11 The EAST-ADL Architecture Description Language for Automotive Embedded Software

Philippe Cuenot[1], Patrick Frey[2], Rolf Johansson[3], Henrik Lönn[4],
Yiannis Papadopoulos[5], Mark-Oliver Reiser[6], Anders Sandberg[7],
David Servat[8], Ramin Tavakoli Kolagari[4],
Martin Törngren[9], and Matthias Weber[10]

[1] Continental Automotive
[2] ETAS
[3] Mentor Graphics
[4] Volvo Technology
[5] University of Hull
[6] TU Berlin
[7] Mecel
[8] CEA LIST
[9] KTH
[10] Carmeq

Abstract. Current trends in automotive embedded systems focus on how to manage the increasing software content, with a strong emphasis on standardization of the embedded software structure. The management of engineering information remains a critical challenge in order to support development and other stages of the life-cycle. System modelling based on an Architecture Description Language (ADL) is a way to keep these assets within one information structure. This paper presents the EAST-ADL2 modelling language, developed in the ITEA EAST-EEA project and further enhanced in the ATESST project (www.atesst.org). EAST-ADL2 supports comprehensive model-based development of embedded systems and provides dedicated constructs to facilitate variability and product line management, requirements engineering, representation of functional as well as software/hardware solutions, and timing and safety analysis.

11.1 Introduction

Current trends in automotive software development focus to a large extent on how to manage the increasing software content. Hybrid vehicle control, active safety systems, diagnostics services, etc., all rely on embedded systems. The automotive industry faces the challenge of incorporating software and embedded systems engineering within traditional mechanical engineering enterprises. This challenge is addressed in many ways, including incorporation of new processes, tools, and the standardization of the embedded software structure. Software standardization is addressed in the AUTOSAR standardization initiative [1]. The AUTOSAR standard specifies how to model the software architecture and

H. Giese et al. (Eds.): MBEERTS, LNCS 6100, pp. 297–307, 2010.

final implementation, but the requirements, functional content realized by this solution, and non-functional aspects such as support for safety analysis, are not covered [2].

EAST-ADL2 is an Architecture Description Language. As such it provides a basis for documenting and managing the various artefacts of an advanced embedded system (requirements, features, desired behaviours, software and hardware components), and their dependencies (refinement, allocation, composition, communication, etc.). Any modelling language is directed by the product aspects and process stages it intends to support. EAST-ADL2 is defined with the development of safety-related embedded control systems as a benchmark. EAST-ADL2 bridges the gap from vehicle content definition and early analysis via functional design to the implementation perspective and back to integration and acceptance testing up to vehicle-level. An early, high-level representation of the system can evolve seamlessly into the detailed specifications of the AUTOSAR language. In addition, EAST-ADL2 incorporates the following system development concerns:

- Modelling of requirements and verification/validation information,
- Feature modelling and support for product lines,
- Structural and behavioural modelling of functions and hardware entities in the context of distributed systems,
- Modelling of variability of the system design,
- Environment, i.e., plant model and adjacent systems, and
- Non-functional operational properties such as a definition of function timing and failure modes, supporting system level analysis.

The main role of EAST-ADL2 is that of providing an integrated system model. As such, EAST-ADL2 must address multiple aspects of a system [3] including:

- Documentation, in terms of an integrated system model.
- Communication, by providing predefined views as well as the information sufficient for generating a number of other views.
- Analysis of a complete embedded system through the description of system structure and properties. Special emphasis has been placed on modelling support for analysis of component interfaces, timing correctness and safety analysis.

EAST-ADL2 and AUTOSAR in concert provide means for efficient development and management of the complexity of automotive embedded systems from early analysis right down to implementation. Concepts from model based development and component based development reinforce one another [4].

The following sections briefly summarize the language capabilities of EAST-ADL2 and present an illustrative example of its use. We conclude by comparing EAST-ADL2 with related work and discuss ongoing activities in the development of the language.

