Modelling Support for Design of Safety-Critical Automotive Embedded Systems

DeJiu Chen[1], Rolf Johansson[2], Henrik Lönn[3], Yiannis Papadopoulos[4], Anders Sandberg[5], Fredrik Törner[6], and Martin Törngren[1]

[1] Royal Institute of Technology, SE-10044 Stockholm, Sweden
{chen,martin}@md.kth.se
[2] Mentor Graphics Corp., SE-41755 Gothenburg, Sweden
rolf_johansson@mentor.com
[3] Volvo Technology Corp., SE-40508 Gothenburg, Sweden
Henrik.lonn@volvo.com
[4] University of Hull, Hull HU6 7RX, UK
Y.I.Papadopoulos@hull.ac.uk
[5] Mecel AB, SE-400 20 Gothenburg, Sweden
anders.sandberg@mecel.se
[6] Volvo Car Corp., SE-40531 Gothenburg, Sweden
ftorner@volvocars.com

Abstract. This paper describes and demonstrates an approach that promises to bridge the gap between model-based systems engineering and the safety process of automotive embedded systems. The basis for this is the integration of safety analysis techniques, a method for developing and managing Safety Cases, and a systematic approach to model-based engineering – the EAST-ADL2 architecture description language. Three areas are highlighted: (1) System model development on different levels of abstraction. This enables fulfilling many requirements on software development as specified by ISO-CD-26262; (2) Safety Case development in close connection to the system model; (3) Analysis of mal-functional behaviour that may cause hazards, by modelling of errors and error propagation in a (complex and hierarchical) system model.

Keywords: Automotive Embedded Systems, Dependability, Model-Based Development, Safety Analysis, Safety Case.

1 Introduction

Safety is posing an increasing challenge for the developers of automotive embedded systems, also referred to as automotive Electrical/Electronic (E/E) systems. While accounting for a large portion of the innovations and flexibility, the underlying computer software and hardware also results in growing product complexity. The last decade has indeed shown that an increasing number of vehicle failures stem from errors related to embedded systems.

Currently, ISO is developing a standard on Functional Safety for Road vehicles (ISO-CD-26262) [1]. As pointed out in its introduction, with the high complexity growth there is an increasing risk of failures in automotive embedded systems. This

M.D. Harrison and M.-A. Sujan (Eds.): SAFECOMP 2008, LNCS 5219, pp. 72–85, 2008.

makes safety a key issue in future automobile development. ISO-CD-26262 requires that a complete Safety Case is developed, presenting evidence that the system is safe. It also specifies the requirements on product development at software level.

While state-of-the-art safety analysis techniques [2] provide support for deriving the causes and consequences of errors, the difficulties remain in capturing and maintaining plausible errors, safety requirements, and other related information along with design refinement, changes and evolution, and in providing the safety argument. Such analysis techniques in turn rely on system modelling and management support, as well as the alignment with tools, processes, and standards. One challenge with current methods for automotive E/E systems development is the lack of systematic approaches to information management, architecting and verification. Solutions relying on social and traditional text-based communication do not scale for handling advanced embedded systems. Software architectures and/or exchange format standards such as AUTOSAR [3] offer a significant improvement of the state of practice. However, experience tells us that advanced and complex systems also require model-based engineering encompassing appropriate abstractions and views for both cost-efficiency and development effectiveness. Over the years, the demand for additional levels of abstractions and views has been continuously raised [6, 12].

System modelling based on an architecture description language (ADL) is a way to keep the engineering information in a well-defined information structure. In this paper we present how the architecture description language EAST-ADL2, complementary to AUTOSAR, provides a basis for systematic development of safety-critical automotive systems. As a language for architecture description, the EAST-ADL2 captures the domain knowledge for automotive embedded systems and provides the modelling means for keeping various engineering information, e.g., across multiple levels of abstraction and concerns, within one infrastructure. Three important areas of EAST-ADL2 will be highlighted in this paper: (1) System development based on models on different levels of abstraction. This enables fulfilling many requirements on software development as specified by ISO-CD-26262; (2) Safety Case development in close connection to the system design; and (3) Analysis of hazardous failures by modelling of errors and the propagations in a hierarchical system model. The integration of these aspects provides structured information handling of requirements, design, safety analysis, other verification and validation information, and design decisions. The approach supports reuse, consistency between models, automated handling of dependencies, view generation, transformations and analysis.

The paper is organised as follows: We first give an overview of EAST-ADL2 showing its capabilities for model-based development, and how it is complementary to AUTOSAR. Then we describe the modelling support for a Safety Case. In the following section we describe error modelling and modelling of error propagation, and the link to the HiP-HOPS safety analysis tool. Finally, we illustrate the approach with an industrial case study on one ECL (Electronic Column Lock) system.

