

**ATESST**

Contract number: 2004 - 026976

Advancing Traffic Efficiency and Safety through Software Technology (ATESST)

Report type	Informal
Report name	The Modelling Approach in ATESST - Overview
Status	Final
Version number	1.0
Date of preparation	2007-03-12

Authors**Editor**

Henrik Lönn

E-mail

Henrik.Lonn@volvo.com

The Consortium

CEA (F)

DaimlerChrysler (D)

ETAS (D)

VCT/MGH (Hu)

Siemens (F)

Volvo(S)

Carmeq (G)

Mecel (S)

Volvo Car (S)

The Royal Institute of Technology (S) Technische Universität Berlin(D)

Revision chart and history log

Version	Date	Reason
0.1	2007-03-07	Initial document based on existing reports and presentations
1.0	2007-03-12	Final version

Table of contents

Advancing Traffic Efficiency and Safety through Software Technology (ATESST)	1
Authors	2
Revision chart and history log	3
Table of contents	4
1 Introduction	5
2 Overview of the EAST-ADL	6
2.1 EAST ADL Abstraction levels	6
2.2 Model organization and language constructs	7
2.2.1 <i>Structural organization</i>	8
2.2.2 <i>Behaviour</i>	9
2.2.3 <i>Requirements</i>	9
2.2.4 <i>Validation and Verification</i>	9
2.2.5 <i>Variant Handling</i>	9
3 ATESSST	10
3.1 Approach	10
3.2 EAST ADL refinement	11
3.2.1 <i>Requirements and analysis</i>	12
3.2.2 <i>Behavior</i>	13
3.2.3 <i>Variability and Re-use</i>	13
3.3 Related approaches	13
3.4 Relation to EUCAR and the Integrated Safety Projects	13
4 Comments and Conclusions	14
5 References	15

1 Introduction

This short description explains the EAST ADL and the intended refinement of the language in the ATESSST project. The document starts with an overview of the EAST ADL and goes on to discuss the modifications addressed by ATESSST.

2 Overview of the EAST-ADL

EAST-ADL is an architecture description language, dedicated to automotive embedded electronic systems, developed in the context of the ITEA cooperative project EAST-EEA (<http://www.east-eea.net/>) finished in 2004.

This language is intended to support the development of automotive embedded software, by capturing all the related engineering information. The scope is the embedded system (hardware and software) and its environment.

2.1 EAST ADL Abstraction levels

One approach to information organization for automotive system models is to introduce different levels of abstraction. This way, the system information can be considered with the amount of detail that is appropriate for the specification, analysis, implementation, etc. at hand (separation of concerns). This implicitly identifies different stages of an engineering process, although the detailed process definition can be company specific. It also provides a way to harmonize system descriptions by defining the expected level of detail and specification contents of models on a particular abstraction level.

This approach was used for EAST ADL. The figure below shows the five abstraction levels that are defined. The system can be considered at each of these abstraction levels, and the level of detail depends on which abstraction level is chosen. Note that there are additional aspects, orthogonal to the abstraction level. For example, the integration level concerns how much of the system is included in a particular model.

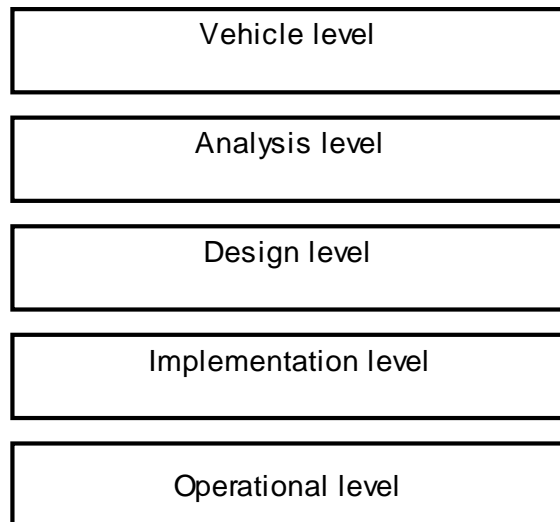


Figure 8: The abstraction levels of EAST ADL

The five abstraction levels are characterized as follows:

