Approach for Iterative Validation of Automotive Embedded Systems

Gereon Weiss, Marc Zeller, Dirk Eilers and Rudi Knorr

Fraunhofer Institute for Communication Systems ESK,
Hansastr. 32, 80686 Munich, Germany,
{gereon.weiss, marc.zeller, dirk.eilers, rudi.knorr}@esk.fraunhofer.de,
http://www.esk.fraunhofer.de

Abstract. Architecture description languages (ADLs) allow specifying system information in architecture models. These are generally used for capturing early design decisions concerning system or software development. Therefore, ADLs can be utilized for an early and iterative validation of the modelled system. With EAST-ADL an automotive-specific ADL is defined which allows describing an automotive system at different layers of abstraction targeting AUTOSAR systems. SystemC is an executable system modelling and simulation language which permits Hardware/Software-Co-Design. With the Transaction-Level Modeling (TLM) methodology the description of different layers of abstraction in SystemC is enabled. This work addresses the early validation of automobile electronic systems by providing a transformation of EAST-ADL models to SystemC at different layers of abstraction. This allows specific analysis with Hardware/Software Co-Simulation iteratively in the development process. The proposed approach is realized in a tool-chain and demonstrated by a typical automotive use case. Hence, we show the potential of an early validation of system and software designs based on architecture models.

1 Introduction

Model-driven design has been successfully introduced into diverse application areas for abstracting from complex systems. With Architecture Description Languages (ADLs) a solution is provided to capture design information on a high level of abstraction [1]. Various model-based tools exist which model the distinct behaviour of functions. ADLs allow to model the interaction of such functions on a system level describing the software and system architecture. An explicit modelling of layers of abstraction of the system provides the possibility to abstract from the system implementation at different points of view. Thus, special attention can be given to specific details of interest at the particular level. As architecture models include system information they can be used for a validation of the architecture even in early design phases. Approaches integrating well with the tool flow enable the validation in iterative steps as the system is refined. This permits early feedback to the development avoiding changes because of late identified design problems.

The automotive domain poses an area with complex interconnected embedded systems. Domain-specific modelling languages have been introduced to realize distinct functional behaviour. As this alleviates the development of single applications, with the more interacting functionality a more course-grained view on the overall system is needed. EAST-ADL [2] as a system level view for the automotive domain allows abstracting from the automotive electronic system at different levels. At implementation level the AUTomotive Open System ARchitecture (AUTOSAR) [3] meta-model is adopted enabling a well defined integration in present development methodologies. For the simulation and validation of hardware/software systems the system-modelling language SystemC [4] was developed. It incorporates a simulation kernel and structures for Hardware/Software-Co-Design. With Transaction-Level Modeling (TLM) [5] different levels of abstraction can be modelled in SystemC. Additionally, a subset of SystemC is synthesizable, e.g. for FPGA implementations.

Since the application of SystemC for a simulation-based validation of automotive electronic systems is a promising approach for design exploration and hardware sizing, its integration within the architecture design has to be pursued. Therefore, we introduce in this work an adoption of SystemC in the development process with architecture descriptions based on EAST-ADL. An automatic transformation on the different layers of abstraction of EAST-ADL to SystemC is presented in this paper which enables a simulation-based validation. Thereby, architecture models can be iteratively refined and improved in the development process.

This paper is structured as follows. In the next section related work to our approach is described. Afterwards, in Section 3 and 4 the concepts and main language elements of EAST-ADL and SystemC are presented. In Section 5 we introduce at first a mapping of the layers of abstraction of both languages. Subsequently, we detail the transformation of language artefacts of EAST-ADL to SystemC. A case study within the automotive domain shows the applicability of our approach in Section 6. This paper is concluded and an outlook on our future work is given in the last section.

2 Related Work

In this section we briefly describe related work to our approach with the focus on architecture descriptions and validation by simulation. The Architecture Analysis and Design Language (AADL) [6] was initially developed for the avionics domain but addresses generally the modelling of embedded real-time systems. It provides a textual and graphical notation for the architecture design of hardware and software components. Several approaches focus on the generation from AADL models for simulations [7] [8].