11.2 Modeling and Analysis Capabilities of the EAST-ADL2

EAST-ADL2 is a domain-specific language specified through a metamodel and implemented/released as a UML2 profile.

The primary structural organization of EAST-ADL2 is the division of the model into different abstraction levels (see Fig. 11.1). On a high abstraction level, only the externally perceivable aspects of the embedded system are handled, while on a low abstraction level the implementation-specific solution is managed in an AUTOSAR-conforming software architecture. This ensures separation of concerns and provides means to trace between the solution and problem domains.

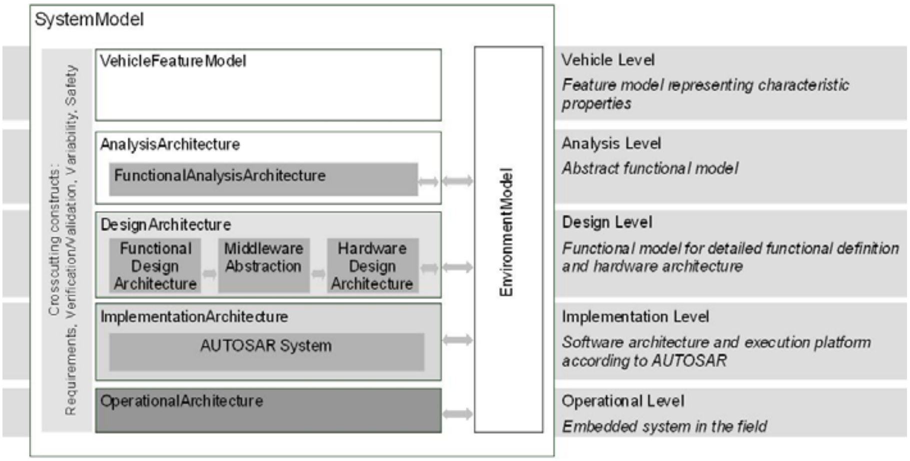
SystemModel
Crosscutting constructs:
Requirements, Verification/Validation, Variability, Safety
VehicleFeatureModel
AnalysisArchitecture
FunctionalAnalysisArchitecture
DesignArchitecture
Functional Design Architecture
Middleware Abstraction
Hardware Design Architecture
ImplementationArchitecture
AUTOSAR System
OperationalArchitecture
EnvironmentModel
Vehicle Level
Feature model representing characteristic properties
Analysis Level
Abstract functional model
Design Level
Functional model for detailed functional definition and hardware architecture
Implementation Level
Software architecture and execution platform according to AUTOSAR
Operational Level
Embedded system in the field

Fig. 11.1. EAST-ADL2 abstraction layers and relation to AUTOSAR. Cross-cutting concerns like requirements, V&V information, variability and safety/error modeling span all abstraction levels.

The abstraction levels implicitly represent different stages of an engineering process. However, the order in which a system is modelled could be top-down, middle-out, or bottom-up, in relation to the abstraction levels.

The structural organization of EAST-ADL2 is the backbone onto which additional modeling constructs are applied. Behavior, requirements, variability and safety aspects are examples of concepts that apply to the structural entities on several abstraction levels. A short summary follows below.

Behaviour

The goal of EAST-ADL2 with respect to behaviour is to define how model components (from different tools, in different modelling languages, or just representing code) are related to each other in order to capture behaviour and algorithms of the vehicle systems as well as the environment [5][6]. EAST-ADL enables structural components to refer to external behavior models, such as a Simulink model. The language also enables specification of triggering of blocks and precedence relations in their execution. The purpose of the behavioural definitions includes documentation, code generation and analysis, and the representation is chosen depending on the respective purpose.

Requirements

Requirements are captured in EAST-ADL2 according to the principles of SysML [7]: Requirements are separate entities that are associated to its target elements with a specific association, "ADLSatisfy".