- *Vehicle Level*
On the Vehicle level, the Electronic/Electric architecture is only seen from the external perspective, i.e. as the end user perceives the vehicle and its functions. It is a description of the vehicle disregarding how the functionality is realized
- *Analysis Level*
This level abstractly defines the functions that realize the vehicle's features. The purpose is to support the analysis of the vehicle electronics as a basis for further refinement into a design.

- *Design Level*
The design level corresponds to a specification of the Electronic/Electric architecture with respect to the functional content and hardware platform.
- *Implementation level*
Implementation of the Electronic/Electric architecture is described in terms of software architecture for applications and middleware and the hardware platform it relies on.
- *Operational Level*
Operational level reflects the run-time electronic/electric architecture as it is deployed in a vehicle.

Note that the environment model spans all abstraction levels, and that requirements and variability constructs apply to modeling elements regardless of abstraction level.

2.2 Model organization and language constructs

The system model is structured into architectures and sub-models which represent information related to each abstraction level see Figure 1. There are many links between entities in different sub-models in order to document various dependencies between them.

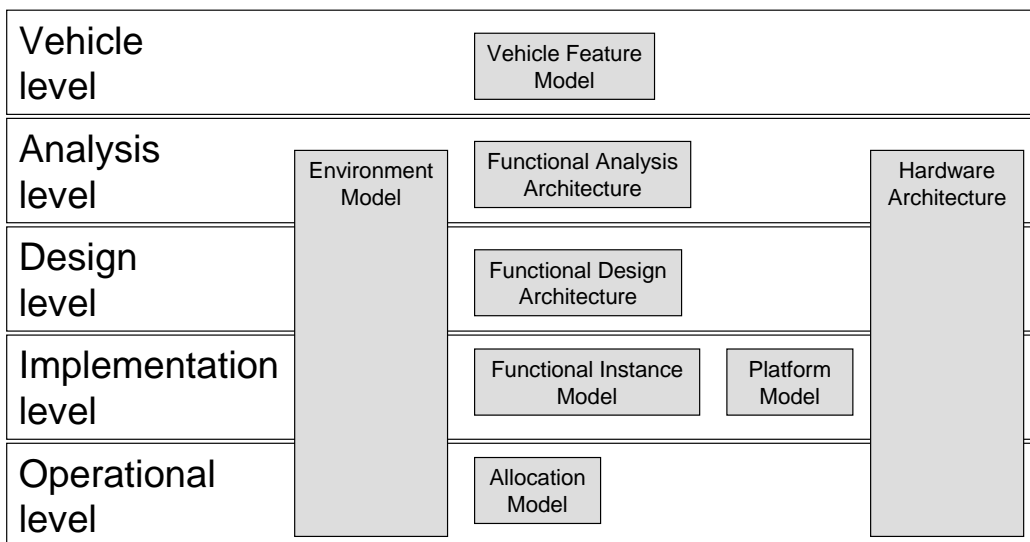


Figure 1. Organization of the EAST ADL1 System Model

The EAST-ADL language constructs support:

- vehicle feature modeling including variability concepts to support product families
- vehicle environment modeling to define context and perform validation
- Structural and behavioral modeling of software and hardware entities supporting refinement to code and binaries in the context of distributed system
- requirements modeling and tracing with all modeling entities
- other information part of the system description, such as a definition of component timing and failure modes, necessary for system verification purposes.

Below these aspects of the language will be further details.

