MAENAD

Grant Agreement 260057

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

Report type Deliverable D7.2.2

Report name Standardization plan

Dissemination level	**PU**
Status	**Intermediate**
Version number	**1.0**
Date of preparation	**2011-03-14**

Authors

Editor	E-mail
S. Torchiaro	sandra.torchiaro@crf.it

Authors	E-mail
Henrik Lönn	henrik.lonn@volvo.com
Tahir Naseer Qureshi	tnqu@md.kth.se
David Servat	david.servat@cea.fr
Carl-Johan Sjöstedt	carlj@md.kth.se
Friedhelm Stappert	Friedhelm.Stappert@continental-corporation.com
Martin Walker	martin.walker@hull.ac.uk
Fulvio Tagliabò	Fulvio.tagliabo@crf.it
Stefano Cerchio	Stefano.cerchio@crf.it
De-Jiu Chen	chen@md.kth.se

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	2010-12-07	First internal release
0.9	2011-02-28	Release for review
1.0	2011-03-14	Intermediate release

List of abbreviations

Abbreviation	Description
TADL	Timing Augmented Description Language
AADL	Architecture Analysis & Design Language
OMG	Object Management Group
AUTOSAR	AUTomotive Open System ARchitecture
FEV	Full Electrical Vehicle
CMM	Cesar Meta-Model

Table of contents

1 Introduction

This document describes the implementation of the standardization strategy of MAENAD.

In particular, the deliverable includes the description of the following selected standards/standardization activities, that are related to EAST-ADL:

- AUTOSAR
- TADL
- OMG Modeling language standards (UML, SysML, MARTE)
- AADL
- Modelica
- Automotive SPICE
- ISO 26262
- CMM – Cesar Meta-Model

Thus, the document presents how EAST-ADL relates to, supports and is adapted to existing standards. Moreover, the deliverable describes the intended (de-facto) standardization of EAST-ADL and its UML2 Profile through EAST-ADL Association and OMG Marte, respectively.

2 Related standards

In this section, standards related to EAST-ADL are presented.

2.1 AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) [1] is a standard for dealing with the ever growing complexity of automotive embedded systems. It provides a basis for modular software architecture with standardized interfaces and specifications of a run-time environment. Configuration management e.g. relocation of functionality from one computation node to another at development time is also supported by AUTOSAR standard. It is also possible to use commercial-off-the-shelf components and the components obtained from different suppliers

The result of the AUTOSAR effort is a framework and methodology [2] for standardized automotive software and hardware. A six-layered software architecture is part of the AUTOSAR framework. The layers are Application, Run Time Environment (RTE) providing a middleware functionality, Service, ECU Abstraction, Complex Drivers, and Micro-controller Abstraction [3]. While the ECU abstraction is the lowest layer, the Application is the highest one. These six layers together comprise of more than 49 software modules ranging from modules for communication, drivers for different devices to services for memory and processor. The software components are categorized into application and infrastructure. While the former is related to providing software functionalities such as control algorithm etc. the latter is used for different services related to an ECU. In addition a concept of virtual functional bus (VFB) is also introduced to resolve control related integration problems.

2.1.1 Relation to O1: Develop capabilities for modelling and analysis support, following ISO 26262

The new release of AUTOSAR integrates new features related to safety. These features include memory partitioning, support for dual microcontroller architectures for fault detection, program flow monitoring for controlling temporal and logical behavior, end-to-end communication protection [4].

2.1.2 Relation to O2: Develop capabilities for prediction of dependability & performance

AUTOSAR is related to the implementation level of EAST-ADL. The analysis carried out at the design level will be reflected directly at the implementation level.

2.1.2.1 Behavioral support

AUTOSAR provides a basis for specifying behavior of its software components. EAST-ADL and AUTOSAR share common concepts at M3 level. In terms of mode, there exist a direct mapping in terms of mode related behavior.

2.1.2.2 Dependability analysis support

Dependability analysis can be carried out in several ways. One such method is the use of hooks for fault-injection as in [5].

2.1.2.3 Timing properties

AUTOSAR supports specification of timing constraints like end-to-end delays (e.g. sensor-to-actuator or communication), minimum/maximum execution times of runnables or specification of triggering events.

2.1.3 Relation to O3: Develop capabilities for design optimization

The language cannot be used of design optimization. Instead, an optimized AUTOSAR configuration / implementation can be developed after design optimization.

2.1.4 Relation to O4: Verify, validate and explain the above capabilities in practical FEV design

AUTOSAR configuration can be used to realize the embedded system in a full electrical vehicle.

2.1.5 Modeling concepts for variability

Variability concept is under investigation.

2.1.6 Conclusions

EAST-ADL2 is aligned with AUTOSAR. Investigations will be carried out in terms of detailed mapping of concepts and tool support for realizing EAST-ADL2 with an AUTOSAR configuration.

2.2 TADL

TADL ("Timing Augmented Description Language") is a modeling language for timing properties and constraints in embedded real-time systems. It has been developed in the context of the TIMMO project [6], and is currently being further developed in the successor project TIMMO-2-USE.

The primary purpose of TADL is the enrichment of design models that are created by typical modeling languages like EAST-ADL and AUTOSAR with data that specify the required and/or existing timing behavior of the parts of the systems described. Hence, TADL enables the analysis of the given system in terms of timing and dynamics.

The design of TADL draws inspiration from three main sources: existing design frameworks such as AUTOSAR, EAST-ADL, and MARTE, existing analysis tools within the automotive domain, and typical use case scenarios from automotive applications where timing is important.