2 Overview of EAST-ADL2

EAST-ADL2 is developed in the ATESST project (www.atesst.org), further extending and refining the EAST-ADL language from the EAST-EEA project (www.east-eea.org). It is a domain-specific architecture description language aiming to

adequately meet the engineers' needs regarding information management and practical methods in the development of advanced automotive embedded systems. The language provides an ontology for all the related engineering information and a set of well-defined constructs for the capturing and structuring of such information in a standard format. The covered system aspects include requirements, vehicle features, functions, variability, software and hardware design, and environment, as well as the related structures and behaviours. For the purpose of early quality assessment and verification, the language also supports the capturing of other necessary non-functional properties and thereby enables the reasoning of system timing and failure modes. Through its constructs for traceability, the EAST-ADL2 allows the modelling of dependencies across requirements, structural items and V&V information.

2.1 Hierarchies and Levels of Abstraction

While stipulating the abstractions and viewpoints that are of particular importance in the development of automotive embedded systems, the EAST-ADL2 further enforces separation-of-concerns and complexity-control through a multi-viewed and hierarchical modelling language. The core concept is to structure the solution architecture into five levels of abstraction: *VehicleLevel*, *AnalysisLevel*, *DesignLevel*, *ImplementationLevel*, and *OperationalLevel*. See also Figure 1. The levels correspond to system views that can be used to support a variety of processes (from top-down to bottom-up), including the typical scenario for platform-based product families, where a new function is added to an existing system. The architectural solution at each abstraction level is self-contained in the sense that it constitutes a complete model of the system under consideration from a particular viewpoint. Within each of these architectural solutions, hierarchies of composition are supported by dedicated constructs to describe the part-whole relations of functions/components.

The models at the *VehicleLevel* provide a top-level view of the E/E system of a vehicle where the intended electronic features are described and elaborated in respect to the related product-line organizations. One view that captures the realizations of such

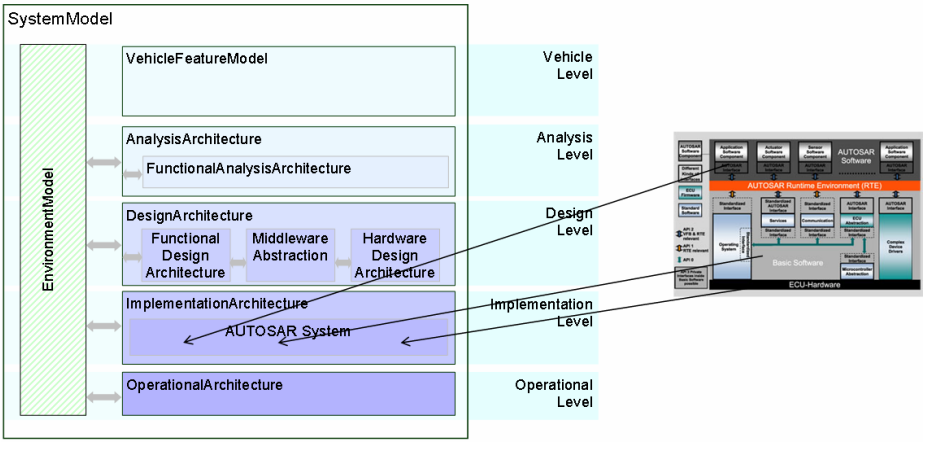

Fig. 1. EAST-ADL2 abstraction layers and its relation to AUTOSAR [9]

electronic features in terms of logical functions and principal interfaces is given at the *AnalysisLevel*. A further refined view is provided at the *DesignLevel* where more implementation-oriented aspects are taken into consideration, such as alignment with intended software decomposition and the target platform, fault tolerance, sensor and actuator interfacing, etc. The support by EAST-ADL2 at this level includes the functional design architecture for application software, the middleware abstraction for platform software (e.g., middleware, RTOS etc.), and the hardware architecture for target platform (e.g. I/O, sensor, actuator, power, ECU, topology and electrical wiring including communication bus). It allows the reasoning of partitioning and allocation of functions as well as the verification of the preliminary design either by simulation or analysis techniques. The overall structure at the *DesignLevel* is such that one or several entities can be later realized by AUTOSAR entities captured at the *ImplementationLevel* [9]. Full traceability is possible from function definitions at the vehicle level to AUTOSAR entities. The *OperationalLevel* is hidden by AUTOSAR concepts via deployment on the AUTOSAR RTE (Run-Time Environment), representing the E/E system as it is realized in the manufactured products.

One example of this hierarchical multi-viewed modelling approach is illustrated in Figure 2, with an electronic feature *Brake* (denoted by the *EFeature* construct), models of more detailed solutions, and the final implementation. The solutions at the *AnalysisLevel* include the logic function *BrakeCtrl* (denoted by the *ADLFunction* construct) and the abstract interfaces *BrakePedal* and *BrakeMotor* for the interactions with the vehicle environment (denoted by the *FunctionalDevice* construct). The corresponding software and hardware design solutions are shown at the *DesignLevel*. While the logic function *BrakeCtrl* is realized by the software function *BrakeCtrl*, the abstract interfaces are represented by hardware devices (denoted by the *DeviceIF* construct) and software components for signal transformation (denoted by the *LocalDeviceManager* construct). The implementation of the design is given by AUTOSAR concepts at the *ImplemenationLevel* (e.g., an elementary *ADLFunction* is mapped to a *RunnableEntity* of an *AtomicSoftwareComponent*).