2.2.1 Structural organization

- *Vehicle Feature Model* (Vehicle level)
The Vehicle Feature Model describes the mix of electronic functionalities for a particular vehicle - Electronic Features. The Electronic Features with associated requirements represents the user view of the vehicle. An Electronic Feature can be used to configuring the vehicle, as the inclusion or exclusion of a an EFeature can be used to decide the parameterization, hardware configuration, etc. It is also useful for communication with other departments, sites or companies, as the description is solution independent.
- *Functional Analysis Model* (Analysis level)
The Functional Analysis Model abstractly defines the functions that realize the vehicle's features. It reflects top level functional decomposition corresponding to a logical structure of the EE architecture, and contains requirements on the corresponding abstraction level. Further, it captures algorithms and control laws in an abstract manner (typically StateCharts, Simulink, etc.) and allows analysis from a functional/control engineering point of view. Most implementation details are hidden and it is thus useful for understanding the function w.r.t
 - behaviour
 - detection of contradicting requirements
 - criticality
 - interaction with other functions and the environmentThe Functional Analysis Model is also useful for communication with other departments or companies regarding what the functions should do.
- *Functional Design Architecture* (Design level)
The Functional Design Architecture is the specification of the Electronic/Electric architecture with respect to the functional content. It models functional decomposition corresponding to intended implementation. Behaviour is defined using state machines, data flow, etc and is typically basis for code generation using tools like Simulink/Target Link, ASCET, Rhapsody, etc. The FDA is Hardware independent but contains sensor and actuator interface definitions. Functional interfaces are defined in terms of engineering parameters like unit, resolution and representation. The requirements contained in the FDA are used to constrain the system design
- *Functional Instance Model* (Implementation level)
The function instance model is the implementation of the software. Software components are defined with behaviour in C code interfaces in terms of implemented variables and data types. The information required for tasks allocation, such as precedence, timing and dependencies is also included as well as requirements that constrain the system implementation.
- *Platform Model* (Implementation level)
The platform model contains the middleware definition of the system. The representation of middleware components and their external interfaces can be used to configure application software and the middleware itself.
- *The Allocation Model* (Operational level)
The allocation model is the run-time Electronic/Electric architecture deployed in a vehicle. Binaries represents the code and buffers and frames represent data exchange.
- *Environment Model* (Analysis level, Design level, Implementation level and Operational level)
This is the plant model containing a representation of the environment of the EE system. It is used to validate the external interfaces of the system and for simulation. Note that there is only one Environment Model i.e. there are no internal abstraction levels of the model.
- *Hardware Architecture* (Design level, Implementation level and Operational level)
The Hardware Architecture contains the ECUs, networks, sensors, actuators, etc. necessary for configuring the software. It is also used for understanding the system configuration and how it influences software. For example, bus redundancy is visible. Even

if not explicitly used at Analysis level it is necessary to know the general ideas of the hardware for the Analysis level. Note that there is only one Hardware Architecture i.e. there are no internal abstraction levels of the architecture.

2.2.2 Behaviour

Behaviour is attached to the structural entities (functions) of the system model as a separate entity. The purpose is to allow different representations of behaviour depending on what is suitable for the respective domain. For example, Simulink may be appropriate for chassis or powertrain systems, while UML statecharts is better for the body domain. In some cases a model representation of behaviour is omitted, and the C-code representation is used alone. Behaviour is attached to functions of the Functional Analysis Architecture and Functional Design Architecture, while the actual code is the behaviour representation in the Function Instance Model.

2.2.3 Requirements

Requirements are model entities of the EAST ADL which can be associated to any other model entity. By attaching requirements to model entities and linking requirements to each other, traceability and requirements allocation is supported. Requirements are part of all EAST ADL sub-models, but the abstraction level of the requirement should comply with its target entities.

2.2.4 Validation and Verification

Validation and Verification information is model information that relates both to requirements (that are verified) and other model entities (that meet the requirement). Verification and validation require a lot of book keeping, e.g. regarding what is verified and how and the outcome. Since V&V concerns requirements, this information is applicable to all sub-models of the EAST ADL.

2.2.5 Variant Handling

Variability is a major concern for automotive systems in general and the software and electronics in particular. Providing means to define and trace how variability influences the systems is therefore very important. Variant handling is part of all sub-models.