The syntax of TADL is compliant with the released AUTOSAR 3.0 meta-model. The semantics of TADL is novel and extends and clarifies the original AUTOSAR semantics. The fundamental underlying concept of TADL is the definition of events, event chains, and specific constraints on these elements, e.g. end-to-end delays or synchronization constraints. A more detailed overview of these concepts can be found e.g. in [7]

2.2.1 Relation to O1: Develop capabilities for modeling and analysis support, following ISO 26262

TADL has no explicit connection to ISO26262. Nevertheless, the standard mentions the importance of describing and analyzing timing behavior in several places, e.g. Part 6, Chapter 7 ("Software architectural design"): *"Static aspects, such as interfaces and data paths of all software*

components, as well as dynamic aspects, such as process sequences and timing behavior, need to be described."

Thus, at least to some extend the TADL also supports modeling and analysis according to ISO26262.

2.2.2 Relation to O2: Develop capabilities for prediction of dependability & performance

TADL provides the capability to describe timing on all levels of abstraction according to the EAST-ADL. The detailed description of timing properties of a system enables engineers to analyze and to some extent, predict the system's performance.

2.2.2.1 Behavioral support

TADL is not intended to model behavior in the sense of e.g. state charts or activity diagrams. However, the language provides means to describe recurrence patterns for events (periodic, sporadic, pattern, arbitrary). This could, in some sense, be seen as a description of behavior.

2.2.2.2 Dependability analysis support

There is no concept for dependability analysis support in TADL.

2.2.2.3 Timing properties

This is the sole purpose of TADL. As mentioned above, timing properties and constraints can be described for events and event chains in the system.

For events, the TADL provides means to describe the recurrence. This may be periodic, sporadic, with a certain pattern, or arbitrary.

An event chain consists of one or several stimulus events, and one or several response events. An end-to-end delay between stimulus and response can be described by means of TADL. Furthermore, synchronization constraints can be defined for the stimuli and/or the response events, meaning that these events shall all occur within a given time window.

These descriptions can be used as the basis for various timing analysis methods, such as response time analysis, schedulability analysis, etc.

2.2.3 Relation to O3: Develop capabilities for design optimization

A detailed description of the timing behavior (by means of TADL) certainly helps to e.g. identify performance bottlenecks in the system, and therefore to identify opportunities for optimization.

2.2.4 Relation to O4: Verify, validate and explain the above capabilities in practical FEV design

The TADL is not specifically designed for FEV. However, there is no reason why it should not be applicable in this area as well.

2.2.5 Modeling concepts for variability

There are no concepts for modeling variability in TADL

2.2.6 Conclusions

The TADL (Timing Augmented Description Language), as developed in the TIMMO project, is already part of the EAST-ADL. Both EAST-ADL as well as TADL are being further developed in the context of MAENAD and TIMMO-2-USE (the successor project of TIMMO). Several partners of MAENAD are involved in TIMMO-2-Use, so there is an informal information exchange between the projects, although not yet any official collaboration.

Furthermore, partners from TIMMO-2-USE and MAENAD are contributing to the AUTOSAR Timing Subgroup. That way, the compatibility of TADL with the AUTOSAR Timing Specification is ensured.

2.3 OMG Modeling language standards (UML, SysML, MARTE)

UML is a general-purpose modeling language that can be specialized or extended for dealing with specific domains or concerns. Among such domains for which extensions to UML are required to provide more precise expression of domain-specific phenomena (e.g., mutual exclusion mechanisms, concurrency, deadline specifications, and the like), there is system engineering covered by SysML, and real-time and embedded software systems, covered by MARTE.

2.3.1 System engineering

The OMG Systems Modeling Language (OMG SysML) is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. In particular, the language provides graphical representations with a semantic foundation for modeling system requirements, behavior, structure, and parametric, which is used to integrate with other engineering analysis models. SysML represents a subset of UML 2 with extensions needed to satisfy the requirements of the UML for Systems Engineering RFP.

From a structural viewpoint, SysML relies on a hierarchical description of component-based systems, in terms of “blocks”. The “block” is the basic unit of structure in SysML and can be used to represent hardware, software, facilities, personnel, or any other system element. The system structure is represented by block definition diagrams and internal block diagrams. A block definition diagram describes the system hierarchy and system/component classifications. The internal block diagram describes the internal structure of a system in terms of its parts, ports, and connectors. Essentially these diagrams are graphical specialization of UML composite structure diagrams.

SysML includes a graphical construct to represent text-based requirements and relate them to other model elements. The requirements diagram captures requirements hierarchies and requirements derivation, and the satisfy and verify relationships allow a modeler to relate a requirement to a model element that satisfies or verifies the requirements. The requirement diagram provides a bridge between the typical requirements management tools and the system models. These notions are widely used and have made SysML a standard in requirement-engineering.

The parametric diagram represents constraints on system property values such as performance, reliability, and mass properties, and serves as a means to integrate the specification and design models with engineering analysis models. This specialization has been adopted by users intending to model physical systems such as environment and/or simulation stimuli, for which mathematical expressions are commonly used to depict the modeled phenomena.

SysML also includes an allocation relationship to represent various types of allocation, including allocation of functions to components, logical to physical components, and software to hardware.

The OMG SysML™ v1.2 was published in June, 2010. The specification documents and schema files are linked from [9]. The specification document without change bars is OMG document formal/2010-06-01. The document including change bars from Version 1.1 is formal/2010-06-02. The schema files are contained in OMG document ptc/2010-03-01.

2.3.2 Real-time and embedded software systems

The OMG had already adopted a UML profile for this purpose, called the "UML Profile for Schedulability, Performance and Time" (SPT). It provided concepts for dealing with model-based schedulability analysis, focused primarily on rate monotonic analysis, and also concepts for model-based performance analysis based on queuing theory. In addition, SPT also provided a framework for representing time and time-related mechanisms. However, practical experience with SPT revealed shortcomings within the profile in terms of its expressive power and flexibility.

Furthermore, when the new significantly revised version of UML, UML2, was adopted by the OMG, it became necessary to upgrade the SPT profile. Consequently, a new Request For Proposals (RFP) was issued by the OMG seeking for a new UML profile for real-time and embedded systems. This profile was named MARTE (an abbreviated form of "Modeling and Analysis of Real-Time and Embedded systems").

