



Contract number: 2004 - 026976

Advancing Traffic Efficiency and Safety through Software Technology (ATESST)

Report type	Deliverable D6.1.1
Report name	Elicitation of overall needs and requirements on the ADL – Part II State of the art and state of practice
Disemination level	PU
Status	Final
Version number	1.1
Date of preparation	2006-04-21

Authors**Editor**

Martin Törngren

E-mail

KTH (martin@md.kth.se)

Authors

Lars-Olof Berntsson

E-mailVolvo Technology Corporation
(lars-olof.berntsson@volvo.com)

DeJiu Chen

KTH (chen@md.kth.se)

Philippe Cuenot

Siemens VDO
(philippe.cuenot@siemens.com)

Joerg Donandt

Daimler Chrysler AG
(joerg.donandt@daimlerchrysler.com)

Sebastien Gerard

CEA (sebastien.gerard@cea.fr)

Rolf Johansson

VCT/MGH (rolf_johansson@mentor.com)

Henrik Lönn

Volvo Technology Corporation
(henrik.lonn@volvo.com)

David Servat

CEA (david.servat@cea.fr)

Martin Törngren

KTH (martin@md.kth.se)

The Consortium

Volvo(S)

DaimlerChrysler (D)

Siemens (F)

ETAS (D)

Mentor Graphics (Hu)

CEA (F)

The Royal Institute of Technology (S)

Technische Universität Berlin (D)

Carmaq GmbH (D)

Volvo Cars (S)

Mecel (S)

Revision chart and history log

Version	Date	Reason
V0.01	2006-02-03	Initial draft of report content and request for contributions
V0.1	2006-03-15	First draft of report with structure and introduction
V0.2	2006-03-20	Elaborated draft of report where inputs from partners are included.
V0.3	2006-03-22	Further refinement of structuring and text
V0.4	2006-03-31	Updates after feedback at the ATESST M1 meeting in Berlin
V1.0-060410	2006-04-10	First release for public review
V1.0	2006-04-21	Update based on review comments
V1.1	2007-07-04	Updated according to observations/comments from EC review 29/03/2007, Review no. 1 covering project month 1 to 12 (01/01/06 – 31/12/06)

Executive Summary

The primary aim of the ATESST project is the development of an automotive description language, EAST-ADL2.0. This report provides a contextual perspective to this work by describing industrial practice and research efforts in the area of model based development and how they relate to the EAST-ADL.

Table of contents

Authors	2
Revision chart and history log.....	3
Executive Summary.....	4
Table of contents	5
List of Figures	6
1 Introduction	7
1.1 Characteristics of Automotive Embedded Systems.....	8
1.2 Model based development.....	11
2 Snapshots from state of practice in the development of automotive embedded systems	13
2.1 Modeling and tool support in state of practice	13
2.2 State of practice for model-based software development - control applications	14
2.3 Example of state of practice from Automotive supplier	16
3 State of the art efforts related to the EAST-ADL	19
3.1 Which efforts? Characterization of efforts and delimitation	19
3.2 Details on selected research efforts	20
3.2.1 AADL and ASSERT	20
3.2.2 AIDE.....	21
3.2.3 AUTOSAR - Automotive Open System Architecture	22
3.2.4 EASIS	22
3.2.5 Families.....	23
3.2.6 IEC 61508 and the forthcoming ISO 26262.....	24
3.2.7 MISRA.....	26
3.2.8 OMG related work.....	28
3.2.9 PROTES	29
3.2.10 RESPONSE/PREVENT	29
4 Added value of the EAST-ADL2.0 with respect to related (research&industrial) efforts.....	32
4.1 AADL	32
4.2 Autosar	32
4.3 MARTE	33
4.4 SysML.....	33
5 Contribution to overall ATESST objectives.....	35
6 Conclusions	36
7 References	37

List of Figures

Figure 1. The electronic architecture of Volvo XC90.	9
Figure 2. Typical software architecture of an ECU.	9
Figure 3. Various stages in controller development.	15
Figure 4. The conception view of safety critical systems adopted in IEC 61508	24

1 Introduction

The primary aim of the ATESSST project is the development of an automotive description language, EAST-ADL2.0. This report provides a contextual perspective to this work by describing industrial practice and research efforts in the area of model based development and how they relate to the EAST-ADL2.0.

To define the scope of this report it is relevant to recall the detailed objectives of ATESSST as described in Section 2.2 in the Description of Work – an extract is as follows:

“The overall objective of this proposed research is to improve the development, verification and validation of software-dependent systems in the road transport context, explicitly considering requirements and safety constraints through a rigorous, model-based development technology and process combining approaches from control- and embedded software engineering. ...

Below is a short description of the concrete objectives of ATESSST:

- *The structural descriptions of EAST ADL will be harmonized with existing approaches, as far as possible, i.e. the AUTOSAR initiative, the OMG (UML2, UML profile for MARTE and SysML) and with the SAE AADL. This will be the back bone onto which further language constructs can be attached. In this context, ATESSST results will be concretized in a new major release of the EAST-ADL: the EAST-ADL2.0.*
- *The support for Requirements and V&V in EAST ADL will be refined. In particular, the V&V aspects of the interaction between the embedded system, its environment and the application will be investigated. Safety related requirements are also added to the EAST-ADL.*
- *Adequate behavioural modelling for EAST-ADL2.0 will be defined. The purpose is to capture behaviour and algorithms of the vehicle systems as well as the environment. This includes a native behavioural notation that allows simulation and verification within the defined system model. This support is currently not part of EAST-ADL. It also concerns means to integrate external tools in an appropriate manner. This means that the links to Mathworks’ Simulink, ETAS’ Ascet and other domain-specific modelling tools will be defined and validated. This enables interfaces between external tools and an ATESSST environment to be realized.*
- *Reuse and variability management with regard to safety aspects will be investigated. Identified language constructs will be added to EAST-ADL.*

... Together, these objectives form a major revised version of the EAST-ADL (EAST-ADL2.0), along with descriptions of an intended methodology.”

Given these goals of the EAST-ADL2.0 the scope of the state of the art study should be confined to model based development approaches, in particular those that address verification, validation, in the context of safety related systems, and to standards/guidelines applicable for such approaches.

Requirements on the EAST-ADL2.0 are related to the product properties, such as complex relationships among product entities, and to the development process, which provides generic needs for the support of design, verification and validation activities.

To set the scene for the subsequent discussion of industrial development practice and for related model based development efforts, we begin in Section 2.1 by briefly describing the special requirements and characteristics of automotive embedded systems. In Section 2.2 we describe the essence of model based development and overall challenges. Section 3 then provides snapshots from the widely varying current state of practice in the modeling and documentation of automotive embedded systems. Section 3 also discusses the relations between the aims of the ATESSST project and the industrial needs. Section 4 provides an overview of related academic and industrial efforts. A central message of the report is provided in Section 5 where we summarize and discuss the relation of the EAST-ADL2.0 to the most closely related efforts. Finally, Section 6 concludes the report by discussing how this work will be continued in the project.

In summary it is concluded that the current state of the art and industrial practices are weak when it comes to modeling electronic architectures. The goals of the EAST-ADL2.0 do meet industrial needs for a formalized description language that supports the development of automotive embedded systems. It is also concluded that the EAST-ADL2.0 – as compared to other state of the art efforts – aims to provide added-value with explicit support for

- automotive embedded systems, providing a broad set of abstractions, properties and views of automotive embedded systems, from requirements to implementation, considering also reuse and variability.
- system analysis, including analysis of safety related properties as well as other structure and behavior related properties that include the interaction between the embedded system, its environment and the application.
- system representations that are semantically compatible with models used in commonly used domain specific design tools such as the Mathworks' Simulink and ETAS' Ascet. Model transformation and exchange with these tools will be demonstrated, illustrating how interfaces between external tools and the prototype ATESSST tool environment can be realized.

1.1 Characteristics of Automotive Embedded Systems

Automotive embedded systems have evolved enormously over the past decades. For example, the first commercial anti-brake locking system (ABS) of Bosch was introduced by Mercedes in 1978. The ABS system improves the braking performance and is today a standard feature in the automotive industry. The system was first presented in 1970 but at that time the available electronics could not cope with the ABS requirements delaying the commercial introduction by 8 years.

To further illustrate the dramatic introduction of computer based embedded control, consider the fact that a Mercedes car in 1986 contained six microprocessors; these were implemented as six stand-alone controllers (in the automotive industry these are referred to as ECUs, standing for Electronic Control Unit). In 1998 a corresponding Mercedes car contained some 60 microprocessor systems, together forming a distributed system including four networks (not to mention in addition some 113 electrical motors!).

The introduction of computer based embedded control has been driven both from the technical viewpoint, that of improving performance or introducing entirely new functions, and by market demands. At the same time the costs for development of electronics of which the vast part is software development increases dramatically, and today it is reaching about 40% of total costs with a tendency for further increase.

A typical example of an automotive embedded system is shown in Fig. 1, illustrating the electronic architecture of the Volvo XC90. The boxes in the figure represent Electronic Control Units (ECUs) and the lines represent communication networks. The intent of the figure is to show the complexity rather than the details. The maximum configuration contains about 40 ECUs. They are connected mainly by two CAN networks, one for power-train and one for body functionality. From some of the nodes, LIN sub networks are used to connect slave nodes into a subsystem. The other main structure is a MOST ring, connecting the infotainment nodes together, with a gateway to the CAN network for limited data exchange. The different networks illustrate the typical automotive domains, including vehicle dynamics control (left network, characterized by strict real-time requirements and safety related motion control), body electronics (including door control, climate control and instrument cluster), and infotainment and telematics. Through this separation, the more critical power-train functions on the CAN network are protected from possible disturbances from the infotainment system.