3 ATESSST

The ATESSST project is aimed at refining the EAST-ADL language in the context of dependability concerns, aligning with OMG standards and the new automotive domain standardization AUTOSAR (<http://www.autosar.org/>).

To cover dependable systems, requirement constructs will be enriched to satisfy the needs of different integrity levels and the modeling entities will be refined to support necessary analysis methods, and an engineering process for safety. Transversal to these concepts, with the same consideration for dependability, the variability constructs of EAST-ADL will be improved to support vehicle product lines, the major productivity driver in automotive industry.

3.1 Approach

An iterative approach was chosen to refine the EAST ADL, see Figure 2. Needs on the language are collected through scenarios and transformed to requirements. Language concepts to meet these requirements are then defined or refined and integrated in a domain model. To be useful in UML tools, and to align with existing UML2 concepts, a profile is defined. The profile can be seen as an implementation (for UML2) of the language specification defined in the EAST ADL2 domain model.

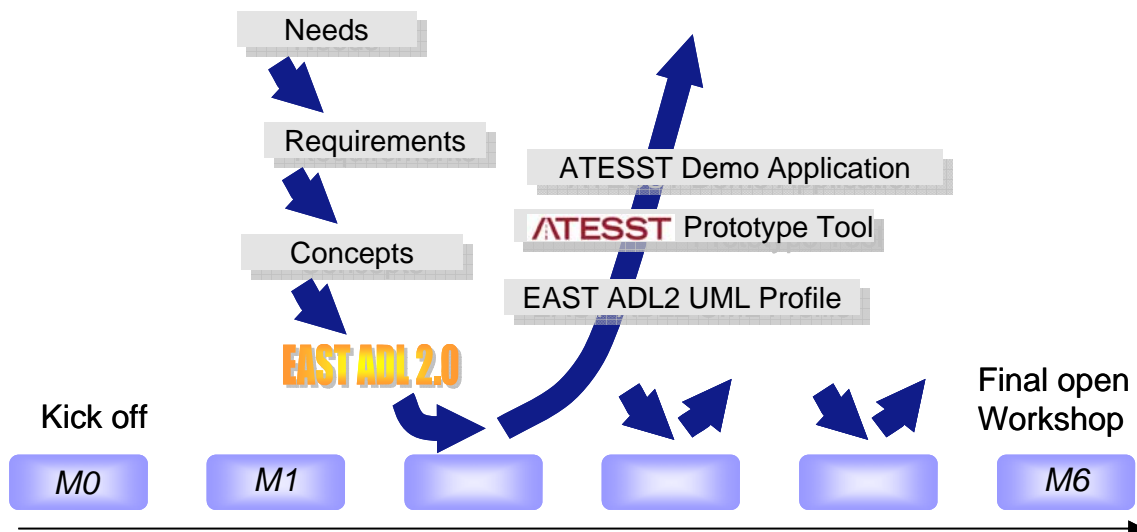


Figure 2. The iterative approach to language definition in ATESSST

To be able to evaluate the language concepts, an Eclipse-based UML2 editor is implemented, see Figure 3. In addition to support UML2 specialization by profiles, this approach allows EAST ADL-specific plugins to be defined in the Eclipse framework. Plugins for variability, error analysis, data exchange, etc. will be implemented.