EAST-ADL (cp. Section 3) is a specific ADL for the automotive domain. It features defined layers of abstraction of the system and an orthogonal single environment model. The realization of the implementation layer of EAST-ADL is provided by AUTOSAR. As EAST-ADL was chosen as automotive ADL for

this work it is more comprehensively described below. AUTOSAR (Automotive Open System Architecture) [3] is a widely spread software architecture in the automotive domain. Instead of the traditionally ECU (Electronic Control Unit) centric development it focuses on the entire system and separates functionality from infrastructure. AUTOSAR provides well-defined interfaces for software components and layers of abstraction for hardware and infrastructure.

The Component Language (COLA) [9] is defined by formal syntax and semantics based upon synchronous dataflow. It also allows the hierarchical decomposition of the system. Even though it addresses the general modelling of embedded systems, it is evaluated for automotive case studies. A transformation to SystemC for an early design validation has also been carried out [10].

An approach to integrate virtual prototyping in the development process of vehicles is presented in [11]. In this work a mapping of *Automotive Open System Architecture (AUTOSAR)* [3] components to an equivalent SystemC model at different levels of granularity is outlined. Due to this mapping it is possible to co-simulate AUTOSAR-conform automotive software systems at different stages of the development process. In [12] and [13] a co-simulation approach for automotive embedded systems is described. The aim of this SystemC based approach is to enhance the diagnosis ability of the system. It integrates the functional model and the hardware specification with multi levels of granularity. In [14] and [15] a methodology for embedded systems based on SystemC TLM [5] is proposed. This methodology enables the rapid prototyping of embedded systems for functional validation and performance evaluation in early stages of the design process. Stepwise refinement of the system model allows the co-simulation at an untimed, cycle approximate or cycle accurate level.

3 EAST-ADL

EAST-ADL (Electronics Architecture and Software Technology - Architecture Description Language) [2] was initially developed and refined during several research projects as an automotive domain-specific ADL. Its main purpose is the model-based management of all engineering information in a single model. EAST-ADL is used at the design stage in the automotive domain. It is an architecture description language which supports different abstract views on an automotive electronic architecture. EAST-ADL integrates the component-based architecture of AUTOSAR [3], making it an AUTOSAR compliant architecture description language. The language is defined as a UML Profile [16] allowing a consistent description of the architecture with UML. The model of the complete system is separated into different layers of abstraction as shown in Figure 1.

The upper modelling layers provide an architecture independent system description which can be mapped to the AUTOSAR architecture description on the implementation layer. Orthogonal to the horizontal layers is the *Environment Model* as it exhibits no abstraction layers. It encapsulates plant models, i.e. models of the behaviour of the vehicle and its non-electronic systems. Func-

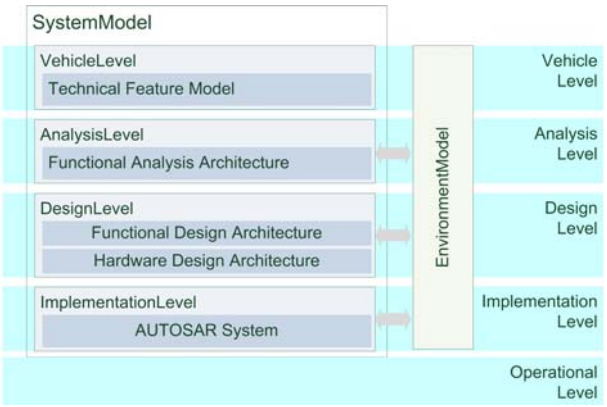

Fig. 1. EAST-ADL layers of abstraction with the orthogonal environment model [2]

tions in the Environment Model are connected with components representing hardware in the Analysis or Design Level by *ClampConnectors*.