The diagnostics access to the entire car is via a single connection to one ECU. The figure shows approximately how the ECUs are placed in various locations in the car. The partitioning of functionality is decided by the location of the sensors and actuators used, but also by the combinations of optional variants that are possible. If a car is sold with only a subset of the full functionality, the amount of physical hardware installed should be limited to the minimum necessary.

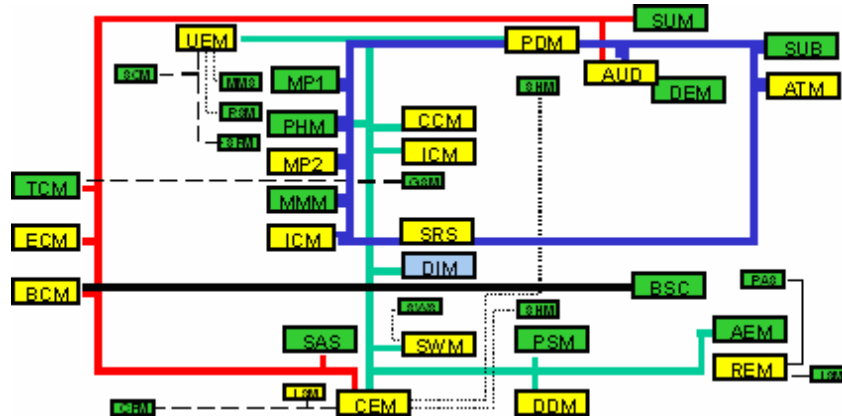


Figure 1. The electronic architecture of Volvo XC90.

Many of the ECUs of a modern vehicle are provided by external suppliers, who work with many different car companies (or OEMs, original equipment manufacturers), providing similar parts. The role of the OEM is thus to provide specifications for the suppliers, so that the component will fit a particular car, and to integrate the components into a product. Traditionally, suppliers have developed physical parts, but in modern cars they also provide software. As the computational power of the electronic control units (ECUs) increase, it will be more common to include software from several suppliers in the same nodes.

The current development trends in automotive software call for increasing standardization of the software structure in the nodes. The need to integrate software from different suppliers, supporting dependable real-time execution, and managing changes all call for a well-defined structure. The node architecture (see Figure 2) includes several important parts:

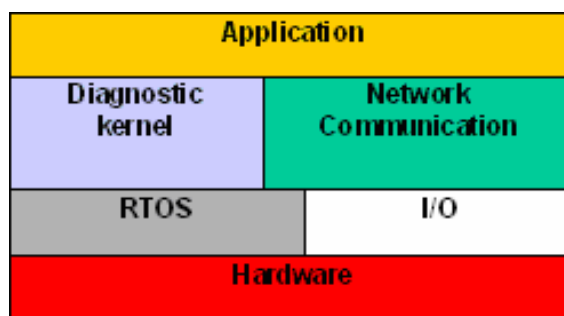


Figure 2. Typical software architecture of an ECU.

Diagnostic kernels provide an implementation of the diagnostic services that each node must implement to act as a client towards the off-board diagnostic tool. It relies on the communication software to access the networks and on the operating system to schedule diagnostic activities so that it does not interfere with the application functionality. Network communication software provides a layer between the hardware and the application software, so that communication can be described at a high level of abstraction in the application, regardless of the low-level

mechanisms employed to send data between the nodes. Real-Time Operating Systems (RTOS) provide services for task scheduling and synchronization.

All these components interact with each other and with the application, and must therefore have standardized interfaces, and at the same time provide the required flexibility. To minimize the use of hardware resources, the components are configurable to only include the parts that are really necessary in each particular instantiation.

For future system development, an important aspect is to create a more flexible software partitioning. The main use for this is probably not to find the optimal partitioning for each car on a given platform, since that would create too much work on the verification side, but to allow parts of the software to be reused from one platform to the next. This puts even higher demands on the node architecture, since the application must be totally independent from the hardware, through a standardized interface that is stable over time. Therefore, further standardization work is pursued within the Autosar initiative [1].

Cars are typically manufactured in volumes in the order of millions per year. To achieve these volumes, and still offer the customer a wide range of choices, the products are built on platforms that contain common technology that has the flexibility to adapt to different kinds of cars. As an example, the Volvo XC90, which appeared in 2002, is based on the same platform as four previous Volvos launched since 1998.

Automotive embedded systems are further characterized by the following:

- **Users.** In contrast to many other advanced machines, such as airplanes and medical devices, automotive products are utilized by all us. This has an important impact on the usability, service and dependability required of the products.
- **Dependability requirements.** Automotive embedded systems have a fairly long life time and users expect the vehicles to function over extended periods of time, leading to strict requirements on reliability, availability and maintenance. Automotive control systems are safety related. Not only is the control system required to operate reliably; the design of the system and its context must be carefully analyzed to consider what might go wrong, and what the system should do in such cases. In addition, security is becoming of increasing importance because of the possibilities and relative ease with which embedded control systems behavior can be modified, e.g. by replacing memories/chips or by network intrusion.
- **Heterogeneity.** As depicted by Fig. 1, automotive embedded systems are heterogeneous. They handle many different types of tasks with widely varying requirements. For example, the motion control related ECUs include functionality that can be characterized as hybrid systems, being composed of components that are best described by continuous- and discrete-time dynamic systems and finite state machines. Motion control is one central part of ECS. Although it's absolute size e.g. in terms of lines of code typically is relatively small compared to other functionality, the motion control functionality is coming along with real-time constraints, environment dependencies, and safety criticality. To handle this heterogeneity the embedded systems are normally structured into a system platform and applications, each with their own hierarchy in order to facilitate changes and reuse. Responsibilities of the system platform include for example logging, communication services and drivers for sensor readings. For the application there will be activities such as motion control, estimation of the environment state, and human machine communication.
- **Real-time constraints.** These constraints arise due to interactions with the environment. From control system specifications, for example referring to required speeds of motion, the timing requirements on the embedded control system can be derived. The speed (or bandwidth) of the closed loop system will provide requirements on the timing of the controller, including the sampling periods and delays that can be allowed. These properties can also be taken into account in the control design, however, providing an additional dependency between the controller and its implementation.

- **Resource constrained implementations.** Automotive embedded systems are often highly resource constrained because of the large series being produced. In such applications, trade-offs between the system behavior (quality of service) and the resources required (processing, memory and power) is essential.
- **Distributed systems.** There has over the last decades been a strong trend to connect stand-alone controllers by networks, forming distributed systems. Another and closely related trend has been modularization, where for example, an electronic control unit is physically integrated into an engine, forming a sort of mechatronic module. Combining the concepts of networks and mechatronic modules makes it possible to reduce both the cabling and the number of connectors, the result of which is facilitated production and increased reliability. Distributed control systems first appeared in process control, and later in the 80s in aerospace, and in the 90s in the automotive industry. Distributed systems are characterized by the mapping problem, i.e. the need to assign functions to different nodes of a distributed system, to define the tasks of the system, and their implementation in software and/or hardware.
- **Tight coupling to the environment.** The tight coupling between the control system and the controlled process is manifested in several ways. Apart from aspects related to physical integration and protection against a harsh environment, the control system is also fundamentally related to the controlled process. Typically, models of the environment are used in control design. In many cases, the control algorithms are synthesized from a validated model of the controlled system. In other cases, the controller parameters are tuned based on the overall system behavior. For control systems this creates a dependency between the environment and the control system, creating a kind of contract between these two entities. Another type of environment coupling exists with humans interacting with the embedded control system. A driver “in the loop” is typical for vehicular systems. The situation arises where conflicts can occur – who is deciding the motion of the vehicle at any given point in time? Careful analysis is required and special care has to be given to the human/machine interface.
- **Parallel activities and triggering.** Since the real world is truly parallel, there is typically a need to describe and handle several parallel activities. A typical control system normally includes both time- and event-triggered activities. In many cases, time-triggering follows naturally from the development of discrete time (sampled data) functions. However, in other cases the controlled process can be inherently event-triggered. This is the case for inherently sampled systems, one example being control of injection in a combustion engine; the point in time of injection depends on the speed and angular position of the engine parts. Event-triggered functions thus include those who are inherently sampled and other functions who are not dictated or preferably implemented as periodic activities.

All in all, the use of embedded control systems has paved the way for large improvements of machinery in terms of enhanced performance, flexible tailoring of product variants, and the introduction of completely new functionality such as for active safety control in cars. As a consequence, and as partly touched upon in section 1, product complexity is becoming a crucial issue in system development. Systems integration is today a serious problem in the automotive industry. This increased product complexity calls for more mature engineering approaches including the use of model and component based development.

1.2 Model based development

It is well known that increased system complexity requires increasing abstraction levels for humans to deal with and develop such systems. The essence of model based development has the aim to provide

- formalized descriptions providing abilities for automated analysis and synthesis
- abstraction capabilities and graphical representations, facilitating communication among system stake-holders

- possibilities for reuse of designs, and of analysis and synthesis capabilities
- the basis for improved solution space exploration, verification and validation

Model based development is common practice in mature domains such as mechanical engineering and control engineering, and supported by computer aided engineering tools. Even though the practice is advanced there are still limitations. For example, in control engineering, the formalized descriptions deal primarily with control system behavior in terms of steady-state and transient behavior of sets of coupled differential equations. There is little, or much less concern, with issues such as implementation oriented structuring (e.g. how to partition algorithms into software modules) and how the hardware/software implementation will affect the control system behavior, e.g. due to time-varying end-to-end delays and transient hardware faults.