The intent was to address the above issues as well as to provide alignment with other OMG standard profiles that enable the specification of not only real-time constraints but also other embedded systems characteristics, such as memory capacity and power consumption. MARTE was also required to support modeling and analysis of component-based architectures, as well as a variety of different computational paradigms (asynchronous, synchronous, and timed) This was formalized in the MARTE RFP.

In response to this request for proposal, a number of OMG member organizations collaborated on a single joint submission. After several years of intense activity, this group, called the ProMARTE consortium, arrived to get issued by the OMG the formal version of the MARTE 1.0 standard.

MARTE is structured around two main concerns, one to model the features of real-time and embedded systems and the other to annotate application models so as to support analysis of system properties. These are shown by the MARTE design model package in following figure, and the MARTE analysis model package, respectively. These two major parts share common concerns with describing time and the use of concurrent resources, which are contained in the shared package called MARTE foundations. A fourth package contains the annexes profiles defined in MARTE, as well as a predefined model libraries that may be used by modelers to denotes their real-time and embedded applications.

The profile is structured around two concerns, one to model the features of real-time and embedded systems and the other to annotate application models so as to support analysis of system properties. These are shown by the MARTE design model package in following figure, and the MARTE analysis model package, respectively. These two major parts share common concerns with describing time and the use of concurrent resources, which are contained in the shared package called MARTE foundations. A fourth package contains the annexes profiles defined in MARTE, as well as a predefined model libraries that may be used by modelers to denotes their real-time and embedded applications.

In support of the modeling of real-time and embedded systems, as for the aforementioned structure of this specification, MARTE offers the following four fundamental pillars:

1. QoS-aware Modeling
    - HLAM: for modeling high-level RT QoS, including qualitative and quantitative concerns.
    - NFP: for declaring, qualifying, and applying semantically well-formed non-functional concerns.

- Time: for defining time and manipulating its representations.
- VSL: the Value Specification Language is a textual language for specifying algebraic expressions.

2. Architecture Modeling
   - GCM: for architecture modeling based on components interacting by either messages or data.
   - Alloc: for specifying allocation of functionalities to entities realizing them.
3. Platform-based Modelling
   - GRM: for modeling of common platform resources at system-level and for specifying their usage.
   - SRM: for modeling multitask-based design
   - HRM: for modeling hardware platform
4. Model-based QoS Analysis
   - GQAM: for annotating models subject to quantitative analysis.
   - SAM: for annotating models subject of scheduling analysis.
   - PAM: for annotating models subject of performance analysis.

Note: (NFPs = Non-Functional Properties, GRM = Generic Resource Modeling, GCM = Generic Component Model, Alloc = Allocation modeling, RTEMoCC = RTE Model of Computation & Communication, SRM = Software Resource Modeling, HRM = Hardware Resource Modeling, GQAM = Generic Quantitative Analysis Modeling, SAM = Schedulability Analysis Modeling, PAM = Performance Analysis Modeling, VSL = Value Specification Language, RSM = Repetitive Structure Modelling).

The current standardization efforts at the OMG related to the UML Profile for MARTE specification are being carried on by the MARTE 1.1 Revision Task Force. This effort was started in June 2009 and will make its final report in september 2010, leading to the MARTE 1.1 version of the standard in Q3/Q4 2010, see [10].

2.3.3 Relation to O1: Develop capabilities for modeling and analysis support, following ISO 26262

No explicit support or link with ISO26262. Essentially SysML and MARTE are modeling languages which can be used to support a variety of design methodologies. The standards are not written to prescribe or even promote a particular design methodology.

2.3.4 Relation to O2: Develop capabilities for prediction of dependability & performance

In all three levels (behavior, dependability and timing) SysML and especially MARTE feature concepts suitable to support prediction and analysis.

2.3.4.1 Behavioral support

Both SysML and MARTE rely on specialization of UML behavioral concepts and diagrams: all kinds of behavioral representation can be introduced, from data-flow oriented (activities), state-based (state machines) or communication-based (sequences and interactions). Specialization of

such materials can be provided if needed to deal with special cases or interpretations of behaviors, by way of dedicated sub-profiles.

A lot of works have been done to link behavioral constructs to various model-based analysis tools.

2.3.4.2 Dependability analysis support

Modeling and analysis of non-functional properties have been a keystone in the specification of MARTE – see above short description. These concepts are generic and feature a specific language to express properties, VSL or Value Specification Language. This language enables to express various properties in mathematical terms, scientific notations, equations, which are relevant for modeling some dependability characteristics such as failure probabilities and the like. Analysis of such properties can be structured along the concepts provided for the analysis of models and possibly specialized to meet particular notations and/or analysis tools. Analysis as such, however is outside of the language and remains to be defined in particular the tools which would be fed by models.

2.3.4.3 Timing properties

This part is especially developed and documented in MARTE, as timing properties are among the most important non-functional properties one must take into account when modeling real-time systems.

The [11] describes how an EAST-ADL model can be analyzed w.r.t. schedulability analysis, by the addition of some MARTE based constructs.

2.3.5 Relation to O3: Develop capabilities for design optimization

Design optimization is not explicitly included in any of the languages, however works have been done to make use of concepts to support architecture exploration, see for instance [12].

2.3.6 Relation to O4: Verify, validate and explain the above capabilities in practical FEV design

FEV domain being quite new, we do not know if works have been done in this field to assess the suitability of MARTE and SysML. However these languages being rather generic, and the means to derive specialization of them being included in the standards, we do not see why there should be any gaps remaining.

2.3.7 Modeling concepts for variability

As such variability concepts are not present in either MARTE or SysML. However some works have been done to see how the analysis concepts of MARTE could support a software product line engineering, see for instance [13].