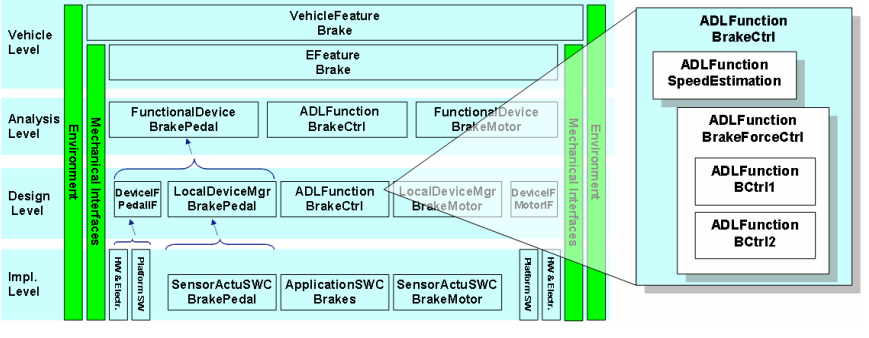

Fig. 2. An example showing the electronic feature (Brake) and its representations with the EAST-ADL2 abstraction levels

2.2 Requirements and Traceability Support

EAST-ADL2 provides explicit support for requirement specification and management in the development of advanced embedded systems. It differentiates between functional requirements, which typically focus on some part of the "normal" functionality that the system has to provide (e.g. "ABS shall control brake force via wheel slip control"), and quality requirements, which typically focus on some external property of the system seen as a whole (e.g. "ABS shall have an MTTF of 10,000 hours"). To allow integration of external requirements tools, EAST-ADL2 provides supports for the mapping of Requirements Interchange Format (RIF) [7] concepts.

The language treats requirements as separate entities and provides specific constructs to support the traceability by extending and adapting related principles from SysML [8]. Typically, based on requirements on the higher abstraction levels of EAST-ADL2, more detailed requirements are derived along with the refinements and decompositions. Specific associations are introduced to relate requirements to their target elements (through the *ADLSatisfy* construct). EAST-ADL2 introduces the notion of *Verification&Validation Case* (denoted by the *VVCase* construct) in order to show how a certain requirement is verified in a particular system context as well as to support the planning, tracking, and updating of V&V efforts,. While linking certain requirements and target entities, each *VVCase* provides a description of the related evaluation information and activities.

3 Safety Case Support in EAST-ADL2

A Safety Case provides structure to the qualitative argumentation about why a system is safe enough. Hence, the Safety Case is dependent on referencing and aggregating information of different types related to the systems functionality and realization. Therefore, integration with an ADL is useful for system development. Currently, there are no requirements for Safety Cases in the automotive industry, but in the upcoming automotive safety standard ISO-CD-26262 [1] such requirements are raised. This section will provide the safety case metamodel which was implemented in EAST-ADL2. A more detailed description is presented in [10].

A Safety Case can consist of large amounts of data and may be very hard to grasp. To mitigate the complexity, a graphical notation, the Goal Structure Notation, have been introduced by Kelly for the argumentation part of a Safety Case [4]. The notation consists of the following building blocks:

- *Goal* – A claim about a property of the system.
- *Strategy* – A description of how and why a Goal can be derived into other Goals.
- *Justification* – Provides further rationale for a selected GSN entity.
- *Evidence* – This is the set of leafs of an argumentation representing the actual evidence that shows satisfaction of the goals it is connected to.
- *Context* – Defining in what context a Goal is given.

A safety case metamodel is shown in Figure 3. It is based on a description of GSN and shows how the GSN entities relate to each other. The safety case entity itself is the top level of a safety case, and it consists of the GSN argument entities. The safety

case can also consist of several other safety cases, in a hierarchical structure. In order to maximize the traceability of the design data, each GSN class can be associated to any EAST-ADL2 entity. This will provide support for consistency of the data as well as support for the change management process.