The situation in software engineering is less mature. For example, there are companies and suppliers in the automotive industry that are certified for level 3 maturity but the maturity of software development in automotive industry has just about reached the lowest two levels of the SW-CMM [2]. The use of software is not new in the automotive industry but the ability to consider networked systems with a proper process is still in its infancy.

Model based development of software is therefore an evolving area with many research efforts in place (see section 3 for an overview).

2 Snapshots from state of practice in the development of automotive embedded systems

This section provides an introductory overview of model based development practices in the automotive industry. The overview is not complete but has the intention provide representative snapshots to illustrate the widely varying state of practice and industrial needs. The interested reader may refer to for example [3] and [4] for more snapshots.

2.1 Modeling and tool support in state of practice

In general, development practices vary to a great extent over different companies, and across the automotive domains. This is not surprising because of the heterogeneity of the automotive embedded systems, as described in Section 1.1. While some domains are characterized by model based approaches supporting the development, other domains still mainly rely on written documents for specifications and hand-written code for software development. Although some analysis is possible on the code level, this level is not suitable for communication among developers and does not support early analysis and design. A particular complication and challenge is provided by the differences in traditions and also characteristics among the domains. An essential facet of the characteristics is the differing models of computation, represented by closed-loop control in vehicle dynamics (continuous and discrete-time dynamics) and logic/state machines in e.g. body electronics (discrete event dynamic systems).

For control systems development in the automotive industry, model based development is in many areas already the standard design approach; however, the adoption and extent varies between different companies and subsystems. CAE tools supporting modeling, simulation and rapid control prototyping (RCP) largely facilitate development even without available mechanics and electronics hardware, and provide means for control system verification and validation, in the lab, as well as in-vehicle [5].

Companies with more mature processes utilize tool chains typically starting from functional design (e.g. using Matlab/Simulink/Stateflow), using rapid prototyping (through code generation and prototyping hardware), software in the loop simulation, production target code generation, and reuse of plant models in hardware in the loop simulation. Less advanced companies still tend to use e.g. Matlab/Simulink, but with no or little connection to the embedded systems implementation.

Model based testing is becoming increasingly used, where one example is the use of hardware in the loop simulation for both subsystem as well as system integration testing. In a hardware-in-the-loop simulator, the computer control system environment (i.e. the vehicle, road, driver actions as well as other relevant environment entities) is simulated in real-time, enabling system testing.

However there are also control domains, such as automotive engine control, which rely heavily on look-up tables and calibration of systems for control purposes – i.e. with little tradition of model based control.

With respect to software modeling, the use of the UML has been slowly increasing. However, the collected experience of the consortium is that the use of the UML is not widely spread for automotive embedded systems. Subsets of UML are used for certain dedicated purposes, mainly involving documentation (e.g. use-cases, MSCs). There are also cases where the use of UML tools is more elaborated including code generation and model level debugging.

The introduction of tool chains supporting model based control engineering is not unproblematic and is strongly related to and affected by organization, process and technology constraints. Introducing tool chains causes a reliance and dependence on particular tool vendors and requires training of personnel. For OEMs, transitioning from specification of control systems to actually implementing them is a large step and has many implications including maintenance of in-vehicle

software and possibly also a larger responsibility. On the other hand, it gives the OEMs better control of vital vehicle functionality.

Many OEMs only specify control systems that are in turn developed by subsystem suppliers (for example; e.g. BMW top models have some 60 ECUs where most are developed by some 30 subsystem suppliers). Increasingly, models such as Simulink diagrams are used in the communication between organizations.

Tools for model based development of automotive embedded systems (e.g. as manifested by software or control engineering tools), today only support the handling of a limited number of aspects (or views). A complete development chain typically involves a multitude of tools and pieces of information that are loosely integrated. Examples of problems in this area include efforts trying to combine Simulink (or Scade or something else) with UML for software, coordinating function specification tools with specific engineering analysis models/tools (e.g. for safety and reliability), and coordinating control engineering tools with architectural design tools. These problems have led to intensive research and industrial efforts, as described in Section 3.

There is consequently today a strong need for standardized ways of

- describing automotive embedded systems, to support communication
- managing the information involved in automotive embedded systems development and integrating the disparate sources of information, e.g. represented by UML, Simulink and safety analysis models, capturing different aspects of the system to be developed
- supporting control engineering, software issues and implementation in embedded distributed computer systems
- supporting various types of formal analysis techniques from different disciplines.

2.2 State of practice for model-based software development - control applications

For control-oriented applications such as chassis systems, engine and powertrain and climate control, a common practice is to use control design tools like Simulink for applications development.

In this approach, control engineers experiment with early and simplified control approaches and refine them while exploring the control strategies. This is done by simulating the control law together with a simulated environment model (plant model). As the control algorithm matures, it is possible to try it on real systems in a Rapid Control Prototyping platform. This means that the prototypical control design runs on an execution platform with access to sensors, actuators and control signals of the real systems and possibly also the vehicle itself. For example, an early design of a new cruise control algorithm can be applied in a real car: The controller model executes in the Rapid Control Prototyping platform. The engine interface and various status information is received from the car's electrical system; cruise control commands from existing and additional buttons are connected to the prototype platform; existing and additional indicators may be connected to the prototype platform over busses or direct connections. The more realistic interfaces that are used in this set-up, the more realistic the control design that is under development becomes.

This development step ends with a fairly mature control design that is still only a model of the real software. To capitalize on the design that is made, and the early validation that has been done during simulation and prototyping, it is common practice to also use the model of the controller to (more or less) automatically generate code. The generated code corresponds to the application software, and needs to be interfaced to system software for various inputs, outputs and services. The integration of application software to the system software can be done manually or using tools that are more oriented towards software development, such as UML based tools (E.g. Ilogix

Rhapsody or Telelogic Tao). In this case, the Simulink code is integrated with the system software, just like a piece of manually written code.

The stages of the development are illustrated in Fig. 3.

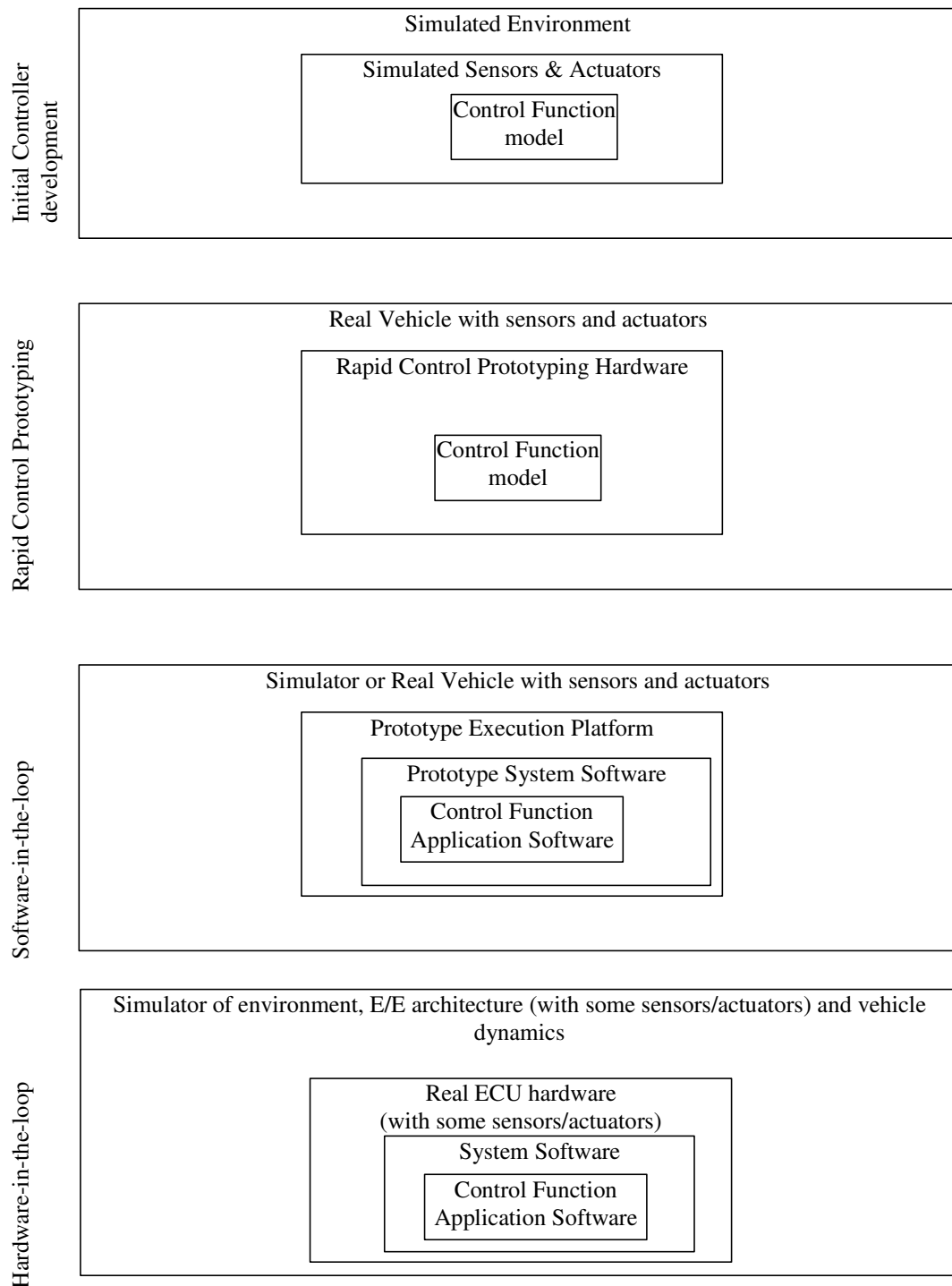


Figure 3. Various stages in controller development.