2.3.8 Conclusions

EAST-ADL is constructed as a domain specific language, but it could be implemented as a UML2 profile, i.e. a specialization of the UML2 language for a specific application domain, which is proven by the Papyrus implementation available. This can be viewed as an implementation of EAST-ADL in UML2, thus allowing for the use of various available UML2 modeling tools. SysML concepts are reused to manage requirements and plant modeling constructs.

Further harmonization was carried on by releasing the EAST-ADL profile as an annex to the MARTE – done in ATESST2 and ADAMS project, see for instance [14].

2.4 AADL (Architecture Analysis & Design Language)

AADL is a modeling language used to describe both the hardware and software of large scale embedded and/or real-time systems. Although it originated in the aerospace industry, it has since been applied in a number of different domains, including the automotive, spacecraft, autonomous systems, and medical industries. It has been standardized by SAE, the Society of Automotive Engineers, as SAE AS 5506 in November 2004 (and later updated to Version 2 in Jan 2009).

The AADL language is focused on the modelling of task and communication architectures, but has many capabilities. It uses a hierarchical, component-based architecture to model both the structural and functional architecture of the system, and can map the software components to an execution platform. It also describes the functional interfaces of the components, e.g. data inputs and outputs, and includes features for modeling performance- and safety-critical aspects of components, e.g. timing and fault tolerance. The latter is part of an entire error annex for modeling dependability and related information. AADL also features a standardized XML interchange format to facilitate model exchange and tool support.

AADL tool support comes in two main varieties: the Open Source AADL Tool Environment (OSTATE), which is based on the Eclipse platform, and different forms of commercial tool support, including extensions to UML tools (e.g. Artisan, Rational Rose), a proprietary tool suite (STOOD), and links to different in-house analysis tools via XML interchange (e.g. at Airbus, Rockwell etc). UML profiles for AADL exist, including a UML profile under development in MARTE.

2.4.1 Relation to O1: Develop capabilities for modeling and analysis support, following ISO 26262

AADL does not presently have any known couplings or direct support for ISO 26262. However, it does have support for error modeling and dependability analysis, both of which are necessary to support ISO 26262. It is likely that at least some of the necessary steps to support ISO 26262 are possible when using AADL models, e.g. simple FMEA & FTA, but it lacks capabilities for other steps, e.g. ASIL allocation.

2.4.2 Relation to O2: Develop capabilities for prediction of dependability & performance

AADL has extensive capabilities for analysis and prediction of both dependability and performance characteristics.

2.4.2.1 Behavioral support

In addition to AADL's existing data flow and state modeling features, a new behavior annex (another extension to AADL analogous to the error annex) is currently undergoing the standardization process. In particular, it includes new features designed to support concurrency behavior and validation of implementation, and expands AADL to allow better modeling of subprograms, message exchange, and thread behaviors etc. More information can be found in the AADL Behavior Annex itself and presentations upon it, e.g.[15][16].

2.4.2.2 Dependability analysis support

AADL supports dependability modeling and analysis primarily via its error annex, an extension to the core SAE AADL. A dependability model consists of both the nominal architecture model and

one or more error models. Error modeling in AADL is based on textual representations of state machines, which can be used to construct a compositional/hierarchical model of the system failure behavior. The error models describe the behavior of components in the presence of internal fault or repair events and external propagations from the components' environment.

An AADL error model consists of a model type and at least one error model implementation. It takes the form of a state machine that can be associated with another AADL element (e.g. component or connection) in order to describe the failure behavior of that element in terms of logical error states. Error models can be associated with hardware, software, and composite component types, as well as the connections between them. Error models can also be constructed hierarchically using composition to represent failure behavior of entire systems or subsystems. This allows AADL to use the same modellng elements to capture hazards at system level, risk mitigation schemes at subsystem level, and low-level failure modes and effects at component level. Error models are also reusable and can be processed to automatically regenerate safety and reliability models when making changes to the nominal architecture model.

Further information can be found in [17][18].

2.4.2.3 Timing properties

AADL contains native support for modeling real-time systems, including such elements as threads, processes, and execution components, and concepts like synchronization and scheduling. Normal AADL modeling tools like STOOD and OSTATE allow these elements to be used in a model, and performance analysis is possible with AADL compatible tools like Fremont and Cheddar. See also [19][20].

2.4.3 Relation to O3: Develop capabilities for design optimization

Although AADL is not specifically designed with automatic optimization in mind, there has been some work in this area. One example is the ArcheOpterix AADL optimization tool, a plugin for the Eclipse-based OSTATE AADL modeling package, which uses evolutionary algorithms to perform Pareto optimization of AADL models. The optimization is parametric-based and evaluates and validates different variations of quality attributes [21].

2.4.4 Relation to O4: Verify, validate and explain the above capabilities in practical FEV design

There does not appear to be much information on whether (or how) AADL has been applied in the design of FEVs. However, AADL may already contain many of the capabilities necessary to model FEVs and its extensible nature means that new additions or a new annex specifically tailored to FEV design may be produced at some point.

2.4.5 Modeling concepts for variability

Although AADL itself does not contain modeling capabilities to support variability, extensions to AADL have been proposed that add these features and accommodate product line engineering [22][23].

2.4.6 Conclusions

As another architecture description language, AADL is some respects a competitor of EAST-ADL. Although it originated in the aerospace domain, it can also be applied to automotive systems, and its capabilities for modeling embedded and real-time systems makes it well suited to EE scenarios.

Compared to EAST-ADL, it has a narrower focus - it focuses primarily on software and hardware modeling, and although an AADL model can be developed over time from a more abstract collection of functional components to a more concrete implementation design, it does not natively support the multiple layers found in EAST-ADL. In particular, it lacks core support for abstract feature modeling and more detailed implementation modeling.