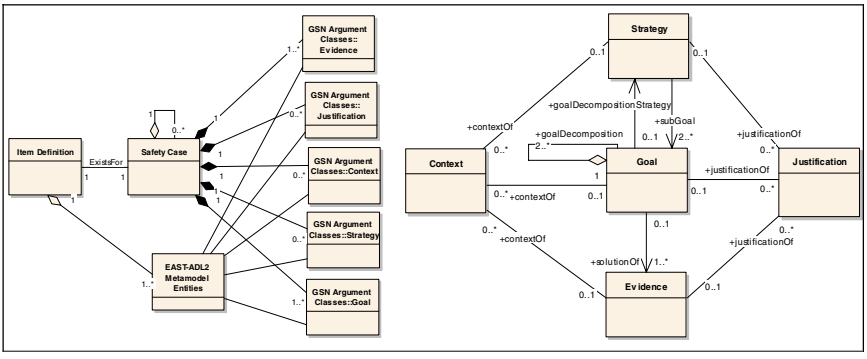

Fig. 3. The safety case metamodel based on GSN

A *Safety Case* is valid for a system or function, and this scope needs to be defined. As shown in Figure 3, the safety case scope is defined by the *ItemDefinition* class that is a collection of EAST-ADL2 entities, i.e. all available specifications entities for the given item. The metamodel also contains the relations between the internal elements. As shown in Figure 4, the *Goal* is the centre of the safety case structure. It can be decomposed directly, or through the usage of a strategy, into two or more *Goals*. Each *Goal* shall have a solution relation to at least one *Evidence*, also known as *Solution* in the GSN notation. The *Evidence* entity can have several specializations, ranging from protocols of V&V activities to design decisions. Each element in the GSN structure can also be related to a *Context* entity indicating that a description of the context can be provided. Similarly, a *Justification* for increased clarity can be provided for each GSN entity, by a justification relation.

The *Evidence* represents any information that supports or, in its ultimate form, proves that the *Goal* it is connected to is achieved. As such, the information can be of many types, e.g. analysis reports, design specifications, requirements, protocols from V&V activities, etc. The system model of EAST-ADL2 captures most of these entities in suitable packages, e.g. a package focusing on verification and validation, the V&V package. In [10], the described safety case metamodel clearly visualizes the GSN entities and interdependencies which has the advantage of facilitating the comprehension and easing the training effort.

4 Error Modelling Support in EAST-ADL2

As an overall system property, safety is concerned with the anomalies (in terms faults, errors, and failures) and their consequences under given certain environmental conditions. Functional safety represents the part of safety that depends on the correctness of

a system operating in its context [11] and addresses the hazardous anomalies of a system in its operation (e.g., component errors and their propagations). The objective of the EAST-ADL2 error modelling is to allow an explicit reasoning of functional safety and thereby to facilitate safety engineering along with an architecture design or maintenance process.

4.1 Key Concepts and Domain Model

EAST-ADL2 facilitates safety engineering in regards to the modelling and information management. While supporting the safety design through its intrinsic architecture description and traceability support, the language also allows the developers to explicitly capture the error logics in terms of component errors and the error propagations in an architecture error model through its error modelling support (see also Figure 4). The error modelling is treated as a separated analytical view. It is not embedded within a nominal architecture model but seamlessly integrated with the architecture model through the EAST-ADL2 meta-model. This separation of concerns is considered necessary in order to avoid some undesired effects appearing when error modelling and nominal design is mixed, during comprehension and management of nominal design, reuse of models, and system synthesis (e.g., code generation).

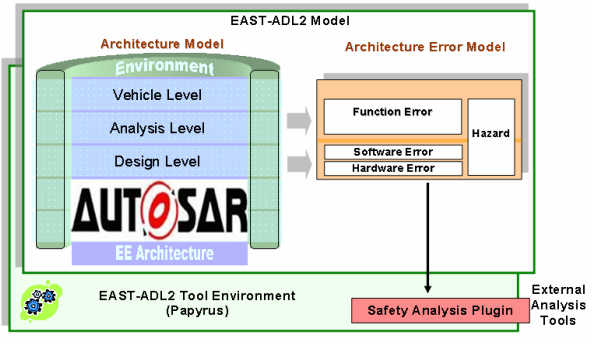

Fig. 4. EAST-ADL2 error modelling extends the nominal architecture in a separate view and provides analysis leverage through external tools

The EAST-ADL2 error modelling package extends a nominal architecture model, typically at the *AnalysisLevel* and *DesignLevel*, with the information of failure semantics and error propagations. The failure semantics can be provided in terms of logical or temporal expressions, depending on the analysis techniques and tools of interest. Such analytical information, together with environmental conditions, forms the basis for identifying the likely hazards, reasoning about the causes and consequences, and thereby deriving the safety requirements. The relationships of local error behaviours are captured by means of explicit error propagation ports and connections. Due to these artefacts, EAST-ADL2 allows advanced properties of error propagations, such as the logical and temporal relationships of source and target errors, the conditions of propagations, and the synchronizations of propagation paths. Hazards or hazardous events are characterized by attributes for severity, exposure and controllability

according to [1]. A hazardous event may be further detailed by e.g. use cases, sequence or activity diagrams. In an architecture specification, an error is allowed to propagate via design specific architectural relationships when such relationships also imply behavioural or operational dependencies (e.g., between software and hardware). Fig. 5 shows the domain model definitions of constructs for the error modelling. The key concepts include:

- *ErrorBehavior*: the definitions of possible failure behaviours of an *ADLEntity* (i.e., an abstract function or component).
- *ErrorModel*: the container for the usages or instantiations of particular *errorBehaviors* in a particular architecture context.
- *propagationPort*: ports through which the faulty events defined in an ErrorBehavior propagate to other *ErrorBehaviors* or result in *Hazards*.
- *ErrorPropagation*: abstractions for error propagations that in turn relies on particular instances of *ADLEntity* (e.g. communication connectors) or the explicit or implicit dependencies between them (e.g., allocations described by the *ADLRealization* construct).
- *ErrorToHazard*: a link between *errorBehaviors and* their effects on the system.