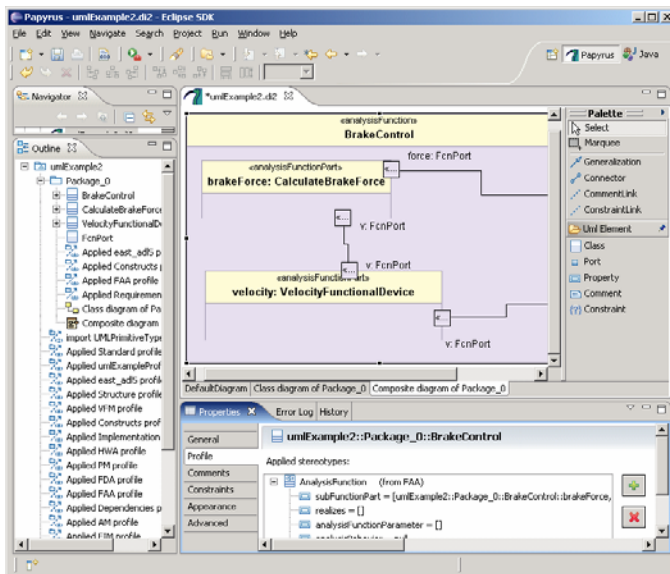


Figure 3. Screen-shot from the Eclipse-based ATESSST prototype tool (Papyrus)

Finally, to validate the modelling concepts, a active safety demonstrator is implemented in EAST ADL using the prototype tool. The draft of the demonstrator was sketched in Simulink for convenience, see Figure 4, and later transformed to EAST ADL.

Experiences from each step of the iteration is the basis for the next iteration which gradually extend the scope of the language extension.

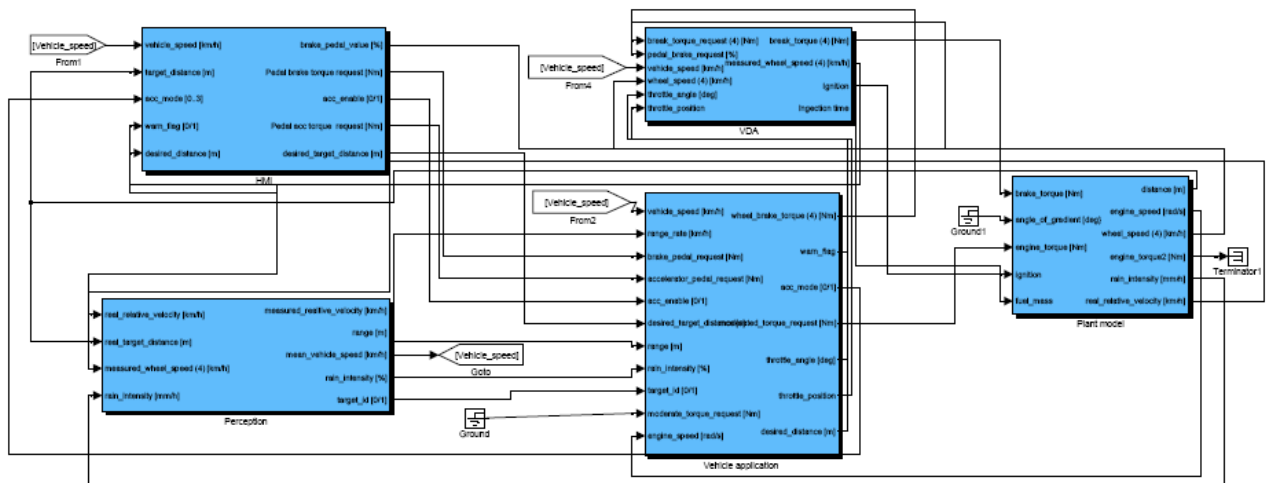


Figure 4. Demonstrator application

3.2 EAST ADL refinement

The abstraction levels of EAST ADL are kept intact so far. Figure 5, shows the intention that the EE architecture is clearly separated from the plant model. This makes it possible to have the same interface to the plant model regardless of abstraction level and development phase. It also allows having the same environment model for several models of the EE architecture, except that the analysis at hand may require different variants and detail level of the plant model.