However, AADL has a very extensible nature and this is reflected by the development of various annexes to provide better support for other analysis and design tasks, such as behavior modeling and dependability analysis. The error annex provides a well-established way of modeling failure behavior and safety requirements; safety requirements can be associated with each component and state machine-based error models can be developed to model component failures. This information can also be stored in libraries and reused, and behaviors can be inherited from as part of the model hierarchy. Safety analyses can then be carried out to obtain further information about the system and verify that safety requirements are met.

AADL is increasingly used in industry and this is at least partly due to its extensive tool support. Both commercial and open source modeling tools are available, in addition to extensions to existing UML modeling tools and other proprietary packages; both AADL tools have been extended with additional capabilities for modeling and analysis, including e.g. optimization and timing analysis capabilities.

It would be potentially useful to investigate AADL further to see which features it possesses that EAST-ADL lacks and that could be easily integrated into further developments of EAST-ADL. Extensive harmonization with EAST-ADL is unlikely to be possible, however, as both languages were developed with different goals in mind and are already becoming increasingly well-established in their own right. In particular, EAST-ADL has a broad scope that attempts to more fully model the entire design process, from requirements capture and feature modeling through more detailed architecture and functional modeling and on to detailed implementation modeling; the scope of AADL is narrower, focusing more on functional and architecture modeling, but its growing array of annexes and tools means that it is further developed and potentially more capable in this area.

2.5 Modelica

Modelica is a language for object-oriented modeling dynamical systems, preferably physical systems. There are many tools supporting Modelica, e.g.:

- Dymola – The oldest and in many ways the most complete Modelica tool, often associated with the Modelica language.
- OpenModelica – An open source-tool for Modelica, developed in Linköping University.
- jModelica – An open source-tool for Modelica, developed in Lund.
- SimulationX – A tool similar to MATLAB/Simulink, but with Modelica support. Also includes HiPHOPS-support, which could be of interest for EAST-ADL

2.5.1 Relation to O1: Develop capabilities for modeling and analysis support, following ISO 26262

Not applicable.

2.5.2 Relation to O2: Develop capabilities for prediction of dependability & performance

Modelica, and related tools can be used to model the behavior of the physical system, and the embedded system.

2.5.2.1 Behavioral support

Modelica supports equation-based behavior modeling, which enable “physical modeling”, which is attractive to use when modeling physical systems. Modelica also support block diagram modeling, like Simulink. Modelica also support discrete-event, and discrete-time synchronous modeling, allowing modeling and simulation of hybrid systems. However, until recently, there has been no graphical support to model such systems (e.g. State Machines). One approach is the ModelicaML, which simulates UML state-machines using the Modelica discrete-event “backend”. Another approach is the Modelica StateGraph 2 library, which is a state-machine library for Modelica.

2.5.2.2 Dependability analysis support

Using the Rodelica language, which is an extension to (or a derivate of) Modelica, failure modes can be defined for each component, which enables model based diagnosis. This language should be useful for fault-injection, design optimization and more.

Rodelica is an object oriented, equation-based language for diagnosis purposes which is very close to Modelica. It is designed for several domains including mechanics, electrical and hydraulics. Rodelica supports high level modelling by composition as well as detailed library component modelling by equations. It allows failure modes to be defined as alternative behaviors controlled by a parameter defining current failure mode.

Based on Rodelica, optimization, FMEA, FTA, diagnostic desicion trees, etc. can be performed. The Rodon tool from Sörman Information is a commercial solution for this purpose.

2.5.2.3 Timing properties

To be investigated

2.5.3 Relation to O3: Develop capabilities for design optimization

Using the Optimica extension (see jmodelica.org), Modelica can be used for dynamic optimization.

2.5.4 Relation to O4: Verify, validate and explain the above capabilities in practical FEV design

A FEV could be modeled using Modelica. A prototype plugin to show that exchange between Modelica models and EAST-ADL is feasible, could possibly be developed. Another opportunity is to use the ModelicaML profile, which is implemented in Papyrus UML, and will be integrated in Papyrus MDT in 2011.

2.5.5 Modeling concepts for variability

To be investigated.

2.5.6 Conclusions

Modelica is focused on simulation, especially of continuous-time systems. Modelica could be used for simulation of models at different levels of abstraction. One opportunity is that ModelicaML will be integrated in Papyrus MDT. The OpenModelica platform is developed in the OpenProd project, and would allow an open-source simulation program to be included in the MAENAD analysis platform.

A Modelica-related issue is the use of the Functional Mock-up Interface, and Functional Mockup Units. VTEC has worked with translation of FMI and FMU to EAST-ADL XML.

The Modelica language will be further investigated by Conti and KTH in a case-study.

2.6 Automotive SPICE

SPICE, Software Process Improvement and Capability Determination, is a framework for the assessment of processes. It is an international standard **ISO/IEC 15504.** Automotive SPICE [24] is an instantiation tailored for automotive use. It allows automotive purchasers to assess suppliers operations objectively.

Automotive SPICE has a *process assessment model* with a set of assessment indicators of process performance and process capability. The indicators are used as a basis for collecting the objective evidence that enables an assessor to assign ratings. Concretely, different processes can be represented, and for each of them different levels of Capability can be assessed. The more mature process, the better the score. For processes that have capability level 1 and above, it is also possible to assess operational performance.

Automotive SPICE also has a Process Reference Model. A set of predifined processes are nominated, categorized in to process categories (Primary Life Cycle Processes, Organizational Life Cycle Processes and Supporting Life Cycle Processes) and process groups (e.g. Acquisition process group, Supply process group and Engineering process group for the first category).

2.6.1 Assesment Levels

Each process is assessed from level 0 to 5, see below:

- Level 0: Incomplete process

  The process is not implemented, or fails to achieve its process purpose. At this level, there is little or no evidence of any systematic achievement of the

  process purpose.

- Level 1: Performed process

  The implemented process achieves its process purpose.