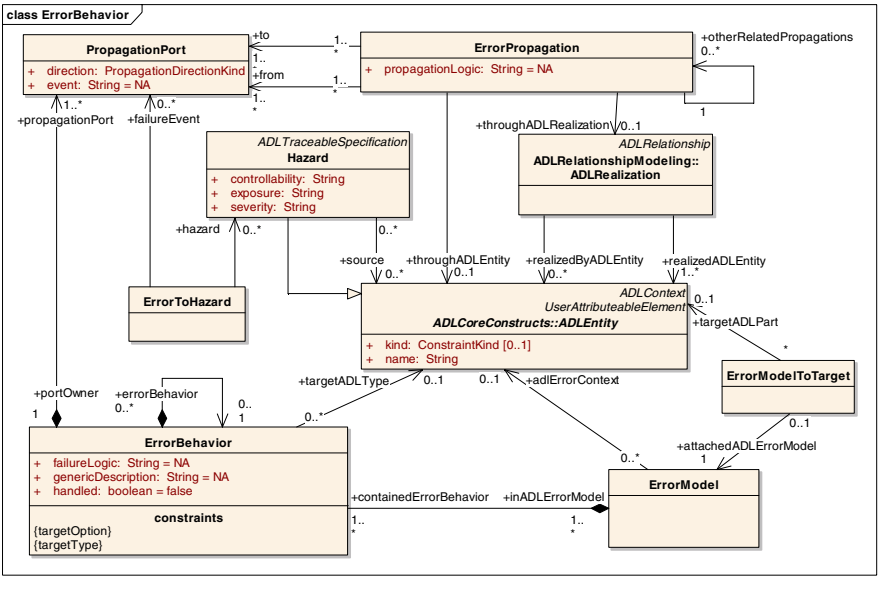

Fig. 5. The EAST-ADL2 domain model definitions for error modelling

In EAST-ADL2, the support for safety requirements and analysis is specifically addressed. The safety requirements, which are specialized to define the safety goals to be met, have attributes and related entities to define the related functional and non-functional requirements and the hazards to be mitigated. Hazards or hazardous events are associated with both errors of abstract functions/components and the environment model and characterized by attributes for severity, exposure and controllability. See

Fig. 6. for the domain model definition. This concept is in line with [1] where each hazard is related to an *Item*, which is defined as "E/E system (i.e. a product which can include mechanical components of different technologies) or a function which is in the scope of the development according to this standard". When modelling a system in EAST-ADL2 this means that for each level of abstraction a complete set of *Items* is identified. The hazardous event may be further detailed by e.g. use cases, sequence or activity diagrams. A safety requirement specifies the necessary safety functions and their effectiveness (i.e., ASIL levels [1]). It can be traced all the way to its derived requirements and thereby to the subsequent hardware and software solutions as well as the needed V&V efforts.

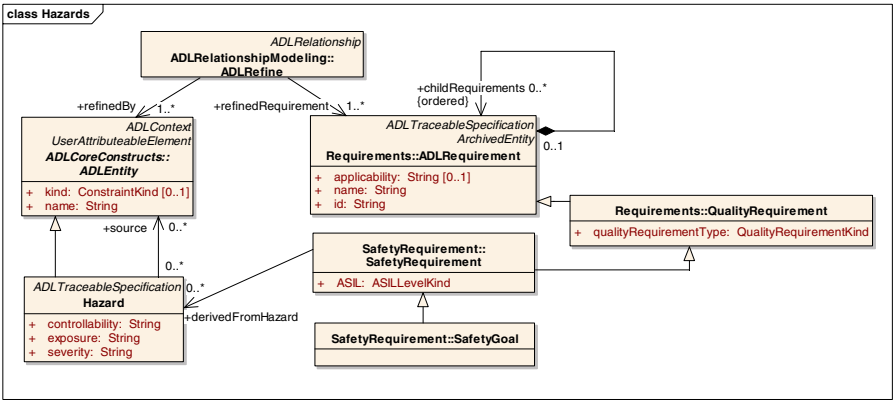

Fig. 6. The EAST-ADL2 domain model definition for hazard and safety requirements

4.2 Analysis Leverage and Tool Support through HipHOPS Method

A proof-of-concept tool integration with the HiP-HOPS method (Hierarchically Performed Hazard Origin and Propagation Studies) [5] has been developed. HiP-HOPS is a model-based safety and reliability analysis technique in which topological descriptions of the system (hierarchically composed if required to manage complexity) that are annotated with formalised logical descriptions of component failures, are used as a basis for the automatic construction of fault trees and Failure Modes and Effects Analyses (FMEA) for a system. Suitable models include a range of diagrams commonly used to express hardware and software architectures.