Requirements are related to each other to support traceability between requirements. Typically, requirements on the higher abstraction levels of EAST-ADL2 are refined to more detailed requirements on lower abstraction levels.

Verification and Validation is supported through the concept of Verification & Validation Cases. These are linked to requirements and target entities, in order to show how a certain requirement is verified in the context of a specific model entity.

An important aspect of traceability is the possibility to follow which requirements are the results of safety concerns. This is needed to comply with the upcoming automotive standard for safety, ISO/WD 26262 [8]. EAST-ADL2 also supports this standard by providing support for safety case, safety integrity levels, and error propagation.

Variability

Variability is captured in EAST-ADL2 both in the feature models on vehicle level and in the architectures at analysis level and down. Feature model variability defines the permitted and expected variability regarding a certain aspect of the complete system. The idea is not to define how the system varies with respect to this aspect, but only that the system should exhibit such variability.

Variability on lower abstraction levels, on the other hand, defines how the feature variability is achieved. Variability mechanisms applied to the entities of EAST-ADL2 defines which of them are optional and under which circumstances they are included or excluded and the effect on structure and hierarchy. The mechanism is linked to the feature models such that variant choices in the feature model affects the variability resolution of the concrete architecture.

The variability management of EAST-ADL2 especially takes into account the automotive-specific challenges, e.g., management of a family of model ranges, different views on variability information (e.g., customer-related as opposed to development-related variability information), and extensibility of the variability management approach, e.g., for AUTOSAR modelling entities.

Error Modelling and Safety

State-of-the-art safety analysis techniques provide analysis support for deriving the causes and consequences of errors, based on a representation of the dependencies between system components. EAST-ADL2 provides means to manage the safety-related information together with the engineering information in a systematic way. An error model is defined with a structure that may be independent of the nominal system architecture. This way, the error model may capture errors and error propagation at the level of detail and according to a structure that is appropriate for the safety analysis at hand.

Further, it is possible to trace from the final implementation back to safety-related design choices and the applicable hazards. The requirement constructs for traceability and explicit support for safety cases according to the Goal Structure Notation [9] are relevant parts of EAST-ADL2 in this context.

11.3 A Small Case Study

To explain how the model is organized, an electric steering column lock function will be used. This is a security function for preventing any steering wheel movement without an authorized key. Traditional solutions for locking a steering column use the position of physical starter key as the authentication and unlocking mechanism. The introduction of immobilizers improved vehicle security by allowing advanced cryptography for authentication control prior to engine start. With a keyless engine start solution the steering lock also needs to be realized by the embedded system: a mechanical lock placed on the steering column is the actuation element, and a control unit reads the immobilizer transponder code and vehicle state and controls the mechanical lock accordingly.

11.3.1 Vehicle Features: Vehicle Level

To document what the embedded system provides to the user and other external stakeholders, a feature model is used. The feature model can be used as an entry point to related requirements, use cases, and other constructs. The feature model can be used to expose what the system provides and how a product line is organized in terms of available options and dependencies between options.

The steering column lock can be represented by a feature tree according to Figure 11.2. Steering column lock may be mechanical or electronic, and the electronic version may be based on a key or be key-less. Top level requirements are linked to each feature (not shown in Figure 11.2).

11.3.2 Abstract Functional Description: Analysis Level

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

The "Functional Analysis Architecture" for the example is sketched in Figure 11.2. The "ECL_Function" is the primary controller. It requires certain inputs including vehicle speed, engine status, the key position and more. The