- Level 2: Managed process

  The previously described Performed process is now implemented in a managed fashion (planned, monitored and adjusted) and its work products are appropriately established, controlled and maintained.

- Level 3: Established process

  The previously described Managed process is now implemented using a defined process that is capable of achieving its process outcomes

- Level 4: Predictable process

  The previously described Established process now operates within defined limits to achieve its process outcomes.

- Level 5: Optimizing process

  The previously described Predictable process is continuously improved to meet relevant current and projected business goals.

For level 1, the process performance attribute is assessed from grade 1 to 4 regarding two indicators denoted base practice and work product. Capability Level 2 to 5 and above have two process attributes. These are assessed from grade 1 to 5 regarding two indicators denoted Generic Practice and Generic Resource.

2.6.2 Process groups relevant for EAST-ADL

The assessment of Automotive SPICE capabilities depends on certain practice in the company. Some of these practices are relevant for EAST-ADL, and the use of systems modelling will improve the assessment.

The acquisition process group has one process with EAST-ADL relevance:

- ACQ.11 Technical requirements

The Engineering process group contains the following processes, all of which benefit from rigorous modelling:

- ENG.1 Requirements elicitation
- ENG.2 System requirements analysis
- ENG.3 System architectural design
- ENG.4 Software requirements analysis
- ENG.5 Software design
- ENG.6 Software construction
- ENG.7 Software integration test
- ENG.8 Software testing
- ENG.9 System integration test
- ENG.10 System testing

Supporting Life Cycle Processes have a subset of processes with EAST-ADL relevance:

- SUP.1 Quality assurance
- SUP.2 Verification
- SUP.4 Joint review
- SUP.7 Documentation
- SUP.8 Configuration management

2.6.3 Conclusion

System modeling has important role in supporting process implementation according to Automotive SPICE. To further show how EAST-ADL can increase the process capability level, a more detailed account for how each base practice and work product and each generic practice and generic resource relates to EAST-ADL is required.

2.7 ISO 26262

The automotive industry shares the view that in the next 10 years 90% of its expected innovations will be based on Electrical Electronic (E/E) systems with a huge emphasis on the Safety Systems. This also because the European Commission has the target to reduce by 50 % (vs. 2001) the dead rate due to road accidents before 2010 and by 75% before 2020. This important goal will be supported by new passive, preventive and active safety systems to decrease the probability that an accident occurs and to mitigate the consequences on vehicle occupants and other road users. New functionalities for active safety are starting to be available on the market to assist the driver in the task of controlling the vehicle to guarantee the Maximum Vehicle Stability and the Automatic Recovery in Emergency Manoeuvres. Driving assistance functions are based on E/E systems and are built by integrating electronic units (and even software components) from different suppliers. These functionalities are potentially safety-critical, because in case of malfunctioning, they could have an impact on the system behavior and, consequently, on the vehicle controllability. These hazardous situations could cause severe injury to the involved people.

The increasing number of safety critical systems installed in vehicles, coupling among different sub-systems and the increased complexity of the architecture make it necessary to define an appropriate methodology for addressing all aspects concerning the effects of potential faults. In other words it is necessary to define a set of methods to allow the application and management of the functional safety.

The methodology has to be unified, internationally recognized and peculiar for the automotive field. The solution adopted by the international community has been to develop and to apply the new standard ISO/FDIS 26262 “Road vehicles – Functional safety” [25]. ISO/FDIS 26262 represents the state of the art regarding the safety processes with the related methods and the safety requirements for the development, production, maintenance and decommissioning of E/E systems installed in series production passenger cars (currently with a max gross weight up to 3,5 t). This standard has a wide implication on the information exchange among OEMs and suppliers in the automotive domain. As most development today is distributed among several companies and departments, it is important that all information exchange is precise enough to enable the OEM to take full responsibility of the entire functional safety process.

The ISO/FDIS 26262 requires to apply the “functional safety approach”, starting from the preliminary vehicle development phases and continuing along the whole product life-cycle. This approach will allow to design a safe automotive system. Furthermore it provides an automotive specific risk-based approach for determining risk classes named ASILs (Automotive Safety Integrity Levels). The new standard uses the ASILs for specifying the item's necessary safety requirements for achieving an acceptable residual risk, and provides requirements for validation and confirmation measures to ensure a sufficient and acceptable level of safety being achieved.

ISO 26262 consists of the following parts:

- Part 1- Vocabulary: specifies the terms, definitions and abbreviated terms for application in all parts of ISO 26262.
- Part 2- Management of functional safety: specifies the requirements on functional safety management for automotive applications.
- Part 3- Concept phase: specifies the risk assessment procedure and the requirements to be applied during the concept phase to define a safe E/E architecture archetype.

- Part 4- Product development- system level: specifies the requirements to be applied during the product development at the system level.
- Part 5- Product development- hardware level: specifies the requirements to be applied during the product development at the hardware level.
- Part 6- Product development- software level: specifies the requirements to be applied during the product development at the software level.
- Part 7- Production and operation: specifies the requirements on production, operation, service and decommissioning.
- Part 8- Supporting processes: specifies the requirements for supporting processes, like qualification of software tools, qualification of hardware and software components, and proven in use argument.
- Part 9- ASIL-oriented and safety-oriented analyses: specifies the requirements to be applied to perform an ASIL-oriented, and the ASIL decomposition approach
- Part 10- Guideline on ISO 26262: is an informative part dedicated only to give an overview on ISO 26262, intended to improve the understanding of the other Parts of ISO 26262.

2.7.1 Relation to O1: Develop capabilities for modelling and analysis support, following ISO 26262

The modeling concepts developed in MAENAD should cover the needs defined in ISO 26262 standard as, for example, the requirements to use a semi-formal notation for specifying the safety requirements (ISO 26262, part 8, clause 6.4.1.1), to make them unambiguous, comprehensible, atomic, internally consistent. Moreover, MAENAD approach should help in defining an architecture description language for model-based development of automotive embedded systems associated to a “safety analysis based”.