This allows re-use of the plant model for several same Also, with a slight

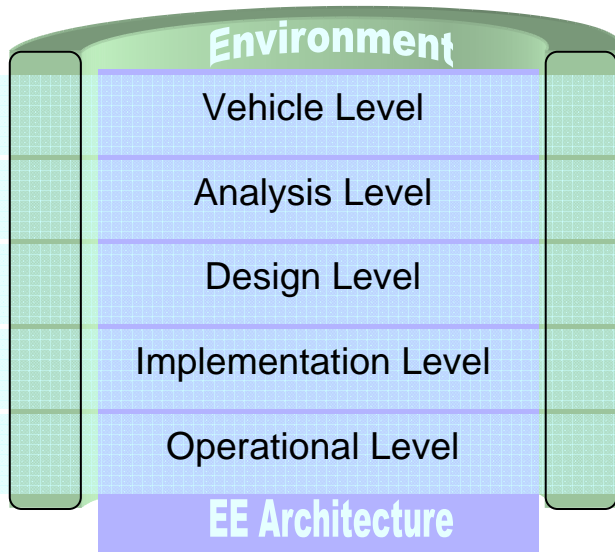


Figure 5. EAST ADL EE architecture vs. plant models/environment

A slightly modified modelling structure is currently proposed for EAST ADL2, Figure 6. New containers are added for each abstraction level to better group relations belonging to a certain abstraction level. Also, new compositions are added to host the Autosar constructs.

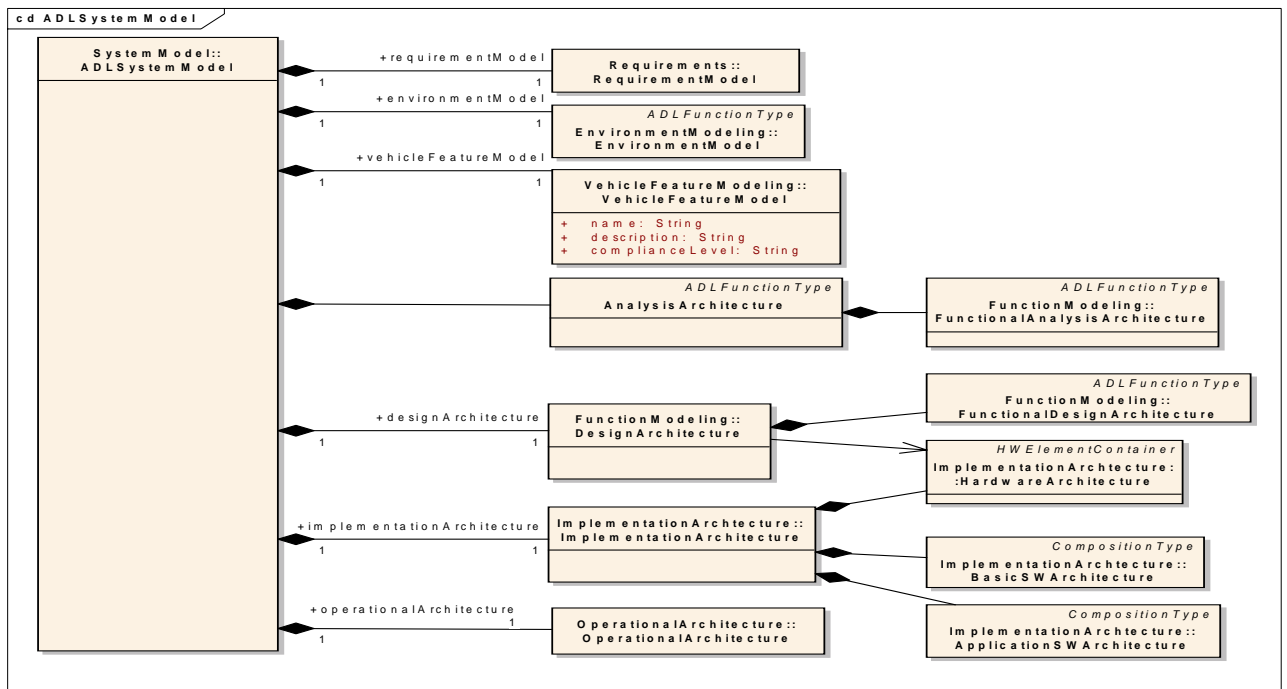


Figure 6. Preliminary organization of the EAST ADL2 System Model

3.2.1 Requirements and analysis

Requirements constructs to support safety related requirements and V&V information are added. Support for error models to support fault tree analysis and FMEA are defined.