Once the application software is generated, a possible approach to do early integration validation is to use software-in-the-loop. This means that the application software is integrated with a

prototypical environment corresponding to the real execution environment with hardware and system software.

At this stage, the application software should be ready for integration in a real ECU with real system software. To validate the system, the ECU may be integrated in a simulated E/E-architecture and environment before the actual installation in a vehicle. This is called Hardware-in-the-loop.

A complication with this relatively common approach for model based control systems development is that the control design made during the early development stages has a focus on understanding and validating the control function. Implementation aspects may call for other design choices than in the pure control related design. It is thus common to replace the early model with a detailed design of the application software. It is still a model, but it is more specialized towards the target architecture.

To summarize, there are 7 steps:

1. Development of executable model of controller
2. Validation/refinement of controller model in a simulated environment
3. Validation/refinement of controller model in a real environment
4. Code generation of application software corresponding to controller
5. Integration of application software in simulated execution platform with real/simulated environment – Software-in-the-loop
6. Integration of application software in real execution platform with simulated EE architecture and environment – Hardware-in-the-loop
7. Integration of ECU in real EE architecture in vehicle

2.3 Example of state of practice from Automotive supplier

This example illustrates a state of practice for model-based software development in the context of automotive supplier business. It is concerned with control applications for different features in a sub-system that is in turn mainly embedded in a single Electronic control Unit.

Process definition

Algorithm modeling is started from system department regarding definition of control laws and associated algorithms for the features. There are two different scenarios for algorithm design, one regarding building of a complete new feature and a second regarding adaptation of existing features. Related to these two scenarios, there are two main processes for algorithm design, first stage called "algorithm prototyping", and second one called "algorithm definition". A complete new feature design includes these two stages, while feature adaptation includes only "algorithm definition".

For the implementation, the models of control laws and algorithms are linked and subjected to code generation, which is managed by software department described in the section of "software development".

Plant models are managed by system department for providing relevant plant model concerning test required. They are used in early algorithm prototyping for pre-validation, and then before integration in the vehicle with vehicle plant model detailed in the section "system integration test".

Tool environment

System simulation is required for the engine plant model, for extension to the vehicle plant model, for restriction to specific physic modeling (typical injector model and combustion chamber). Models

are built within a powerful simulation environments (build with physical libraries like exhaust-gas library, hydraulics library, etc.) called AMESIM.

Software design of control algorithm for Model Based Development is build with MatLab environment.

Synchronous software designed for control law are based on MatLab/Simulink and StateFlow for state diagram definition. On the top of Simulink environment for detailed algorithm definition a proprietary solution has been developed to model architecture concern regarding operating system entities (synchronous and asynchronous mechanism), system event modeling for mode management and infrastructure (or basic software) interface modeling. This environment has the capability to manage composition of Simulink models in the context of feature (or function implementation).

On the other hand, asynchronous software build around state machine functionality which is typical for the Body electronics area uses the StateMate environment.

Both environments include simulation features and code generation not fully automated.

Algorithm prototyping

For completely new features, the first objective is a prototyping algorithm in order to fulfill the functional requirements. This must be made in a very reactive way in order to optimize time for the basic definition of algorithm control. Control engineers use the MatLab environment and basic Simulink features to validate features concepts using plant model required for features environment simulation. This could be restricted part of vehicle plant model, or dedicated model plant of sensor/actuator including physical continuous model if necessary. When basic algorithm is evaluated, rapid prototyping unit is used to validate the function into the real environment, means on board vehicle (or engine) test.

Algorithm definition

When the algorithm concept is developed, complete algorithm review takes place in consideration of configurable features, adaptation to standard platform architecture, and partitioning of features in a system. At this step extra requirements are added, mainly in the objective to consider reuse of the functionality. Model allocation in ECU and communication interface are refined and detailed at this step.

Algorithms captured in MatLab/Simulink are transferred in the proprietary compatible environment. Main change concerns allocation in different internal software component. Algorithm are also reviewed to be reusable (for example calibration lookup table extension), to be able to manage configuration (for example consideration cylinder number as a parameter of the algorithm), to integrate hardware platform interface, to describe software detailed architecture interfaces (Operation system, system event, infrastructure interface).

Then, composition of software element in a global feature is modeled to define a high level of abstraction, still considered reusable supporting variability mechanism at the top level of the modeling. This abstraction view of the feature is not fully modeled, for example variant configuration purpose is paper managed in addition to the executable model.

So far these composition of model is build, individual software elements are simulated in the proprietary Simulink-based environment, and then integration is performed at the upper abstraction model level. For this simulation no model plant is used.

Test case specifications are generated before the simulation step in order to define test plan of the features, elementary test case and integration test case.

At this stage specifications are generated from the model (paper or dedicated format of proprietary tool) and transferred to software department for software development.

This modeling environment does not have the full capability of software integration.

Software development

Out of the software model, application software code is generated semi-automatically or by hand, and linked with software platform dependant modules (basic software) for a final integration, and final binary generation.

Implementation aspects are more specialized towards the target architecture and hand made. Compositions of software elements are allocated in logical element of operating task by software designer in respect to code software analysis.

Software code elements are tested with unitary test via target code simulation. Then, integration is performed on bench with E/E simulated environment.

Software test do not use plant model for integration.

System integration test

Once software binary has been generated, ECU (hardware and binary software) and application software features inside are tested in front of a simulated vehicle plant model. This model includes communication network and sub-system features if necessary. This vehicle plant model is executed in real time on hardware in the loop system including closed loop model. It is still a model, but more specialized.

This stage is performed before installation in a vehicle.

Steps of model based development, summarized

To summarize, there are 11 steps:

1. Development of executable model of prototype controller
2. Validation of prototype controller model in a simulated environment
3. Validation of prototype controller model in a real environment
4. Enrichment/Refinement/Configuration of executable model of detailed controller
5. Validation of detailed controller model in a simulated environment
6. Code generation of application software corresponding to controller
7. Validation of code controller with simulated environment
8. Link of application software with infrastructure – Binary generation
9. Integration of applicative in real execution platform with simulated EE (open loop)
10. Integration of application software in real execution platform with simulated EE (closed loop) – Hardware in the loop
11. Integration of ECU in real EE architecture in vehicle

3 State of the art efforts related to the EAST-ADL

The increasing complexity of embedded systems, and lack of methods and techniques to support model based development, have led to a multitude of academic and industrial efforts. The research efforts span several academic disciplines (there are efforts originating from various disciplines such as computer science, software engineering, control engineering, etc.).

The purpose of this section is to give an overview of the types of efforts being pursued. Because of the multitude of efforts we first give a coarse characterization (grouping) of approaches and discuss them. We then focus the presentation on the efforts we find most strongly related to the ATESSST project. Finally, in Section 5, we discuss the relation of these efforts to the EAST-ADL and assess the added value of the EAST-ADL2.0.

3.1 Which efforts? Characterization of efforts and delimitation

In this overview of state of the art efforts we would like to mention four categories, those focusing on

- *Modeling languages for systems representation*
- *Platforms for model/tool integration and management*
- *Engineering guidelines/methods*
- *Industrial architecture standards for automotive embedded systems*

The first category covers languages such as the AADL, SysML and the UML, with an emphasis on approaches that address architecture level model based design, in particular those addressing system level analysis and design: safety, timing and functionality. Originating from different communities or domains, the examples given are quite heterogeneous both in focus and in nature; for example while the AADL represents an effort that was devoted to avionics applications, the UML was initially intended for general purpose software systems. In recent years, these two languages are being continuously evolved to meet the specific needs of modeling and design relating to automotive embedded systems. For example, more recently work is being undertaken to provide more capabilities for non-functional aspects by defining UML profiles.

Platform efforts striving towards model/tool integration have the aim to somehow bridge the gap between different types of models and tools, for example by developing transformations and exchange formats by which modeling formalisms can be exchanged between tools. Tool integration efforts provide techniques by which tools can cooperate, for example supporting co-simulation. Platform efforts focusing on model management can essentially be considered as database approaches, providing support for the definition of product information models and services for information access and usages like configuration management. Some of these platforms also support the integration of domain tools. Examples in this category include Tool-net and Generalstore.

The third category includes standard and guidelines that in some way prescribe how ATESSST-related products have to be developed. Examples in this category include relevant safety standards and best practices such as IEC61508 and Misra

The fourth category covers standardized architecture initiative potentially built on the top of previous categories with objective to federate domain area with open standard facilitating system element exchange and reuse (models, tools). The related architecture standard for middleware and exchange mechanism is AUTOSAR.

These four categories of approaches cover different aspects of engineering and are highly related to each-other. The modeling languages and platform approaches strongly complement each-other. The modeling language approaches typically provide very specific systems definition with respect to the type of products under consideration (compare with EAST-ADL and AADL). The platform

approaches, on the other hand, tend to be more generic in nature and can for example be applied to many kinds of products. For example, it is possible to use a modeling language such as the EAST-ADL as an input for defining the product information model of a platform. The third category provides advice on development practices, in particular for safety critical systems. Although this advice varies from general to detailed considerations, they do not provide that much details on how to employ models and analysis for safety critical embedded systems. More on this in the subsequent sections of this report.

Delimitation

In following survey we focus on modeling languages, guidelines/methods and a specific position of AUTOSAR initiative due to relevant domain. Platform environments for model integration and management are also related to ATESSST results, but only considered as prototyping material for method evaluation purpose. During case study work, tool environment will be evaluated mainly targeting methodological results, and a special focus with selected scenario will be set up for on demonstrator material. Expected results of the project are methodological bricks with plug-in based on a "simple" domain specific modeling environment and the UML profile of EAST-ADL. Graphicals and more generally platform environment is not the main core of the project.