Components communicate with each other through specializations of *FunctionPorts*. *FunctionFlowPorts* are inspired by SysML FlowPorts [17] and used for data flow-based communication. Additionally, for client-server interactions components can also interact through *FunctionClientServerPorts*. *FunctionPowerPorts* denote physical interactions between the environment and the sensing or actuation functions. *FunctionPorts* are typed by *EADataTypes* which represent data types in EAST-ADL, e.g. integer within a specifiable range as *EAInteger*. Hardware components communicate through specialized *HardwarePins*. *CommunicationHardwarePins* represent hardware connection points of communication buses. A *PowerHardwarePin* is used for modelling power supply. *IOHarwarePins* denote electrical connection points for digital or analog I/O.

At the most abstract layer, the *Vehicle Feature Level*, only features of the vehicle are modelled allowing the integration of product variability. Variability at all lower levels can be modelled through *VariationPoints*. Features can also be grouped and are realized by FunctionTypes. Diverse dependencies between features can be modelled as *VariabilityDependencyKind*. The focus of the *Analysis Level* lies on the modelling of the system in a way which is suitable for analysis. The architecture model at this level is called *Functional Analysis Architecture*. Components can be defined by *AnalysisFunctionTypes* and *FunctionalDevices* (the latter represent actuators and sensors on the Analysis Level). *AnalysisFunctionPrototypes* denote instances of these two. Software components are interconnected by *FunctionConnectors*.

At *Design Architecture Level* the software and hardware is represented in distinct models, the *Functional Design Architecture* and the *Hardware Design Architecture*. Software components are represented by *DesignFunctionTypes* or *LocalDeviceManagers* (the latter represent software interfaces to sensors and actuators). Hardware components are modelled by *Nodes* (ECUs), *Sensors*, *Ac-*

tuators and *LogicalBusses*. They are interconnected with *HardwareConnectors*. A *LogicalBus* represents the allocation target for *FunctionConnectors*, i.e. exchanged data in the Functional Design Architecture. Nodes are the allocation targets for *DesignFunctionTypes* and *LocalDeviceManagers*. *HardwareComponentPrototypes* are properties typed with the above mentioned hardware types representing instances.

On the transition from the Design Level to the *Implementation Level* a mapping to the AUTOSAR meta-model is foreseen. Therefore, modelling artefacts on this level are compliant to the AUTOSAR specification in version 3 [3]. The Operational Level refers to the deployed and running system and is not modelled for this reason. Behaviour is not explicitly considered in EAST-ADL. It can either be modelled externally (e.g. in domain-specific tools like Matlab or in a platform-specific programming language like C/C++) or internally in EAST-ADL utilizing UML behaviour modelling (like Activity Diagrams or Statecharts). As we have introduced EAST-ADL and its layers of abstraction, in the next section the language and a methodology to abstract different levels of SystemC are outlined.

4 SystemC - Transaction-Level Modeling

SystemC is a standardized system modelling and simulation language which supports Hardware/Software-Co-Design and Co-Simulation. It is standardised and promoted by the *Open SystemC Initiative (OSCI)* [18] and has been approved by the IEEE Standards Association as IEEE 1666-2005 [4]. Based on the wide-spread programming language C++, SystemC provides artefacts to simulate concurrent processes and an event-driven simulation kernel. Although, having semantic similarities to hardware description languages (like VHDL and Verilog), SystemC can be used to model the holistic system using plain C++.

A SystemC model usually consists of several modules (*sc_module*) which may be organized hierarchically. Computation in SystemC is modelled by so-called *processes* which are enclosed in modules. Processes are inherently concurrent. Communication from inside a module to the outside - mostly other modules - is realized via ports (*sc_port*). These are connected to channels (*sc_channel*) by SystemC interfaces (*sc_interface*). This enables the modelling of complex communication structures (e.g. FIFO or network bus) in SystemC.