3.2.2 Behavior

Behavioral constructs to support behavior defined in external tools such as Simulink or ASCET are added. Also, native constructs are investigated to support state machines and data flow behavior in an appropriate manner for the EE architecture end the plant. In particular, the relation to AUTOSAR task model is addressed.

3.2.3 Variability and Re-use

Concepts for feature definition and component level variability is refined compared to EAST ADL1. Experiences from product family research, feature modelling and variability propagation are used to improve the variability constructs.

3.3 Related approaches

The EAST ADL is refined by adding concepts from several related approaches. Below, some of these are listed:

AUTOSAR: Being the expected dominant approach to modelling of automotive software components and their hardware platform, AUTOSAR harmonization is critical. The implementation level of EAST ADL is therefore replaced by AUTOSAR concepts, to allow integration of AUTOSAR entities in the same system model as the other EAST ADL constructs. This way a traceable information model is achieved from top level user requirements down to implementation.

MARTE: Marte concepts for embedded systems modelling will be part of the EAST ADL profile. In addition, the EAST ADL profile will be an annex to the MARTE profile covering automotive embedded systems.

SysML: EAST ADL2 is a specialization of SysML for those parts that are applicable for automotive systems.

AADL: AADL is dedicated to an implementation level description of embedded systems, with a primary focus on aerospace. Because AUTOSAR to some extent overlaps with AADL, the primary harmonization is devoted to AUTOSAR. However, certain concepts on the functional modelling level as well as error modelling are taken from the AADL.

3.4 Relation to EUCAR and the Integrated Safety Projects

ATESST provides technology for the development process, and therefore complements the application-oriented Integrated Safety Projects such as AIDE and PReVENT. EASIS is different in that it addresses platforms and process to a larger extent than the previous projects. ATESSST adds a modelling approach to the EASIS concepts, which supports the safety analysis methodology and development process defined by EASIS. The demonstrator architecture is based on the Integrated Safety Projects' architecture approach, and one of the demonstrator's components is based on a PReVENT application.

4 Comments and Conclusions

The main driver for the ATESSST project and the motivation for EAST ADL is the vision to provide an integrated system modelling approach for automotive electronics.

Adequate information handling is a pre-requisite for correct systems and for coping with the complexity of advanced safety functions. Word & Excel specifications are not enough to meet the complex modelling needs, and the demand on consistent and traceable specifications. Similarly, domain-specific tools such as Simulink and StateCharts do not meet the information integration needs. It is necessary to have a common system model for integration, while external tools can be used to manage the details. An alternative is to use internal tools similar to PDM systems to manage the company internal data. However, this approach does not promote tool and methodology evolution and data exchange.

While defining a standardized modelling approach, it is important to relate to and harmonize with existing approaches. AUTOSAR is standardizing the specifications for automotive software components and their integration in hardware. Harmonization with AUTOSAR is therefore critical. As a complement to AUTOSAR, ATESSST adds modelling means on the functional specification level and above, as well as orthogonal aspects such as requirements, variability and plant/environment models.

SysML is proposed from the OMG as a means for modelling Systems Engineering information. SysML concepts are relevant and should be used where applicable. However, SysML is general and not specialized for embedded systems or Automotive systems. The ATESSST approach is therefore to continue the alignment with SysML that was already initiated during the EAST-EEA project.

5 References

- [1] EAST-EEA Embedded Electronic Architecture: Deliverable D3.6 Definition of language for automotive embedded electronic architecture, Version 1.02, 30.06.2004: www.east-eea.net
- [2] Description of Work: "Advancing Traffic Efficiency and Safety through Software Technology", IST FP6 Project 026976, Annex I - "Description of Work"