Further delimitations are as follows: The Modelware project is not included in the survey [6]. This project aims at defining and developing the complete infrastructure required for the large-scale deployment of model-driven development strategies. Modelware does not consider safety critical embedded systems and is therefore of less interest to study in detail for ATESSST.

3.2 Details on selected research efforts

This section describes a selection of related projects, and their possible relation to ATESSST.

3.2.1 AADL and ASSERT

The Architecture and Analysis Description Language (AADL) [7] is a textual and graphical language supporting the model-based engineering of embedded real-time systems. The AADL was approved and published as an SAE Standard AS-5506 by SAE in Nov 2004. The AADL is based on the earlier MetaH effort by Honeywell research with an original basis in aerospace, avionics, military control system applications.

The AADL has its emphasis on detailed implementation architecture. There is no explicit support provided for higher levels such as functional modeling although it is claimed that this is also covered to some extent (through a refinement concept, e.g. the thread abstraction could initially refer to an "activity" and later on to a concrete platform thread). The AADL was developed to cater for modeling and analyzing non-functional properties, in particular for timing analysis (e.g. GRMA, timed automata), safety and reliability analysis (FTA, Fmea, Markov analysis), and hybrid systems modeling and extensions. Some features for this type of analysis were present already in MetaH. The AADL language is currently under further development, with work on an AADL programming language API, run-time system integration/mapping (ADA, Ada Ravenscar, C), an error model annex, UML alignment (UML profile for AADL) and an AADL behavior annex.

Further work on the AADL is pursued in the US (SEI/CMU, military, SAE) and in Europe by the ASSERT project [8]. The goal of ASSERT is to improve the system-and-software development process for critical, embedded real-time systems in the Aerospace and Transportation domains. ASSERT seeks to improve the Systems Engineering practice in this area, and addresses a proof-based approach. In addition, a reference architecture that can be reused and instantiated in critical applications is developed.

Relevance for ATESSST

The ATESSST scope is wider than AADL, which is more focussed on the implementation aspects. However, several areas, such as the concepts for handling safety and reliability in the AADL, are of interest when refining the EAST-ADL.

In particular, the AADL is not targeted to mass-produced systems and therefore has less emphasis on optimized solutions considering rather heavy run-time systems.

In contrast to ASSERT, ATESSST is aimed at an architecture description language that integrates several existing approaches, targets transportation only, hosts all engineering data relevant for the software development, and provides support for a wide range of verification and validation techniques.

3.2.2 AIDE

AIDE [9] is an integrated project in the eSafety area. The focus in AIDE is on system support to handle human behavioral aspects of new safety functions. AIDE applications represent the kind of complex, safety-related systems that require rigorous development support to manage their complexity and achieve correctness and safety.

AIDE is decomposed into 4 sub-projects:

- Sub-Project 1 (SP1): Behavioral Effects and Driver-Vehicle-Environment Modeling

In this sub-project, aspects on models of the driver, the vehicle and the environment are elaborated. A set of requirements on these models are defined, and actual models mimicking the Driver, the Vehicle and the Environment are derived.

- Sub-Project 2 (SP2): Evaluation and Assessment Methodology

The goal of this subproject is to develop a cost efficient and industrially applicable methodology for quantifying behavioral effects of IVIS (In-Vehicle Information & Communication System) and ADAS (Advanced Driver Assistance System) functions, and their relation to road safety. Existing methods for the assessment is collected and reviewed, and a refined approach is presented. A prototype evaluation tool is also developed.

- Sub-Project 3 (SP3): Design and Development of and Adaptive Integrated Driver-vehicle Interface

As the name reveals, this SP will elaborate on the driver-vehicle interface. As opposed to SP1 and SP2 it concerns the actual in-vehicle systems. The deliverables cover scenarios and use cases; Requirements and specifications; and System Architecture for the human-machine interfaces. In addition, Driver-Vehicle-Environment monitoring modules for the vehicle will be designed and developed. In addition, nomadic devices (PDA:s and other connected devices for e.g. entertainment and information) are investigated.

- Sub-Project 4 (SP4): Horizontal Activities

This SP deals with project-wide activities such as standardization and dissemination.

Relevance for ATESSST

Interaction between ATESSST and AIDE is expected through partners active in both projects. AIDE interaction is useful for identification of the engineer's needs and case study definition.

For example, the environment models in SP1 may serve as a reference for the plant models that need to be modelable using EAST ADL2.

SP2 is mainly about assessing the vehicle systems, and therefore has limited impact on ATESSST. It may be seen as an input to V&V methods modeling, but because it is an advanced and rather specific area of analysis, it is not in the immediate scope.

The vehicle systems addressed in SP3 should be possible to model in ATESSST. Because the scope of AIDE is the human-machine interface aspects only, it remains to be seen if these applications are the most representative for automotive eSafety systems.

As AIDE results are partly confidential, it has to be investigated on a case-by-case basis, how much can be used in ATESSST.

3.2.3 AUTOSAR - Automotive Open System Architecture

AUTOSAR [1] is a partnership that regroups all the European automobile constructors. Its objective is the establishment of an open standard for automotive E/E architecture. It will serve as a basic infrastructure for the management of functions within both future applications and standard software modules. The goals include modularity, scalability, transferability and re-usability considerations. ATESSST results will guide the rigorous development of eSafety systems based on AUTOSAR E/E-Architectures considering expertise acquired by the project partners being also member of AUTOSAR consortium.

Relevance for ATESSST

Autosar only focuses on the lower levels of implementation. In addition, Autosar has a major emphasis on non-functional issues such as portability and software modularity, and less on qualities such as performance and reliability. There are close connections between Autosar and ATESSST, both are addressing automotive systems and have an overlap between the partners. Harmonization is being discussed among the projects.

3.2.4 EASIS

The goal of the EASIS project [10] is to define and develop a powerful and highly dependable in-vehicle electronic architecture and appropriate development support. These aspects are considered non-competitive for OEMs and suppliers, and therefore suitable for collaborative efforts.

The objectives, in more detail, are:

- Modular scalable E/E-architecture for active, passive and integrated safety systems
- Standardized signal and functional interfaces for environment detection systems, telematics, powertrain, chassis and HMI
- Embedded system safety analysis
- Provision of high availability and safety - Move from fail-silent to fail-operational system behavior
- Provision of introduction plan for new concepts into existing automotive system architectures
- Preparation for standardization

The project includes the following main work packages:

WP 0: Integrated Safety Requirements. This WP collects, together with the predecessor project VEESA, requirements for the rest of the project.

WP 1: Software Architecture. This WP delivers requirements, concepts and designs for a software platform for Integrated Safety Systems, including required safety services.

WP 2: Hardware Architecture. This WP delivers requirements, design guidelines, concepts and specification of the hardware architecture. In addition, a simulation environment and a hardware prototype will be produced.

WP 3: System Dependability. This WP concerns Hazard identification and classification, Design principles for safety-critical components and systems, Validation and verification and Safety case construction. The WP will deliver assessments of existing approaches in these areas as well as guidelines for their application.

WP 4: Processes and Tools. In this WP an EASIS Engineering Process will be defined to address the emerging needs from Integrated Safety Systems. Special attention is given to early phases of the development process, and a tool chain recommendation based on the EAST ADL framework will be given.

Relevance for ATESSST

The EASIS project will deliver state-of-the-art regarding the architecture for integrated safety systems, as well as for the related safety processes and development processes – EASIS is finishing by the end of 2006. For ATESSST, several of the results will be relevant: The software and hardware architectural solutions should be possible to model. Also, the development process and the identified safety analyses and methods need to be addressed in ATESSST. Since EASIS results are partly confidential, it is being investigated how much can be used broadly in ATESSST. Regardless of this, the EASIS partners of ATESSST will be in a position to ensure that the integrated safety systems on processes and tools will be considered in the project.

3.2.5 Families

This section describes a selection of related projects, and their possible relation to ATESSST.

FAMILIES [11] is an ITEA project dealing with ‘system family’ development. The Families project proposed results supporting the separation of the domain aspects, the technical aspects (quality of services) and the technological aspects (platforms), in consistent models, in the MDA standardization frame. Amongst other things, they have defined a common meta-model for variability management in order to achieve this goal.

The main relevant task for ATESSST within Families is the 4.1 task with two tracks:

- Domain and application modeling practices (Task 4.1)
- Guidelines for Variability Modeling (Task4.1 Track 2)

This task was focused on model assets definition for product families in an MDA context. The main objective was to capitalize on ESAPS and CAFE results and to make a standardization proposal whenever it’s possible.

The problem of modeling variability both in space (variability in a product line's current product offering) and time (differences that arise in the product line over time due to evolution) has been studied quite thoroughly. However, there are still many individual approaches. There is no agreed upon conceptual model of variability nor notations for capturing variability that can be applied consistently to assets at different levels of abstraction and phases. The approaches typically come from different backgrounds or have different base assumptions or constraints. Agreeing upon a common conceptual model and standardizing this is essential so that variability modeling can be taken into common practice and so that tool vendors can build in support for variability.

The goal of this task was to consolidate the results from ESAPS and CAFE and to systematically compare and evaluate existing conceptual models and notations for capturing variability as a first step towards standardization.

The resulting conceptual model of variability which was obtained in the project captures the elements related to variability (e.g., variation points, decisions, constraints...). The model also captures the properties of the elements and relations among these elements. Finally, this model relates the variability elements to “regular” single system model elements (e.g., requirements, features, classes, associations).