With SystemC new models can be connected easily to existing hardware or functional models - either in platform-specific programming languages like C/C++ or domain-specific modelling tools, e.g. Matlab/Simulink within the automotive domain. Furthermore, it is possible to include any existing C or C++ library in the own system model. Thus, suppliers, for example within the automotive domain, can interchange pre-compiled hardware or software modules with other suppliers or car manufacturers and do not need to disclose their intellectual property.

To integrate Hardware/Software Co-Simulation effectively in the development process of networked embedded systems, a stepwise refinement of the mod-

els is necessary. This can be realized by using SystemC because it implements a top-down design process according to the *Transaction-Level Modeling* (TLM) [5] methodology. TLM is a methodology used for modelling digital systems which separates the details of communication among computational components from the details of the computational components. Details of communication or computation can be hidden in early stages of the design and added later. Therefore, communication mechanisms such as interconnection buses are modelled as *sc_channels* which can be accessed by *sc_modules* using SystemC interface classes. Transaction requests take place by calling interface functions of these channels. Low-level details of the communication process are encapsulated by the *sc_interfaces*. By this, the refinement of computation is separated from the refinement of communication. This approach enables the evaluation of different interconnection systems without having to re-implement the computation models that interact with any of the buses, because the computation models interact with the communication model through the common interfaces.

The OSCI Transaction Level Working Group has defined different levels of abstraction for TLM [19]. The most abstract level is denoted as *Communicating Processes* (CP). At this level the behaviour of the system is partitioned into parallel processes that exchange complex, high-level data structures through point-to-point connections. *Communicating Processes with Time* (CPT) is identical with CP, but introduces timing annotations. The next more detailed level, *Programmers View* (PV), is much more architecture-specific. Bus models are instantiated to act as transport mechanisms between the model components and some arbitration of the communication infrastructure is applied. *Programmers View with Time* (PVT) is functionally identical with PV but is annotated with more accurate timing information than CPT. At the level called *Cycle Callable* (CC) computation models are clocked and all timing annotations are accurate to the level of individual clock cycles. Communication models are fully protocol compliant. After introducing the abstraction levels of SystemC TLM we introduce in the next section our approach of mapping the architecture description of EAST-ADL to SystemC TLM.

5 Simulation-based Validation With Architecture Models

Architecture models capture information of the system development and allow a simulation-based validation. As described in Section 3 EAST-ADL enables the modelling of a system on different layers of abstraction. With TLM different levels for abstracting the modelled system are introduced in SystemC (cp. the previous Section 4). A combination of this ADL with the SystemC allows for an iterative design with early feedback on the models through Hardware/Software-Co-Simulation. For combining the advantages of an ADL and of simulations, the levels of abstraction have to be aligned. Thus, in the following we motivate a mapping of the levels of abstraction of EAST-ADL to the SystemC TLM levels as depicted in Figure 2.