Through the EAST-ADL2 error modelling support, a HiP-HOPS study can be performed on the abstract models at the *AnalysisLevel* or on the more detailed architecture models at the *DesignLevel*. This creates opportunities for systematic identification of safety related requirements, re-use of earlier analysis, and the ability to achieve a consistent and continuous assessment in the centre of which lies the design of the system itself. Given an EAST-ADL2 model which contains descriptions of *ErrorBehaviours*, a global view of system failure can be captured via HiP-HOPS in a set of system fault trees which are automatically constructed as expressions that describe local fault propagation are being evaluated during the traversal. The synthesised fault trees are interconnected and form a directed acyclic graph sharing branches and basic events that arise from error propagations defined in the model. Classical

Boolean reduction techniques and recent algorithms for fault tree analysis that employ Binary Decision Diagrams (BDDs) are applicable on this graph. Thus, qualitative analysis (e.g. of abstract functional models) or quantitative analysis (e.g. calculation of system-level failure rates from known probabilistic component data) can be automatically performed on the graph to establish whether the system meets the desired safety or reliability. The logic in the graph can also be automatically transformed into a simple table which is equivalent to a multiple failure mode system.

5 Example Case Study: Electronic Column Lock

Steering column lock is a security function for preventing any steering wheel movement without an authorized starter key. Traditional solutions use the position of physical starter key as the securing and unlocking mechanism. For the reasons of user-friendliness, as well as crash safety and vehicle security, keyless engine start solutions with the immobilizer transponder and start button have been increasingly adopted, allowing advanced cryptography for authentication control prior to engine start. As a physical starter key is no longer present, there is a need to replace the traditional steering column lock principle. With electric steering column lock (ECL), a logical key position rather than the physical key position is used to enable and disable steering. The implementation normally consists of a mechanical lock placed on the steering column as the actuation element, and a control unit for reading the immobilizer transponder code and vehicle state and for controlling the mechanical lock. Fig. 7. depicts the modelling coverage by EAST-ADL2 for an ECL system.

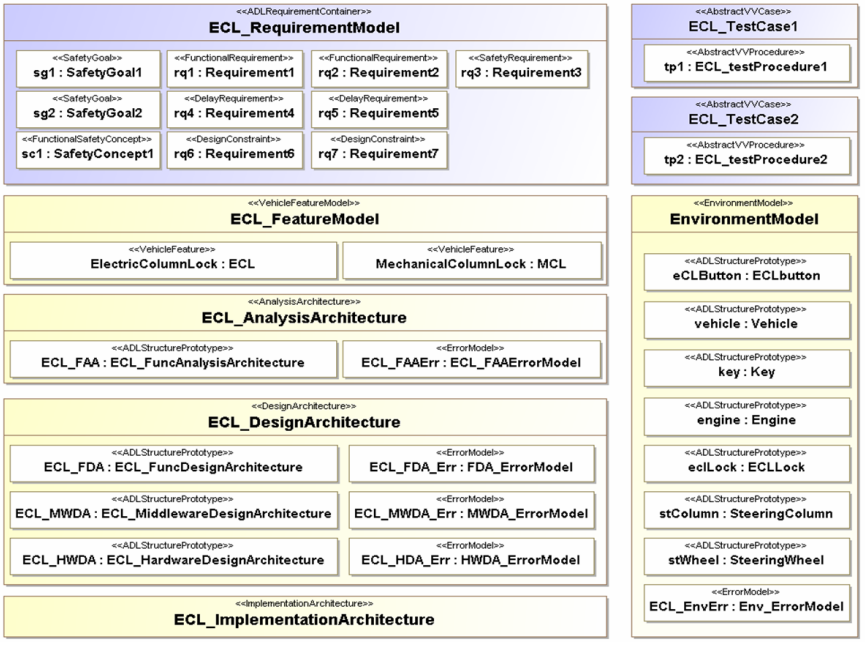

Fig. 7. The modeling coverage of EAST-ADL2 for an ECL system

Taken from the legislation, 95/56/EC annex IV, some of the basic requirements for the security function with relevance for safety are: (1) Devices to prevent unauthorized use shall be such as to exclude any risk of accidental operating failure while the engine is running, particularly in the case of blockage likely to compromise safety; (2) Locking shall only be possible after making one operation to stop the engine and then a second operation designed to lock the column.

The major top level hazard is*: Steering is disabled while driving*. This top level hazard must be controlled to a risk level where the risk of the hazard is kept sufficiently low. To model this hazard we use the mechanisms described in Fig 3. The initial Safety Case for ECL will consists of a root 'Item', with one 'Hazard' and two 'ADL Functions', One being the user function and the other the Environmental Model entity for the steering wheel. The safety assessment; in this case this is an function with the highest safety integrity level (ASIL D according to ISO 26262), is required to enable the correct actions on technical and process elements on the function development. The safety analysis for the function requires that we make an architecture defining which outputs and inputs are needed in order to perform the user function. Fig 8. shows the abstract functional definition of the ECL.