2.7.2 Conclusion

EAST-ADL provides language-level support for the concepts defined in ISO 26262, including vehicle-level hazard analysis and risk assessment, the definition of safety goals and safety requirements. One of the aim of MAENAD is to enrich EAST-ADL with new concepts for overall safety assessment, like safety mechanism to avoid “latent faults”, “systematic faults” and “random hardware faults”, to unsure the covering of the whole safety design process.

2.8 CMM – CESAR Meta Model

In the CESAR project, the so called CESAR meta model (CMM) is being developed. It is a unified approach for integrating different modeling languages. The CMM effort is highly relevant for EAST-ADL and MAENAD for the following reasons:

- CMM is to a large extent based on EAST-ADL but has also been influenced from other sources (Thales Arcadia MM, and the HRC MM from Offis). The current CMM is thus not fully compatible with the EAST-ADL.

- The CESAR project has a large momentum due to its size and follow-up projects including MBAT, it is thus likely to have an impact. CESAR is currently developing a standardization strategy and one of the candidates mentioned for standardization is the CMM.

2.8.1 Relation to O1: Develop capabilities for modelling and analysis support, following ISO 26262

CMM is not directly related to ISO 26262.

2.8.2 Relation to O2: Develop capabilities for prediction of dependability & performance

2.8.2.1 Behavioral support

Behaviors in CMM for component/system, component contract, and errors are based on hybird-automata (i.e., HRC). EAST-ADL on the other hand focuses on the "behavior containers" for the integration and handling of black-box component behaviors (in terms of functionTrigger and functionBehavior). Such black-box behavior containers (as well as requirements and use cases) can however be annotated with additional parametric, state-machine, computation constraints through the EAST-ADL Behavior Annex. In MAENAD, KTH are working on an integration of EAST-ADL Behavior Annex and HRC.

2.8.2.2 Dependability analysis support

The CMM only supports component/system error definition based on generic concepts, not directly related to dependability. The definition itself is based on the same behavior model as for normal behavior (HRCBlock). This "rich" behavior modeling support complements EAST-ADL well in regards to the error behavior/logic (e.g., in EAST-ADL, the expression of error behavior/logic uses currently a proprietary format: HiPHOPS).

There is a conceptual difference on the semantics of errors: In EAST-ADL, dependability related information is treated as a separate nonfunctional constraint (e.g., faults/errors/failures are analytical annotations for safety analysis and safety goals&requirements); In HRC/CMM, error definition (i.e., FailureCondition) is treated as an integrated part of component contract/behavior.

One issue to be resolved from EAST-ADL side is that the support for test modeling in the context of fault injection, where actual faults and malfunctions need to be executed and not only modeled in an analysis tool. This implies the integration of analytical assumptions (e..g, faults) with the component/architecture definition so that faults (e.g., a single-point of failure identified through FTA analysis based on EAST error model) can propagate through the nominal architectures. In this regard, it seems that the HRC/CMM provides a good emphasis on the actual component behavior.

2.8.2.3 Timing properties

Timing support in TADL/EAST-ADL represents a budget-oriented approach to timing constraints, while HRC/CMM supports the definition of temporal behaviors through its hybrid-automata semantics (e.g., the Derivate operator in the Expression). Alternative approaches like UPPAAL can extend HRC/, CMM with explicit timing expression.

Events in EAST-ADL exist only in the timing package for denoting occurrences (e.g. function triggering, data receiving) that are of particular concern for the assignment timing constraints. There is no notion of event in CMM.

2.8.3 Relation to O3: Develop capabilities for design optimization

To be investigated.

2.8.4 Relation to O4: Verify, validate and explain the above capabilities in practical FEV design

To be investigated.

2.8.5 Modeling concepts for variability

CESAR uses the CVL variability model, compared with CVM that EAST-ADL uses. This has implications that need to be sorted out.

2.8.6 Conclusions

For MAENAD it is important to make sure that the CMM and EAST-ADL are not diverging, preferably they should be converging, or at least there should be a well defined mapping between them.

In order to accomplish this the MAENAD consortium intends to

- Maintain a close dialogue with the CESAR projects, using the partners which are partners of both (including Volvo)
- Try to influence the direction of the CMM evolution. CMM is still in a draft stage, which means that there should be possibilities to influence it.
- Volvo and KTH are already part of the CESAR CMM team.

3 EAST-ADL Association

The EAST-ADL language has been developed by different project and stakeholders over time. To synchronize further refinement of the language and provide a single information portal for EAST-ADL, the EAST-ADL association is about to be formed.

The EAST-ADL Association is a non-profit, non-governmental organization with the aim of assisting and promoting the development and application of the EAST-ADL.

The EAST-ADL Association will stipulate the content of new versions of the EAST-ADL language. This will be done through collaboration between the members of the association and within projects and organizations working with EAST-ADL.

The EAST-ADL association claims no rights to the EAST-ADL, as this currently lies with the consortia developing EAST-ADL V1, V2 and V2.1.

Membership in EAST-ADL Association is open to individuals and organizations who agree to support the purpose of the association. Members can be affiliated with consortia, companies, institutes, universities or other organizations. The initial set of members are companies and individuals from the ATESST2 consortium.

The EAST-ADL Association has no fees or funds, and each member carry any costs for contributing.

An agreement for the EAST-ADL association is currently being worked out taking legal aspects and practical aspects of the intended collaboration into account. The launch date is depending on the time needed to define the membership agreement.

4 Standardization Strategy

To summarize this document, the standardization strategy based on the existing standards has been discussed.

The document lists several formal and de-facto standards that are related to EAST-ADL. They fall into three categories:

- Standards that are supported EAST-ADL

Modelica, Rodelica and AUTOSAR representations can be integrated in an EAST-ADL model, for example by referencing a Modelica model as plant model behavior.