Along with the conceptual model, agreed upon notations for capturing the variability information have been proposed based on the evaluation of existing approaches. To aid understanding, adoption and use, the consolidated conceptual model and notations have been illustrated by the agreed upon digital watch example. The resulting conceptual model and notations may not be completely general and suited for all product line situations. The results have been evaluated and the advantages and disadvantages have been discussed along with any assumptions or constraints where the results may best be applied.

The Families conceptual model of variability was presented at an OMG meeting. It received positive feedback, which might lead to a request for interest in the future.

Relevance for ATESSST

Reuse of results from FAMILIES in ATESSST WP4 is expected, the conceptual model of variability will have to be examined with respect to ATESSST specific needs. The work done along the Families project by CEA is also relevant: it dealt with prototyping tool based on a UML2 profile built on the conceptual model. It added analysis, product derivation and decision model generation based on a system family model.

3.2.6 IEC 61508 and the forthcoming ISO 26262

IEC 61508 is a standard from 1998 concerning functional safety of electrical/electronic/programmable electronic (E/E/PE) safety-related systems. The standard is generic, i.e. it is not aimed for a specific industrial branch. However, this standard enables application sector international standards dealing with safety to be developed. For the time being, there is such an initiative in the development of the automotive safety standard ISO 26262 (see further below). The intention of IEC61508 is to cover all lifecycle phases as for example from initial concept, through design, implementation, operation and maintenance to decommissioning.

On the basis of a particular viewpoint on the structure of a safety critical system, IEC 61508 assumes a conception view of how to describe a system, and hence also how to assure its safety. See also Figure 4.

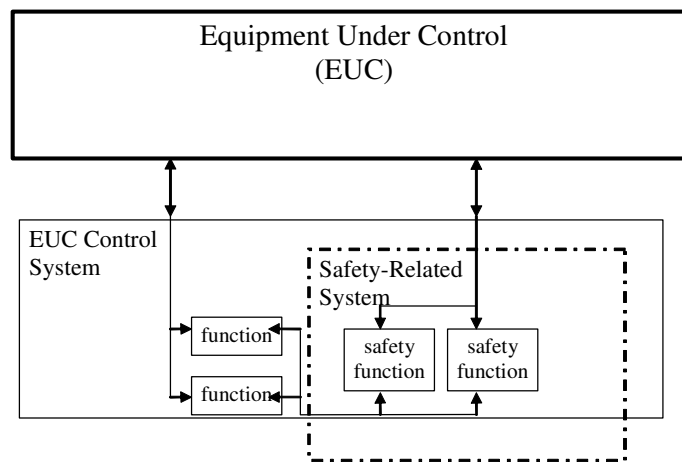


Figure 4. The conception view of safety critical systems adopted in IEC 61508

Key concepts in this conception view include

- Equipment Under Control (EUC) - Equipment, machinery, apparatus or plant used for manufacturing, process, transportation, medical or other activities.

- EUC Control System - System which responds to input signals from the process and/or from an operator and generates output signals causing the EUC to operate in the desired manner.
- Safety-Related System - Designated system that both implements the required safety functions necessary to achieve or maintain a safe state for the EUC; and is intended to achieve the necessary safety integrity for the required safety functions. The required safety integrity level may be achieved by implementing the safety functions in the EUC control system or by separate and independent systems dedicated to safety.
- Safety Function - Function to be implemented by an E/E/PE safety-related system, other technology safety-related system or external risk reduction facilities, which are intended to achieve or maintain a safe state for the EUC, in respect of a specific hazardous event.

To assure a safe system, it is critical that the safety functions are satisfactorily performing under all the stated conditions within the stated period of time. The probability for this is called Safety Integrity. IEC 61508 uses discrete levels of safety integrity, Safety Integrity Levels (SIL), for specifying the safety integrity requirements of the safety functions to be allocated to the E/E/PE safety-related systems. The standard adopts a risk-based approach for the determination of the SIL requirements, and sets numerical target failure measures for E/E/PE safety-related systems which are linked to the SILs. In this context “risk” is the combination of probability of occurrence of harm and the severity of that harm.

ISO 26262 is a not yet released standard that is aimed to be the future safety standard for road vehicles. So far there has not been any dedicated automotive standard in the field, but the MISRA Development Guidelines for Vehicle-Based Software (ISO TR 15497). When starting this ISO 26262 project of defining a dedicated automotive safety standard, there were a number of issues where the IEC 61508 was found to be not enough for the automotive industry, e.g.:

- Supplier relations are not described in the IEC 61508 at all, thus a dedicated automotive standard needs to include requirements on manufacture/supplier relation and distributed development processes.
- Design and testing of embedded automotive systems is not treated detailed enough in IEC 61508. A dedicated automotive safety standard must for example consider application of hardware-in-the-loop tests, lab cars, fleet tests, and user oriented tests during validation.
- In IEC 61508, the safety life cycle is not designed for consumer goods but originated from process and automation industry, e.g. validation after installation etc. A dedicated automotive standard needs an adaptation of the safety life cycle to typical automotive development and operation phases.
- Electronic automotive components typically have quite short availability cycles. Probabilistic data are therefore often not available at the time of a safety proof.
- A dedicated automotive standard should support the implementation of development processes and safety assessments (e.g. Automotive SPICE).
- Most requirements are not SIL-dependent. Tailoring of the requirements for projects and different development groups in a company is difficult due to that.

Relevance for ATESST

For IEC61508, the above mentioned and other problems are intended to be solved during the ongoing standardization work that has about the same timing schedule as the ATESST project, i.e. with a deadline at the end of 2007.

The timing of this development is thus very suitable for information exchange among the standardization effort and ATESSST. The ATESSST project has good connections to the ISO 26262 effort through Rolf Johansson (Mentor) who is part of the mirror/corresponding working group in Sweden involved in the standardization work, and will be in a position to influence the ISO standardization effort with regards to model based development practices.

3.2.7 MISRA

The Motor Industry Software Reliability Association – MISRA - is a UK based section of the Motor Industry Research Association (MIRA). With the contributions of mainly UK-based automotive manufacturers, suppliers and research institutes it started in the mid 1990s to produce the “Development Guidelines for Vehicle Based Software” [12], and the more widely known (and recognized) “MISRA-C-Guidelines for the use of the C language in critical systems” [13]. These documents are carefully maintained and kept up-to date. The effort to collect and describe the information was to a part funded by the European Union.

An initiative to extend the scope of MISRA-C to the C++ language is underway.

The “Development Guidelines” is an overview paper summarizing the recommendations, which are described in detail with their respective rationales in accompanying eight reports:

Report 1 Diagnostics and Integrated Vehicle Systems: This report covers the software part of the engineering of vehicle communication and diagnostics networks. It addresses the overall (electric/electronic) vehicle architecture, communications and data-transfer on busses , on-board and off-board diagnostics, as well as tools and testing equipment.

Report 2 Integrity: This report has a strong affinity to the 61508, which at time, when the report was written, was under standardisation as DIN 1508. The summary states, that

“In order to ensure the safe operation of safety-related programmable systems, it is necessary to recognise the various possible causes of failures, and to ensure that adequate precautions are taken against each one. As part of a safety analysis controllability categories are assigned to each hazard; this defines the Integrity Level necessary for the design of the system associated with that hazard. ... It gives guidance on how the different levels of confidence can be achieved through the use of appropriate design, development and quality management processes during the system life-cycle. It also recommends that some form of independent assessment should be performed to confirm that the necessary degree of confidence has indeed been obtained, in particular for the higher Integrity Levels. The overall aim is to provide a degree of confidence in the final system, which corresponds to the Integrity Level, that the risks associated with it are indeed at an acceptable level.”

MISRA sets a different perspective on controllability than is used in 61508. The same (automotive) view will probably be found in the upcoming 26262 standard.

Report 3 Noise EMC and Realtime: Electromagnetic compatibility is a means to prevent external interferences entering and disturbing on board ECUs at the same time avoiding unacceptable emission from these ECUs. The report bases on the principle, that “... if electromagnetic interference cannot enter the critical parts of the system, it cannot cause malfunction”. This leads to strict requirements for hardware measures as a defence. The report goes on “... it should (not) be assumed that hardware alone is sufficient protection; rather, it should be assumed that some disturbing signals or corrupted data will enter the system, which will require a defensive approach to software programming.” The report concludes with many well tried “recipes” on what to do and to avoid when building a critical embedded control system.

Report 4 Software in Control Systems: This report can be characterized by a citation from its summary: “It is recommended that textbooks on control are read for detailed discussion of the theory. A recommended reference list is appended. Theory is split into linear and nonlinear control.

The former is the simpler, and most readily understood.” and “... no particular recommendation is made regarding methods to be used for calculating stability”.

Report 5 Software-Metrics: The report surveys and recommends a number of software attributes and metrics to be used to estimate the quality of software. Its summary continues “At present the majority of automotive software is developed in assembler for reasons of speed and size. However, there has been some movement towards the use of high level languages. The software orientated metrics presented in this document have been selected due to their applicability (in most cases) to both low level and high level languages. The report also deals with collection and interpretation of metric results, with an emphasis on interpretation of results and how data can be fed back into the development process in order to improve it. ... it is intended that the document should be a reference document to aid in the determination of appropriate metrics to be used on software projects.”

Report 6 Verification and Validation: This report presents the verification and validation activities that are considered necessary for sub-systems, especially the software sub-systems, and upon the vehicle as a whole.

These verification and validation activities are presented in the order that they would normally be applied when adhering to a typical life-cycle.”

The important fact that quality cannot be designed-in as an after thought is emphasized, and so early consideration of the issues highlighted in this report is essential.