Sensors for vehicle speed, engine speed and key-position are required for ECL operation. For unlocking there are requirements on using an approved key, this is not fully considered in this example. There is also the case of retry strategies that can use

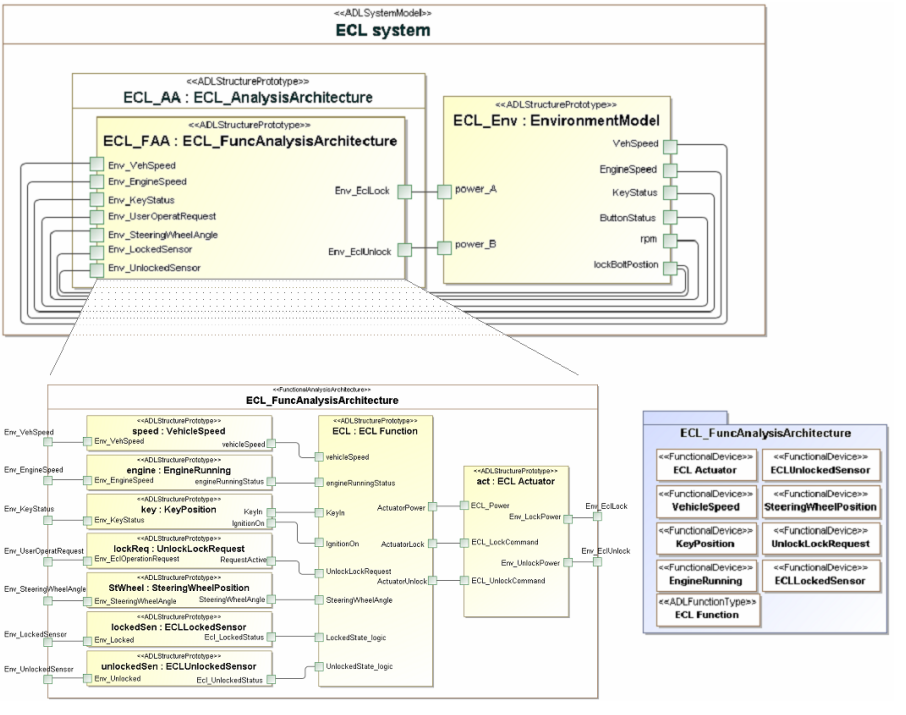

Fig. 8. Functional Analysis Architecture with Electrical Column Lock function, where the components from the package "ECL FuncAnalysis Architecture" act as parts

movement of the steering wheel as a trigger. The results of the safety analysis are the basic safety requirements and how these must be implemented. In this example, the outcome is that if the ECL lock is unpowered when the vehicle is moving and/or the engine is running, the Hazard cannot occur (unless there are electrical hardware or mechanical failures). The following architecture requirements (shown in Fig. 7) capture this and should be implemented in the design of the ECL function:

- **SafetyGoal1** – The ECL must not be powered when vehicle is moving and ignition is on.
- **SafetyGoal2** – The ECL must not be powered when engine is running and ignition is on.

As we rely on vehicle speed, engine running and ignition state on an ASIL D class function, we need to consider the need for redundant decision making on this set of data. As illustrated in the example case, the *ADLFunctions* on the highest level are similar between the analysis and the design level respectively. When we realize the *AnalysisArchitecture* (AA) on the analysis level with the *DesignArchitecture* (DA) on the design level, we can still hide a number of details on the highest level of the chosen ADLFunction hierarchy. Please observe the difference between the *realization* that is done when going from the more abstract analysis level to the design level (relation between "Architectures" in Figure 7), and the unpacking of details that is done when going down an ADLFunction hierarchy (the composition relation between contained and containing functions in Figure 8).

To perform a HiP-HOPS based hazard analysis, the first step is the establishment of *ErrorBehaviors* of components (i.e. functions, hardware or software elements) as failure expressions which show how output failures of each component can be caused by internal malfunctions and deviations of the component inputs. A variant of Hazard and Operability Studies (HAZOP) can be used to identify plausible output failures such as the omission, commission, value (hi, low) or timing (early, late) failure of each output and then to determine the local causes of such events as combinations of internal component malfunctions and similar types of input failures. Thereafter, the structure for error propagations in a particular architecture context is determined using *ErrorModels* for instantiating the predefined *ErrorBehaviors* and *ErrorPropagation* for the propagations. The system effects of failure events in terms of *Hazards* are captured with the *ErrorToHazard* construct. This global view in terms of fault trees are the generated with the HiP-HOPS plug-in.

Our experience from case studies suggests two useful design patterns that can be derived from this type of analysis: (a) when the analysis indicates that the omission of a function has only marginal effects while commission and value failures have catastrophic effects, a design recommendation should be made to design the function in a way that it "fails silent"; (b) on the other hand, when all potential failure modes of a function are shown to have catastrophic effects on the system then a design recommendation should be made to allocate the function to a fault tolerant architecture. With the help of these results, the abstract functions are allocated to appropriate hardware and software architectures in which case HiP-HOPS studies can become much more detailed and quantitative in nature making use of available information about component failure modes and failure rates.

ErrorBehaviours are now extended to include real failure modes and probabilistic component failure data. Such failure modes include electrical and mechanical

component failures caused by wear and environmental conditions or, in the case of programmable components, statistically observed functional failures caused by unspecified random or systematic faults. Note that credible probabilistic failure data (often not available) is not essential for producing useful results. Qualitative application of the technique can still produce useful results. The logical reduction of fault trees into minimal cut-sets and FMEA, for instance, can indicate single points of failure in the system and point out potential design weaknesses. Clearly, the ability to iterate fast this process ultimately also defines the ability to manage effectively the evolution of an EAST-ADL2 design.