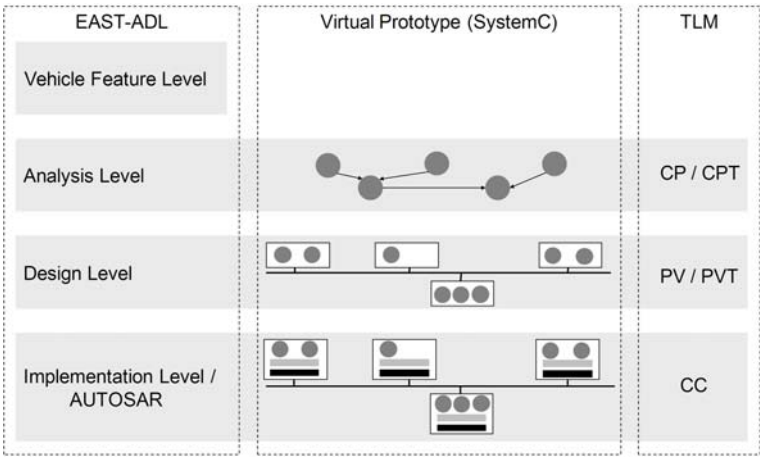

Fig. 2. EAST-ADL layers of abstraction in comparison to SystemC TLM levels

The most abstract functional levels in EAST-ADL and TLM are the Analysis Level and the CP. As the Vehicle Level only includes features it is not considered in this mapping for a simulation of the system behaviour. The TLM level Communicating Processes (CP) represents parallel processes which communicate via point-to-point connections. On the Analysis Level the inter-dependencies between the modelled functionalities and their externally visible behaviour are described. The functions of the Analysis Level represent abstract, communicating components. Thus, they can be transformed to parallel communicating processes and a mapping of the Analysis Level to the CP is possible. A specific transformation of the EAST-ADL component of this level is shown in the following section. As CPT adds timing annotation to the CP it corresponds to the Analysis Level with consideration of timing in the model.

On the next more detailed level are the EAST-ADL Design Level and the TLM PV. The Programmer's View (PV), additionally to the CP, incorporates bus architectures and arbitration of the communication infrastructure. Also, the Design Level introduces hardware models and bus infrastructure in EAST-ADL and refines the more abstract layer. Thus, the Design Level of EAST-ADL with its modelled software and hardware distributed on several ECUs may be mapped to the PV in SystemC. The software functions (*DesignFunctionTypes*) represent the processes and the hardware with interconnecting buses can be modelled as hardware architecture and corresponding communication infrastructure. Because the Programmers View with Time (PVT) includes more precise timing information than at CPT level, it is consistent with a model at the EAST-ADL Design Level with timing.

The most detailed functional abstraction layer in EAST-ADL is the Implementation Level. It is represented by AUTOSAR models and includes the most detailed and accurate system information. As this level is platform specific, rep-

resents a specific implementation and provides the necessary detailed information for a cycle-accurate simulation, it can be transformed to the TLM Cycle Callable Level (CC). The latter is cycle accurate with respect to individual clock cycles and thus well suited for simulating time-accurate AUTOSAR models. The aligned levels of abstraction as shown in Figure 2 include specific artefacts of the modelling languages which need to be transformed for a mapping. These are in the focus of the next subsection.

5.1 Mapping of EAST-ADL Artefacts to SystemC TLM

In the previous section we pointed out the general commonalities of the different levels of abstraction of EAST-ADL and SystemC. For an integration of a validation in SystemC the single artefacts of EAST-ADL have to be mapped to SystemC language elements. In the following we address such a mapping for the structural parts of the upper two functional levels of EAST-ADL the *Analysis Level* and the *Design Level*. As the *Implementation Level* is realized with AUTOSAR models, a mapping does not relate to EAST-ADL but to the AUTOSAR meta-model. Thus, a simulation with SystemC TLM can be realized by an approach based on AUTOSAR software components as presented in [11]. By this transformation particular emphasis has to be put on the semantic mapping and preservation of the defined artefacts. The structure of the model is preserved in the transformation. Thus, a tracing of the model artefacts to the SystemC code-level artefacts is possible, with the drawback of not optimized code. This also enables the feedback from the simulation to the respective model elements.

The following mapping focuses on the structural parts of the languages as the behaviour is not modelled explicitly in EAST-ADL (s. Section 3). Timing definitions have been integrated as non-funtional properties in the last version of EAST-ADL. As this relates to the non-functional property timing, it does not define the functional behaviour. An integration of the EAST-ADL timing semantics (Timing Augmented Description Language [20]) is currently in progress as future work. Only references to behaviour specified externally (e.g. UML behaviour or Matlab/Simulink Models) are included in the architecture model. Thus, the behaviour can be mapped directly to SystemC. For realizing the mapping of the behaviour off-the-shelf C/C++ code generators for the referenced behaviour can be used, e.g. TargetLink or UML Statechart generators. The generated platform code can be integrated as behaviour of SystemC Modules (`sc_modules`).