The depth and rigor of the verification and validation activities will depend upon the integrity level of the system (see MISRA Integrity Report). It is therefore prudent to identify the required integrity level as soon as the system is conceived in order to avoid unnecessary costs. Final acceptance is based upon information concerning the system. The integrity level of the system will affect the nature and the volume of the required documentation, i.e. from formal reports, at the highest level, down to notes and comments, at the lowest level.

Report 7 Subcontracting of Automated Systems: “This report ... is intended to be an engineer’s view of the topics which should be considered by engineers, managers, project managers and purchasing departments involved with software products---buying, selling, creating, managing.”

Report 8 Human Factors in Software Engineering: “Ergonomics, or Human Factors, refers to designing for human use ... it seeks to change the things people use and the environment within which they use these things to better match the capabilities, limitations and needs of people.” “The advent of computers and microelectronics has resulted in an increase in information for an individual to take in and comprehend.” “As with any system safety analysis, it is essential that Human Factor Engineering considerations are addressed in the early design phases as rectifying human factor incompatibilities at a later stage is costly and often arduous.”

The MISRA-C guidelines describe rules, how to use the C language in critical environments. These rules intend to restrict the programming language to a “safe subset” and encourage a programming style to avoid constructs and habits, which have been proven error-prone or which are expected to be ambiguous or produce unexpected program behaviour. Most of these rules can be statically checked by specialised tools (often called MISRA-C-checker).

The value of following the MISRA-C rules is most prominent, when coding C by hand. Modern code generators tend to have the rules “built-in” or at least provide a generation option for MISRA-C-code, thus reducing the fault-finding efficiency of the MISRA-C-checker.

Relevance for ATESSST

The MISRA reports cover many topics, some of which are worth further consideration in the ATESSST project. Here are some recommendations:

- Report 1 Chapter 2 (on Architecture) could be worth a closer look
- Report 1 Chapter 3 (on Networking) could be used to check ADLs modelling

- “Off-Board Diagnosis” could be used as a scenario, where different systems – considering ADL descriptions – are connected and must work together.
- “on-Board diagnosis” poses the question, how one would model (using the ADL) the limp-home behaviour required by almost all critical ECUs
- Report 3 poses question about details: timing constraints (for which the ADL does provide something), processor interrupt and their handling (which should be considered in the ADL), real number and floating point arithmetic (and the trouble with precision, underflow, ...) (related to behavior descriptions and exceptions – error detection and handling) and processor specifics and watchdogs.
- Report 6 (on V&V) mainly describes the V&V processes (which should be reflected at least in part in the ADL)
- Report 4 contains a section on software safe states, redundancy and diversity and two sections on the intricacies of dealing with real hardware, data acquisition and sensor variability. These sections might also be useful when checking the expressiveness of the ADL
- Report 7 on subcontracting software sets the theme, when ADL is used in a supplier/OEM context. That material could probably be of help, when discussing the issue of preserving the intellectual property of each partner and still get working system of sufficient quality.
- Report 2 is on safety issues (mentioning different analyses); it is at a quite general level, so it is questionable whether it can provide any hints for ATESSST.
- Report 8 treats an important topics but is not technical enough to be useful for inputs.

3.2.8 OMG related work

OMG related standards have a strong momentum, which ATESSST will exploit. While UML1 and UML2 as such are not directly useful for safety-critical systems, there are several OMG developments that try to address such issues, thus complementing the UML2 norm. For example, the SPT specifies a UML profile that defines standard paradigms for the modeling of time-, scheduling-, and performance-related aspects of real-time systems. This standard has yet to be aligned with UML2.0. A current OMG request for proposal – MARTE, [14] - is addressing this, in order to define a new UML profile for real-time embedded systems, and is also enriching the SPT with QoS & Fault Tolerance consideration. This work is done within the CARROLL research project PROTES (see below). SysML is a visual modelling language for systems engineering applications, [15]. SysML supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. It is implemented as a UML2 profile, adding some diagrams and constructs to the UML2 (mainly as regard requirements and parametric associations). These systems may include hardware, software, information, processes, personnel and facilities.

Relevance for ATESSST

All these standardization initiatives are highly relevant for ATESSST. From now on, one cannot pretend to issue a profile for some domain without taking into account the existing profiles, complementing and/or aligning with them. It is expected that EAST-ADL2 will in the end refine some of the constructs present in broader scope profiles, such as MARTE or SysML. Yet the EAST-ADL domain model should be devised in a self-contained manner.

Another issue is to show with the evaluation of the EAST-ADL2 language in WP6 that design methods are eased with the use of the profile. Profiles or domain models provide only for an agreed, standardized way of using modeling constructs within a particular domain, they in general do not provide for an engineering method description, which is as much important if not more.

3.2.9 PROTES

PROTES – this research project is part of the CARROLL program [16]. It was launched by CEA-List, the DaRT, ESPRESSO, OSTRE and TICK teams of the INRIA, in collaboration with THALES Communications France, THALES Airborne Systems, THALES Underwater Systems, and the MIRROR Pilot Program. It deals with the topic of model-driven engineering for embedded, real-time systems. This project aims at standardizing a UML 2.0 profile for real-time embedded systems at the OMG.

Relevance for ATESST

ATESST will use and influence the results of PROTES and the OMG standardization of MARTE via CEA which is part of this project and which is both leading the writing of the proposal and also coordinating the working task at OMG level. So far (end of March) results are not publicly available but will be as soon as the profile is released to the OMG. The tool chain developed within ATESST is taking into account the opportunity to re-use the implementation of the MARTE profile (support of MagicDraw and RSA).

3.2.10 RESPONSE/PREVENT

PReVENT[17] is an integrated project on preventive safety. The PReVENT project investigates vehicle systems that increase safety through various chassis, powertrain and driver warning functions. PReVENT applications are complex, safety-critical and have stringent performance and correctness requirements.

Of particular interest within PReVENT is the work within the sub-project RESPONSE which is described in the following in more detail [18].

RESPONSE 3 is elaborating a European Code of Practice for an accelerated market introduction of Advanced Driver Assistance Systems (ADAS). The Code of Practice will contain descriptions of procedures for analysis of user requirements, definition of systems and validation procedures. It will have a special focus on an integrated “system safety - human factors” approach and give advice on the evaluation of driver behaviour and expectations. It will be a key instrument to help engineers develop safe products, industry to control their risks and the authorities to understand and follow the development and testing processes.

In ADAS it is necessary to manage not only the problem of specification, implementation or realization errors and failures but also to prevent [unwanted consequences] from reasonably foreseeable operational errors and reasonably foreseeable misuse.

This implies to establish “principles” for the development and evaluation of ADAS on a voluntary basis, as a result of a common agreement between all involved partners and stakeholders, mainly initiated by ADAS manufacturers.

The RESPONSE 2 project has been the first move towards agreed validation procedures to be incorporated in a Code of Practice. Further methodological developments for validation (of system safety and safety of usage) and the final definition of the Code of Practice are part of the current ongoing project RESPONSE 3,

The Code of Practice will

- subsequently be translated into a specification framework
- describe methods how to perform risk analysis
- help finding the possible methods to clarify these risks
- consider state of the art for ADAS with respect to safety issues
- argue the reasonable safety and duty of care within the development of ADAS

The state-of-the-art in automotive safety systems for ADAS is characterized as follows [17]: “Today in general there is no risk scale for possible ADAS malfunctions which can be regarded acceptable by an OEM. Moreover, the OEM cannot refer to a guideline that shows the implications for the development and validation process. If a specific risk has been identified for a specific application, the OEM has no approved guideline to deal with this situation. In particular, all ADAS with interaction of driver and driver tasks are difficult to assess with respect to safe function in each and every driving situation.”

RESPONSE3 uses a Three-Level-Classification of ADAS for their legal analysis:

Information and warning systems: The System only provides information to the driver. The driver remains in full control of the vehicle at all times. Visible and audible information is provided to the driver in order to assist careful and proper driving. The driver is therefore responsible for exercising due care when driving. The information provided by the system may be incorrect or inaccurate. If this is the case, manufacturer or distributor liability should also be taken into consideration.

Overridable intervention systems: (Intervention systems which the driver may override at any time) Potential driver liability concerning the overridable intervention system will very much depend on the individual case. Important factors will include inherent system limitations and driver perception of warnings and limitation consequences as well as reasonable driver behaviour like overriding the system in order to avoid critical traffic situations. System malfunction may also lead to liability of the manufacturer.

Non-overridable intervention systems: If the driver is not able to override the system due to the ADAS design (e.g. due to his/her reaction time), liability due to inappropriate driver behaviour caused by the system arises, because such liability requires that the driver is not able to influence the system. If the driver cannot override the driving support system manufacturer liability is more likely. If the driver cannot override the system, the manufacturer is confronted with a liability case.

Relevance for ATESSST

Because of the application characteristics, the PReVENT applications are highly relevant as input for ATESSST case studies. PReVENT has ATESSST partners in the consortium and the potential use of PReVENT results is being investigated.

From the above summary of the project, the following points for discussion regarding usage of the EAST-ADL can be derived:

Driver-in-the-loop: “A principle to avoid critical situations of the ACC (automatic cruise control) was the layout to give the driver always enough time for adequate reaction, when the system limits are reached. This will limit the possible range of the system, but will also assure that the driver will experience the limits often enough that he will stay in the loop.”

- Should the EAST-ADL be able to describe a system model, where “driver will ... experience ... limits ... (to) ... stay in the loop.”? Wouldn't this require to model driver behaviour, too?

For ATESSST this is a very relevant question. The driver can be said to belong to the ATESSST environment model, and suitable behavioral models will be investigated in this context.

System safety process: The automotive industry has started to standardize such a safety process of complex electronic systems for safety applications (the ISO work).

- Should ATESSST respond to these activities? How?