We should note that there is a range of other emerging techniques that are aiming at automation of system safety analysis (Altarica [13] and FSAP-NuSMV [14] among others). Most use model-checking and simulation as means of inferring the effects of component failures in a system. However, the analysis of individual failure modes via simulation or model-checking is computationally expensive and the inductive nature of the analysis (from causes to effects) creates difficulties, especially when combinations of failures need to be considered. Assuming that there are N possible component failures in a system, assessment of combinations of M of those failures requires that the analysis is repeated N!/((N-M)! x M!) times. For a system that has 1.000 failure modes, assessment of the effects of combinations of 2 failure modes requires that the analysis is repeated approximately half a million times. In HiP-HOPS, the analysis of propagation of failures is deductive (from effects to causes) and therefore the technique always synthesises fault trees in linear time not determined by the highest order cutset (i.e. the maximum number of failure modes considered in combination which is defined only by the positioning and nesting of AND gates in the error propagation model). The fast algorithms of HiP-HOPS have not only enabled its application on large systems but also its combination with computationally greedy heuristics such as Genetic Algorithms for the purpose of architectural optimisation with respect to dependability and cost [15] - a capability which is unique in HiP-HOPS. Moreover, HiP-HOPS has advanced capabilities for probabilistic analysis which include Poisson, Binomial and Weibull calculation models, as well as capabilities for common cause and zonal analyses, while most formal techniques tend to focus on functional safety analysis only. Clearly there is often a need in software design to consider the probability of failure of components.

6 Conclusions

In this paper we have presented how the architecture description language EAST-ADL2 supports the development of safety-critical automotive E/E systems. The integration of the safety case metamodel, safety analysis, and the system modelling will achieve several benefits. The Safety Case development will be eased by the systematic development that is supported by the EAST-ADL2, including structured information handling, support for reuse, consistency between the models, etc. The EAST-ADL2 language will benefit by having support for the Safety Case approach, an important technique in safety relevant system development. Further, the safety case metamodel will provide support for motivating why certain design decisions are needed and provide means for connecting the argumentation and design information. The connection between error modelling and system modelling supports quick safety design iterations, the creation of views, and structured information management.

We believe this approach presents an important step in making the design and safety processes more efficient and effective. Future work will concentrate on further evaluation of the approach, developing systematic support for integrating several relevant analysis techniques, and considering optimization with respect to safety properties. Another direction is to assess how the proposed approach, biased by automotive specifics and standards, is applicable to other domains.

Acknowledgments. This work was supported by contribution of all the partners of the ATESST project consortium funded by the European Commission. We wish to acknowledge feedback from the anonymous reviewers.

References

1. International Organization for Standardization: Draft 26262. ISO Committee (2008)
2. Chen, D.J., Törgren, M., Lönn, H.: Elicitation of relevant analysis and V&V techniques. D2.2.1. ATESST EC FP6 (2007), http://www.atesst.org
3. AUTOSAR Development Partnership, http://www.autosar.org
4. Kelly, T.P.: Arguing Safety - A Systematic Approach to Managing Safety Cases. PhD Thesis. University of York (1998)
5. 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)
6. Sangiovanni-Vincentelli, A., Di Natale, M.: Embedded System Design for Automotive Applications. IEEE Computer 40(10), 42–51 (2007)
7. HIS Members and Partners: Specification Requirements Interchange Format (RIF). v1.1a (2007), http://www.automotive-his.de
8. SysML Partners: Systems Modeling Language (SysML). Open Source Specification Project, http://www.sysml.org
9. Cuenot, P., Frey, P., Johansson, R., Lönn, H., Reiser, M.-O., Servat, D., Tavakoli Kolagari, R., Chen, D.J.: Developing Automotive Products Using the EAST-ADL2, an AUTOSAR Compliant Architecture Description Language. Ingéniurs de l'Automobile 793, 58–64 (2008)
10. 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. Technical Paper Series, 2008-01-0127. SAE (2008)
11. International Electrotechnical Commission: Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 0: Functional safety and IEC 61508 (2005)
12. Martin, T., Chen, D.J., Malvius, D., Axelsson, J.: Chapter - Model based development of automotive embedded systems. In: Navet, N., Simonot-Lion, F. (eds.) Automotive Embedded Systems Handbook. Industrial Information Technology. Taylor and Francis CRC Press, Abington (2008)
13. Arnold, A., Griffault, A., Point, G., Rauzy, A.: The Altarica formalism for describing concurrent systems. Fundamenta Informaticae 40, 109–124 (2000)
14. Bozzano, M., Villafiorita, A., et al.: ESACS: an integrated methodology for design and safety analysis of complex systems. In: ESREL European Safety and Reliability Conference, Balkema, pp. 237–245 (2003)
15. Papadopoulos, Y., Grante, C.: Evolving car designs using model-based automated safety analysis and optimization techniques. Journal of Systems and Software 76(1), 77–89 (2005)