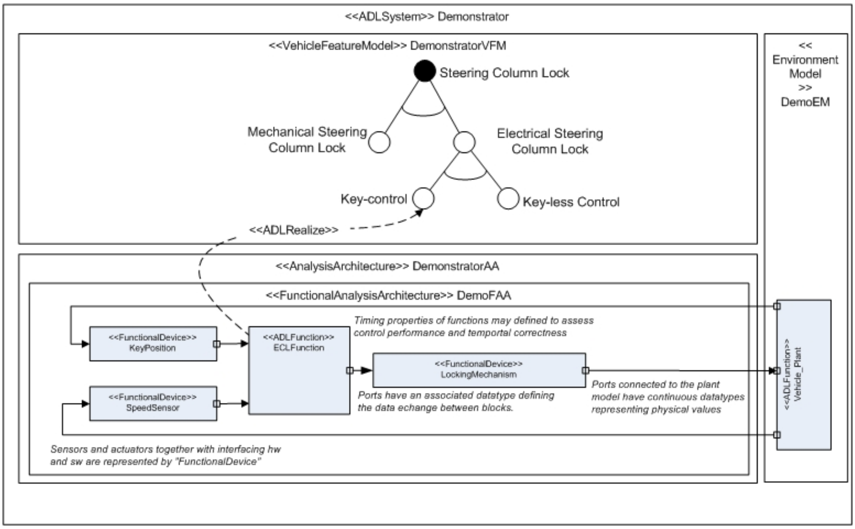
<<ADLSystem>> Demonstrator
<<VehicleFeatureModel>> DemonstratorVFM
Steering Column Lock
Mechanical Steering Column Lock
Electrical Steering Column Lock
Key-control
Key-less Control
<<ADLRealize>>
<< Environment Model >> DemoEM
<<AnalysisArchitecture>> DemonstratorAA
<<FunctionalAnalysisArchitecture>> DemoFAA
<<FunctionalDevice>> KeyPosition
<<FunctionalDevice>> SpeedSensor
<<ADLFunction>> ECLFunction
<<FunctionalDevice>> LockingMechanism
<<ADLFunction>> Vehicle_Plant
Timing properties of functions may defined to assess control performance and temportal correctness
Ports have an associated datatype defining the data echange between blocks.
Ports connected to the plant model have continuous datatypes representing physical values
Sensors and actuators together with interfacing hw and sw are represented by "FunctionalDevice"

Fig. 11.2. The feature model and abstract functional representation of the Steering Column Lock

"ECL_Function" outputs signals to the ECL actuator. Signals for exchange with the environment are continuous while signals within the embedded system are typically modelled as discrete. All sensors and actuators are modelled as FunctionalDevices with ports connecting to the physical environment. For those familiar with dynamic simulation environments such as MATLAB/Simulink, this is a similar view, allowing mixed discrete/continuous signals, and implicit time-marching.

11.3.3 Concrete Functional Description: Design Level

On the Design level, models are refined with more implementation-oriented aspects that allow a subsequent software decomposition of the functional architecture. While our functional analysis architecture above did not differentiate between application software, middleware and hardware, the functional design architecture now separates these areas of the system implementation. To distribute the systems functionality among these areas already constitutes an important design decision.

The abstract interface elements on analysis level ("FunctionalDevices") are realized by hardware elements such as sensors, actuators and amplifiers, and the software parts for signal transformation ("LocalDeviceManager"). Middleware abstraction projects the platform services and functionality (OS, AUTOSAR Basic software, etc.) to the functional level. The hardware architecture is introduced in parallel to capture the hardware entities as abstract elements (e.g. I/O, sensor, actuator, power, Electronic Control Units (ECU), electrical wiring including communication buses) describing the topology of the electronic architecture of the systems. Design level allows preliminary allocation of functional entities to ECUs and provides the basis for verification either by simulation or analysis techniques such as timing and dependability analysis.

The DesignArchitecture contains three parts, as shown in Figure 11.1, representing the application software, execution platform and hardware respectively. To show how they are related, a part of the example related to vehicle speed is shown in Figure 11.3. The Hardware Design Architecture can be seen as a circuit diagram of the system. The “StalkEcu” with its connected speed sensor is represented with its wiring. In addition, a transfer function of hardware devices can be specified to capture sensor and hardware characteristics as well as other behaviour of the hardware architecture.