The orthogonal *Environment Model* can be represented by a single `sc_module` which includes the environment behaviour as sub modules, e.g. as code generated from a Matlab/Simulink model. `ClampConnectors` are specific elements for interfacing environment components with the horitontal Functional Analysis Architecture and the Functional Design Architecture. This connections can be realized with `sc_channels` and `sc_interfaces` in SystemC.

In the previous section the general relation of the Analysis Level of EAST-ADL and the CP level of TLM has been explained. For a specific mapping of these levels rules for transforming the EAST-ADL artefacts to SystemC elements have to be defined. As shown in Figure 3 EAST-ADL components can generally

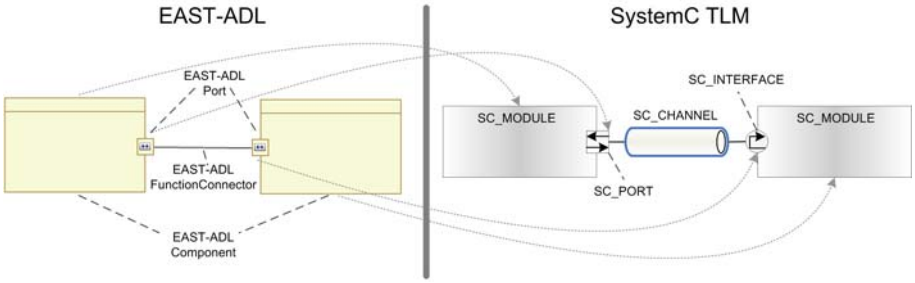

Fig. 3. Mapping of EAST-ADL artefacts to SystemC elements

be represented by `sc_modules`. The EAST-ADL ports and connectors can be transformed to `sc_ports`, `sc_channels` and `sc_interfaces`.

At Analysis Level sensors and actuators are represented by `FunctionalDevices`. `AnalysisFunctionTypes` are used to model functions. These components can directly be transformed to `sc_modules`. `FunctionFlowPorts` are ports for a data flows between AnalysisFunctionTypes. `FunctionClientServerPorts` can be utilized for function calls through a defined interface. Ports are interconnected by `FunctionConnectors`. The ports and connectors are transformed to simple `sc_signals` or `sc_channels` and `sc_interfaces` at CP level.

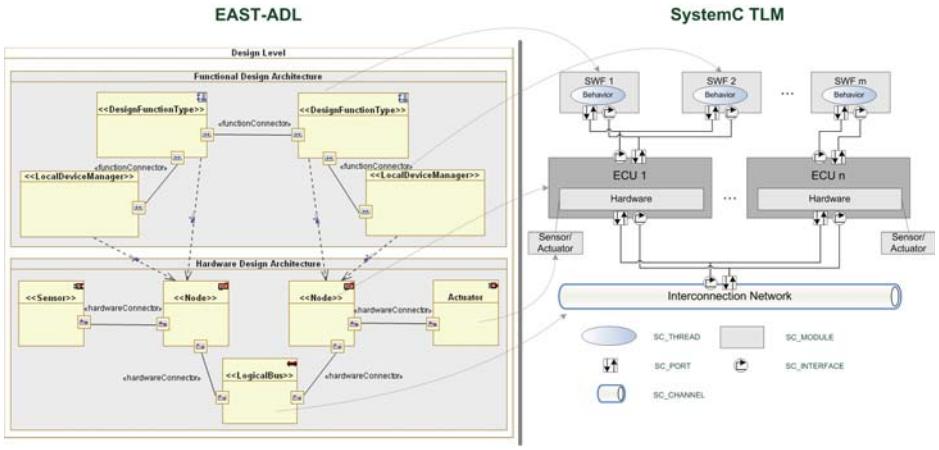

Fig. 4. Mapping of the EAST-ADL Design Level to SystemC PV simulation

At Design Level the simulation of software functions distributed over hardware platforms (ECUs) is addressed. Therefore, we introduce a SystemC-based framework which allows the modelling of automotive-specific elements in SystemC PV, like ECUs or software functions. As shown in Figure 4 the system is simulated in our approach at PV with software functions scheduled on

ECUs communicating via interconnecting buses. The respective components in SystemC are specialized `sc_modules`. For example, a DesignFunctionType is mapped to a software function (SWF) defined in the framework which is derived from a `sc_module`.