- Standards where EAST-ADL shows compliance or contributes to meeting the standard

ISO26262, SPICE, Automotive SPICE, etc. are standards that benefit from using EAST-ADL.

- Standardization of EAST-ADL

EAST-ADL Association and OMG Marte profile represent de facto Standardization of the EAST-ADL language and profile itself.

There is no plan for formal standardization of EAST-ADL in the context of ISO, SAE, or the like. This is because the overhead and formality that goes with such standardization was not considered appropriate. Instead there are three channels foreseen for the de-facto standardization of EAST-ADL:

- The EAST-ADL association for de-facto standardization of the language.
- The EAST-ADL profile for Marte for standardization of the UML Profile of EAST-ADL.
- AUTOSAR Consortium for standardization of a subset of EAST-ADL concepts by influencing the AUTOSAR consortium to adopt concepts from the project.

5 References

[1] AUTOSAR Website. http://www.autosar.org/ .

[2] The AUTOSAR Consortium, \AUTOSAR Methodology," Tech. Rep. V1.2.1, R3.0, Rev 001, http://www.autosar.org/download/AUTOSAR_Methodology.pdf, 2008.

[3] The AUTOSAR Consortium, \Layred Software Architecture," Tech. Rep. V2.2.1, R3.0, Rev 001, 2008.

[4] Fürst et al. AUTOSAR – A Worldwide Standard is on the Road.

[5] Patrick E. Lanigan, Priya Narasimhan, Thomas E. Fuhrman, Experiences with a CANoe-based Fault Injection Framework for AUTOSAR, IEEEIIFIP International Conference on Dependable Systems & Networks (DSN), 2010.

[6] TIMMO Website. http://www.timmo.org/

[7] F. Stappert, J. Jonsson, J. Mottok, R. Johansson. “A Design Framework for End-to-End Timing Constrained Automotive Applications”. In *Embedded Real-Time Software and Systems (ERTS),* Toulouse, France. May 2010.

[8] SysML specifications, tutorials and documentation, http://www.omg.org/spec/SysML/1.2/

[9] MARTE wiki, specifications, tutorials and documentation, http://www.omgwiki.org/marte/

[10] Completing EAST-ADL2 with MARTE for enabling scheduling analysis for automotive applications Saoussen Anssi, Sara Tucci-Pergiovanni, Chokri Mraidha, Arnaud Albinet, Francois Terrier, Sébastien Gérard, ERTS’2010, http://www.erts2010.org/Site/0ANDGY78/Fichier/PAPIERS ERTS 2010 2/ERTS2010_0123_final.pdf

[11] Leveraging analysis-aided design decision knowledge in UML-based development of embedded systems, Huascar Espinoza, David Servat, Sébastien Gérard, Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge, SHARK '08, 2008, http://dblp.uni-trier.de/db/conf/icse/shark2008.html

[12] MARTE Mechanisms to Model Variability When Analyzing Embedded Software Product Lines, Lorea Belategi, Goiuria Sagardui and Leire Etxeberria, *in* SOFTWARE PRODUCT LINES: GOING BEYOND, Lecture Notes in Computer Science, 2010, Volume 6287/2010, 466-470, DOI: 10.1007/978-3-642-15579-6_38

[13] Guidelines and evolutions for MARTE use in automotive, ADAMS project, Deliverable 2.2 , version 3,june 2010, http://www.omgwiki.org/marte/sites/default/files/D2.2_T0+24_ADAMS_d20100606.pdf

[14] AADL Behavior Annex presentation (from SAE AS2C, Detroit, Apr 2006): http://www.aadl.info/aadl/documents/MamounBehaviorAnnexApril2006.pdf

[15] AADL Behavior Language Compliance and Application Program Interface (by the AADL Subcommittee, Apr 2007): http://aadl.sei.cmu.edu/aadl/documents/Behaviour_Annex1.6.pdf

[16] Steve Vestal, *Overview of the Error Modeling Annex for the Architecture Analysis and Design Language*, SAE AADL Meeting, Tucson, Jan 2007: http://aadl.sei.cmu.edu/aadl/documents/AADL%20Error%20Modeling%20Overview_Jan2007.pdf

[17] Peter Feiler & Ana Rugina: *Dependability Modeling with the Architecture Analysis & Design Language (AADL)*: http://www.sei.cmu.edu/library/abstracts/reports/07tn043.cfm

[18] Pierre Dissaux, Jérôme Legrand, Alain Plantec, Mickael Kerboeuf, Frank Singhoff: *AADL Design-Patterns and Tools for Modelling and Performance Analysis of Real-Time systems.* Embedded Real-time Software and Systems Conference (ERTS2010), May 2010.

[19] Performance Analysis with Cheddar: http://beru.univ-brest.fr/~singhoff/cheddar/

[20] Aldeida Aleti, Stefan Bjornander, Lars Grunske, Indika Meedeniya: *ArcheOpterix: An extendable tool for architecture optimization of AADL models.* MOMPES, pp.61-71, 2009 ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software, 2009.

[21] Youngseok Oh, Dan Hyung Lee, Sungwon Kang, Ji Hyun Lee. *Extended Architecture Analysis Description Language for Software Product Line Approach in Embedded Systems.* 5th IEEE/ACM International Conference on Formal Methods and Models for Codesign, 2007. MEMOCODE 2007. Jun 2007.

[22] Shin'ichi Shiraishi. *An AADL-Based Approach to Variability Modeling of Automotive Control Systems*. Model Driven Engineering Languages and Systems - Lecture Notes in Computer Science, 2010, Volume 6394/2010, 346-360, DOI: 10.1007/978-3-642-16145-2_24

[23] AADL Website: http://www.aadl.info/aadl/currentsite/

[24] Automotive SPICE, http://www.automotivespice.com

[25] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: *Final Draft International Standard ISO/FDIS 26262*, 2010.