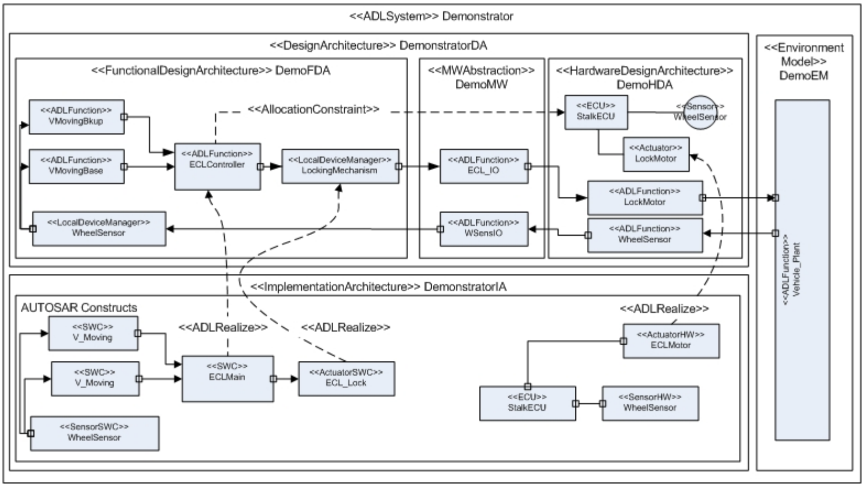
<<ADLSystem>> Demonstrator
<<DesignArchitecture>> DemonstratorDA
<<FunctionalDesignArchitecture>> DemoFDA
<<ADLFunction>> VMovingBkup
<<ADLFunction>> VMovingBase
<<ADLFunction>> ECLController
<<LocalDeviceManager>> LockingMechanism
<<LocalDeviceManager>> WheelSensor
<<AllocationConstraint>>
<<MWAbstraction>> DemoMW
<<ADLFunction>> ECL_IO
<<ADLFunction>> WSensIO
<<HardwareDesignArchitecture>> DemoHDA
<<ECU>> StalkECU
<<Sensor>> WheelSensor
<<Actuator>> LockMotor
<<ADLFunction>> LockMotor
<<ADLFunction>> WheelSensor
<<Environment Model>> DemoEM
<<ADLFunction>> Vehicle_Plant
<<ImplementationArchitecture>> DemonstratorIA
AUTOSAR Constructs
<<SWC>> V_Moving
<<SWC>> V_Moving
<<SensorSWC>> WheelSensor
<<ADLRealize>>
<<ADLRealize>>
<<SWC>> ECLMain
<<ActuatorSWC>> ECL_Lock
<<ADLRealize>>
<<ActuatorHW>> ECLMotor
<<ECU>> StalkECU
<<SensorHW>> WheelSensor

Fig. 11.3. Parts of design architecture and implementation architecture with realization and allocation relations. Note that only selected model entities and relations are shown.

The Middleware Abstraction contains a representation of the software platform that the applications rely on. In the example, the driver/interface software for the speed sensor is represented. The “WSensIO” represent the platform functionality that provides pulse rate of the wheel sensor.

The Functional Design Architecture contains the functionality that is subsequently realized by application software. The “WheelSensor” LocalDeviceManager translates to wheel speed according to the characteristics of the wheel sensor in use. The abstract function “ECLFunction” on analysis level is realized by three separate functions, two of which represent a redundant decision on whether the vehicle is moving.

11.3.4 Software Architecture: Implementation Level

System implementation in software is not represented by EAST-ADL2 entities, since this is the scope of AUTOSAR. However, the AUTOSAR entities are part of the system model to support traceability. Figure 11.3 also shows the AUTOSAR model and its relation to the EAST-ADL2 functions. As a realistic example would be too complex, only one-to-one mappings between AUTOSAR and EAST-ADL2 entities are shown. Readers not familiar with AUTOSAR may ignore the details and consider the shown entities as parts of the software and hardware architecture, respectively: The AUTOSAR software architecture typically shows a different structure than the functional architecture on design level. The purpose of the AUTOSAR hardware entities is to capture details necessary for the correct configuration of software. EAST-ADL2 provides a more abstract view of the hardware architecture, with a functional description of hardware elements and support for early assessment of feasibility of the system realization.

