and the way to allocate one on the other. Roughly speaking this covers the needs for the models that are dealt with the Autosar gateway (see D5.2.1 [2]), which takes as input a functional and hardware description of a system and generates a potential Autosar architecture, where software components and runnables are guessed from the allocation pattern in the EAST-ADL model.

An important part that is still under investigation is the way to map the end-to-end flow requirements of EAST-ADL to MARTE concepts. It will be described in a forthcoming version of this deliverable, when the profile will be issued.

3 EAST-ADL profile for MARTE specification

In this chapter the specification of the new profile will be presented, following the recommendations of the OMG document format.

As of M12 the specification of the EAST-ADL profile conforms to 2.1.9 version – non encompassing MARTE ongoing work. It is found in a separate document – see [4].

4 References

[1] ATESST2 Deliverable D4.1.1 EAST-ADL Profile Specification, June 2010.

[2] MAENAD Deliverable D5.2.1 MAENAD Analysis workbench, June 2011

[3] OMG: UML Profile for MARTE, Version 1.0, June, 2009.

[4] EAST-ADL profile specification M2.1.9, august 2011