For ATESSST this is a good opportunity to interact with the standardization work and influence it with respect to model based system development which so far has not been treated thoroughly by standards.

Support for risk assessment: “As there is no standard risk assessment at the moment available, the automotive industry is currently proposing a methodology for risk assessment (risk being

expressed by a combination of severity for a specific situation outcome, the probability to be in that situation and the controllability of the driver to avoid a critical outcome).”

- Should the ATESSST project respond to the risk assessment activities? How?

This question is somewhat similar to the previous one but at a more methodological level. The implications, if any, on the EAST-ADL will be studied.

Safety requirements: “Basically, it is important to consider the risk of malfunctions ... safety requirements have to be derived.”

- Should “safety requirements” be represented in the ADL? How?

This is a clear aim of the EAST-ADL to be treated in the ATESSST project.

Safety functions: “Besides the principles of designing the positive functionality of systems ... complex systems have also to cope with system failures ...”

- Should the EAST-ADL be able to describe, what (dangerous) system faults are foreseen and how the architecture will handle their consequences (in a safe way)?

This question is also strongly related to the previous ones. The EAST-ADL has as a clear aim to support representation of safety related information and analysis of safety related behaviors so this issue will be studied in the ATESSST project.

4 Added value of the EAST-ADL2.0 with respect to related (research&industrial) efforts

For the safety related standards and guidelines, IEC61508, ISO 26262 and MISRA, it is concluded that these, in general, lack of specifics when it comes to model based specifications, analysis and synthesis. ATESSST can provide inputs to and influence the ongoing work on the ISO26262 with respect to model based development. The timing for this is appropriate since the ISO effort and ATESSST run in parallel. As pointed out in section 3.2.7, it is also concluded that some work within the MISRA guidelines should be further studied to assess its relevance and impact for ATESSST. Some of these issues are also raised in the projects AIDE, PREVENT/RESPONSE and EASIS, were issue deals with humans in the loop, how to model the humans (mainly the driver) in the embedded systems environment and incorporate them into systems analysis.

The relation between the more closely related language development efforts, AADL, SysML and MARTE, and the EAST-ADL are discussed in the following. In addition to language Autosar standardization will also be considered for alignment on conceptual elements from EAST-ADL and discussed below. For each approach we briefly summarize and discuss

- Scope
- Types of abstractions, levels of abstraction and refinement
- Missing/weak support
- Added values of the EAST-ADL in comparison to the related efforts.
- Potential inputs and further investigation by the approach

4.1 AADL

Scope:

- Avionics background. Software and hardware integration

Missing/weak support:

- Support for higher abstraction levels (reqs, functions, variability) and environment modeling

Missing/weak support:

- Specifics for automotive systems such as networks.

Added value:

- EAST-ADL2.0 has the aim to provide this support including methodological support.

Potential inputs and further investigation:

- Inputs: Modeling and component concept, attributes and analysis of low level abstractions
- AADL support for networks?
- AADL support and ongoing work on reliability modeling and analysis
- AADL alignment with the OMG

4.2 Autosar

Scope:

- Automotive middleware, portability, platform independent SW, SW component and platform modeling, tools

Missing/weak support:

- Support for higher abstraction levels (reqs, functions, variability), environment modeling

Missing/weak support:

- Weak on quality attributes such as timing for behavior modeling and reliability.
- Weak on variability management and product line.

Added value:

- EAST-ADL2.0 aims to provide this.

Potential inputs and further investigation:

- Modeling interfaces standardization, attributes and analysis of low level abstractions
- Alignment, as far as possible, is important since both Autosar and ATESSST target the automotive industry.

4.3 MARTE

Scope:

- A UML2 profile for defining timing, quality of service and fault-tolerance concepts.

Missing/weak support:

- MARTE does not support the overall modeling according to automotive needs – it focuses on all kinds of real-time software systems.

Missing/weak support:

- MARTE suffers from the same weaknesses when it comes to automotive embedded systems as does the UML2; generic modeling of software systems. The decomposition of the system into abstraction levels and aspects is not part of MARTE”.

Added value:

- EAST-ADL2.0 aims to provide tailored support for automotive embedded systems.

Potential inputs and further investigation:

- Useful constructs for e.g. timing should be reused as far as possible.

4.4 SysML

Scope:

- A generic systems engineering language – broad scope for “all” domains.
- Intended for systems engineering information with an emphasis on higher abstraction levels
- SysML proposes to use specializations for different domains (mentioning UML2 as a specialization for SW).

Missing/weak support:

- SysML suffers from the same problem as UML2 for implementation and “design” – where it is more geared towards generic systems and does not provide explicit semantics of models.
- SysML has no alignment towards Autosar.
- SysML does not provide explicit support for variability.

- SysML does not have the same explicit abstraction levels as the EAST-ADL.

Added value:

- The EAST-ADL can be seen as a further specialization for automotive needs providing explicit abstraction levels and aligning with Autosar.
- The EAST-ADL complements SysML and provides ontology for automotive systems, with specific models/semantics towards automotive system requirements such as variability.

Potential inputs and further investigation:

- Constructs for higher level abstractions, in particular requirements and parametric diagrams should be investigated.

5 Contribution to overall ATESST objectives

This report has had the aim to provide an overview of industrial practices and related research/industrial efforts in the context of the ATESST project.

Within the project as a whole the investigation of the most related state of the art efforts will continue. Several of the topics treated will be further studied in the continuation of the project in the development of the EAST-ADL. Many of the approaches mentioned in this report map directly to one or more work-packages.

The relation of the EAST-ADL to the Autosar and MARTE standardization efforts is further elaborated and discussed in the ATESST deliverable D7.2.1. The relation is complicated by the fact that these efforts are closed, but facilitated for ATESST since some of the partners are partners in these standardization efforts as follows:

- Autosar: DC and SV are core members, ETAS, VTEC and MGH are premium members. It is estimated that Autosar deliverables will be delayed at least until the end of 2006.
- CEA is a core member of the MARTE standardization work. The potential for attaching EAST-ADL to MARTE is being explored including investigation of technical issues, releases and overlap. A first open MARTE deliverable expected by the summer.

The corresponding partners of the ATESST consortium will thus continue their interactions with these efforts to exploit alignment and synergy.

6 Conclusions

The presentation in the report is based on the collected experiences of the consortium and our understanding of available/published information.

As one further step to validate the contents of the report, it is our intention to send the report to those responsible for the related approaches – providing an opportunity for feedback and to correct any potential mis-interpretations on our side. This will provide another iteration of this report.

7 References

- [1] *Autosar: Webpage reference*. March 30, 2006 [Online]. Available <<http://www.autosar.org/find02.php>>
- [2] U. Huber and U. Näher 2004. *Failure-free electronics – seven levers for optimized electronics R&D*. McKinsey&Company, Automotive & Assembly. 2004.
- [3] ARTIST2 - Network of Excellence on Embedded Systems Design, *Roadmap for Embedded Software and Systems*. March 30, 2006 [Online]. Available <<http://www.artist-embedded.org/FP6/ARTIST2Events/Publications/Roadmap/>>
- [4] M. Törngren and O. Larses. *Characterization of model based development of embedded control systems from a mechatronic perspective - drivers, processes, technology and their maturity*. Technical Report. TRITA-MMK 2004:23. ISSN 1400-1179. ISRN/KTH/MMK/R-04/23-SE
- [5] R. Jeutter, and B. Heppner 2004. Model-Based System Development – Is it the Solution to Control the Expanding System Complexity in the Vehicle? *SAE World Congress 2004*. Detroit, MI. March 8-11. 2004. SAE 2004-01-0300.
- [6] Modelware Project. March 30, 2006 [Online]. Available <<http://www.modelware-ist.org/>>
- [7] *Architecture and Analysis Description Language – AADL: SAE standard AS5506*, issued Nov. 2004. Available also at <www.aadl.info> and <http://www.sae.org/servlets/productDetail?PROD_TYP=STD&PROD_CD=AS5506>
- [8] ASSERT - Automated proof based System and Software Engineering for Real-Time. March 30, 2006 [Online]. Available <<http://www.assert-online.net>>
- [9] AIDA - Adaptive Integrated Driver-vehicle InterfacE. March 30, 2006 [Online]. Available <<http://www.aide-eu.org>>
- [10] EASIS- Electronic Architecture and System Engineering for Integrated Safety Systems. March 30, 2006 [Online]. Available <<http://www.easis.org>>
- [11] FAMILIES Project. March 30, 2006 [Online]. Available <<http://www.esi.es/en/Projects/Families/>>
- [12] *Development Guidelines for Vehicle Based Software*. ISBN 0952 4156 07.
- [13] *MISRA-C-Guidelines for the use of the C language in critical systems*. ISBN 0952 4156 4X.
- [14] *Request for Proposal of a (UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) RFP) / Document 2005-02-06*. March 30, 2006 [Online]. Available <<http://www.omg.org/cgi-bin/doc?realtime/2005-2-6>>
- [15] *SysML Partners, Systems Modeling Language (SysML) Specification (ad/05-01-03)*. March 30, 2006 [Online]. Available <<http://www.omg.org/docs/ad/05-01-03.pdf>>.
- [16] PROTES Project. March 30, 2006 [Online]. Available <<http://www.carroll-research.org/uk/projets/projets.htm>>
- [17] PReVENT Project. March 30, 2006 [Online]. Available <<http://www.prevent-ip.org/>>
- [18] RESPONSE 3 – Code of Practice for Development, Validation and Market Introduction of ADAS – A PReVENT Project. March 30, 2006 [Online]. Available <<http://www.prevent-ip.org/download/Events/20050601 ITS Hannover papers/25043.pdf>>.