11.4 Related Work, Conclusions and Further Work

Model based development for embedded systems, and in particular automotive systems can be supported in various ways. The AADL is a modelling language dedicated to embedded systems with its roots in the aerospace domain. Compared to the EAST-ADL2 and AUTOSAR combination it covers parts of this scope. However, because of its overlap with AUTOSAR on the software architecture level, and the lack of complementary abstraction levels it does not provide an appropriate structural framework for automotive systems development. Also, the support for feature modelling, requirements and variability is unique for EAST-ADL2.

SysML and MARTE are UML profiles that augment plain UML with constructs for systems engineering and embedded real-time systems modelling, respectively. Both approaches, and even plain UML are useful tools in automotive development and EAST-ADL2 has integrated some of these concepts, for example requirement concepts from SysML and timing constructs from MARTE. But the abstraction levels and tailored model structure as well as complementary constructs of EAST-ADL2 adds a framework that both supports the modelling needs and guides modelling in a way that improves model exchange and understanding between stakeholders.

Off-the-shelf tools like SCADE, ASCET, Simulink, etc. all support model based development with analysis and synthesis to various degrees. Our conclusion, however, is that no single tool will be used for an entire vehicle development project, but model integration is necessary. EAST-ADL2 supports this aspect by allowing external representation of behaviour and concepts for integration with requirements management tools.

Another effort with large impact on automotive systems is the safety standard developed by the ISO working group on functional safety for road vehicles (ISO TC 22/SC 3/WG 16), ISO/WD 26262. The standard calls for rigorous development methods and requires documentation that shows that adequate measures are taken to achieve safety. EAST-ADL2 provides a framework that makes this possible and includes dedicated constructs for safety assessment and documentation.

Having had the opportunity to define this architecture description language in parallel with the dynamic phases of the definitions of AUTOSAR and ISO/WD 26262, EAST-ADL2 has a good potential to become a de facto standard as it fits well with the major critical needs of the automotive industry of today.

The language has received further momentum from its deployment in several industrial research projects that claim the central role of EAST-ADL2 in their work. Among others, the ADAMS project (`www.adams-project.org`) leads the dissemination of MARTE, where AUTOSAR and EAST-ADL2 are of primary importance for the automotive domain. Another example is the EDONA project (`http://www.edona.org`) in which EAST-ADL2 is a cornerstone of an integrated tool suite for the automotive domain. Finally, TIMMO (`www.timmo.org`) defines a methodology and representation of timing aspects in automotive embedded systems, where EAST-ADL2 together with AUTOSAR is the basis.

To enable a wide spread use of EAST-ADL2, its UML2 implementation is released as a public UML2 profile. The profile is supported in the open-source UML modeller Papyrus which can be downloaded on the `www.atesst.org` or the `www.papyrusuml.org` websites. In the ongoing European research project ATESST2, the EAST-ADL2 is currently extended in several areas:

- Modelling concepts for requirements and verification and validation are extended to support e.g. views on requirements and product line support.
- Timing modelling extensions, as being developed in the TIMMO project, will be integrated into the EAST-ADL2.
- Various aspects of native behavior descriptions are being further investigated for potential inclusion including explicit support for modes of operation and representation of continuous-time behavior as part of environment models.
- Variability mechanisms are used to choose between different behavioural representations.
- Variability concepts are extended to support product-line oriented manufacturer-supplier exchange.
- New dependability and cost modelling concepts are being developed to support multi-objective optimisation of system models, in conjunction with tools such as HiP-HOPS [10] that provide such advanced capabilities. The aim of optimization is to automatically evolve models that do not necessarily meet dependability requirements (e.g. safety, reliability or availability) to designs that fulfil such requirements with minimal costs [11]. Optimization can be done via exploration of potential design spaces using meta-heuristics such as genetic algorithms. The specification of design alternatives and variant sub-architectures the combinations of which define the potential design space can be described in EAST-ADL2 by using the variability constructs of the language. This work pushes the boundaries of the state-of-the-art in this area, as no modelling language provides support for such unique capabilities in design.

- Specialised plug-ins, based on the UML profile of the language, are being developed to achieve practical integration of EAST-ADL2 with existing tools. For example, data exchange will be supported with a plug-in for the RIF [12] requirement interchange format. As another example behavioural simulation, safety analysis and optimisation of models will be supported with plug-ins for Hip-HOPS and MATLAB/Simulink..

In addition, a methodology for the EAST-ADL2 will be developed, that explains the use and the interrelation of the different modelling concepts on the different abstraction levels during system specification and design as well as integration and testing.

Acknowldegements

The authors acknowledge the financial support provided by the European Commission through the Project ATESST2 (call FP7-ICT-2007-2, grant agreement number 224442).

References

[1] AUTOSAR Development Partnership: AUTOSAR Development Partnership (2007), http://www.autosar.org

[2] Sangiovanni-Vincentelli, A., Di Natale, M.: Embedded system design for automotive applications. Computer 40(10), 42–51 (2007)

[3] Törngren, M., Chen, D.J., Malvius, D., Axelsson, J.: Model based development of automotive embedded systems. In: Handbook on Automotive Embedded Systems. Taylor and Francis CRC Press - Series: Industrial Information Technology (invited) (forthcoming 2008), ISBN=9780849380266

[4] Törngren, M., Chen, D.J., Crnkovic, I.: Component-based vs. model-based development: A comparison in the context of vehicular embedded systems. In: EUROMICRO-SEAA, pp. 432–441 (2005)

[5] ATESST consortium: Report on behavioral modeling within east-adl2, d3.2 deliverable. Technical report (December 2007), http://www.atesst.org/

[6] Sjöstedt, C.J., Shi, J., Törngren, M., Servat, D., Chen, D., Ahlsten, V., Lönn, H.: Mapping Simulink to UML in the Design of Embedded Systems: Investigating Scenarios and Structural and Behavioral Mapping. In: OMER4 Post-Proceedings (2008)

[7] SysML Partners: Systems Modeling Language (SysML) open source specification project, http://www.sysml.org

[8] International Organization for Standardization: ISO Working Draft 26262 Baseline 10 (2007)

[9] Kelley, T.P.: Arguing Safety - A Systematic Approach to Managing Safety Cases. PhD thesis, University of York (1998)

[10] Papadopoulos, Y., McDermid, J.A.: Hierarchically performed hazard origin and propagation studies. In: Felici, M., Kanoun, K., Pasquini, A. (eds.) SAFECOMP 1999. LNCS, vol. 1698, pp. 139–152. Springer, Heidelberg (1999)

[11] Papadopoulos, Y., Grante, C.: Evolving car designs using model-based automated safety analysis and optimisation techniques. J. Syst. Softw. 76(1), 77–89 (2005)

[12] HIS: Specification Requirements Interchange Format (RIF), version 1.1a (2007)
[13] OMG: Uml profile for modeling and analysis of real-time and embedded systems (marte), beta1, omg document number: ptc/07-08-04 (August 2007)
[14] The Motor Industry Software Reliability Association (MISRA): Development guidelines for vehicle based software (1994)
[15] International Electrotechnical Commission: Functional safety of electrical/electronic/ programmable electronic safety-related systems - part 0: Functional safety and iec 61508 (2005)
[16] Törner, F., Chen, D.J., Johansson, R., Lönn, H., Törngren, M.: Supporting an automotive safety case through systematic model based development - the east-adl2 approach. In: SAE World Congress (2008), SAE paper number 2008-01-0127