A general mapping of the EAST-ADL elements at Design Level to SystemC is depicted in Figure 4. The Design Level consists of software and hardware models. `DesignFunctionTypes` represent software functions. Software interacting with hardware sensors and actuators are modelled as `LocalDeviceManagers`. The hardware components at this level are `Sensors`, `Actuators` and `LocalBuses`. These components can be transformed to `sc_modules` at PV level as mentioned above. `HardwarePorts` and `HardwareConnectors` are transformed to `sc_channels` and `sc_interfaces` level. An extract of the main EAST-ADL

Table 1. Overview of the mapping of core elements of EAST-ADL onto SystemC TLM elements

EAST-ADL	**SystemC TLM**
AnalysisFunctionType	*sc_module*
FunctionalDevice	*sc_module*
FunctionConnector	*sc_channel,sc_interface*
DesignFunctionType	*sc_module*
LocalDeviceManager	*sc_module*
HardwareConnector	*sc_channel,sc_interface*
Node	*sc_module*
Sensor	*sc_module*
Actuator	*sc_module*
LogicalBus	*sc_module*
FunctionFlowPort	*sc_port*
FunctionClientServerPort	*sc_channel,sc_interface*
HardwarePort	*sc_port*

elements and their corresponding SystemC elements is given in Table 1. In the next section these transformation rules are applied in an automobile case study emphasizing their applicability for enabling an iterative validation within the development process based on an architecture model.

6 Case Study

The use of architecture models for a validation by simulation is evaluated in a case study for the previously described transformation of EAST-ADL to SystemC. Thereby, the focus of this work lies more on the structural mapping and generation of simulations than on the validation or analysis itself.

The afore introduced transformation of EAST-ADL models to executable SystemC models has been realized in a prototypical tool-chain. For evaluation purposes an automotive case study [21] has been modelled in EAST-ADL and

transformed to SystemC simulations. The use case is within the so-called body domain of an automobile and consists of the four features *exterior light*, *direction indication*, *central door locking* and *keyless door entry*.

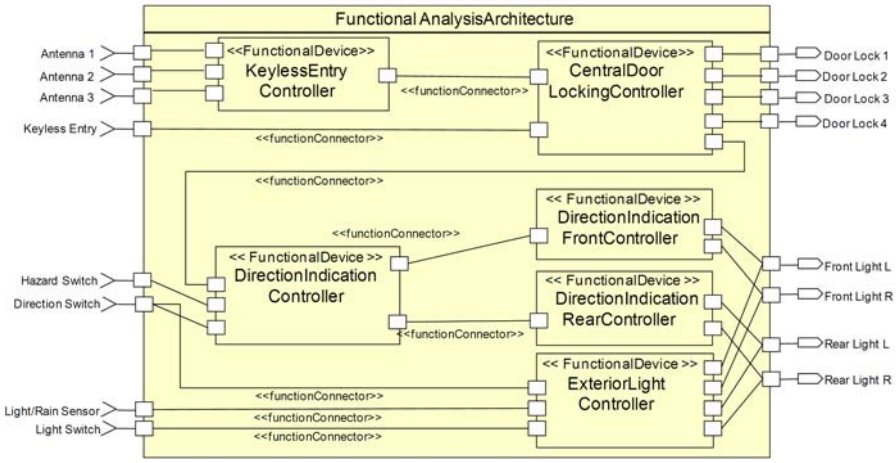

Fig. 5. Composite diagram of the use case at Analysis Level

The exterior light feature allows controlling the front and rear lights of the vehicle. The lights can be switched on/off manually or automatically through darkness or rain detected by the rain/light sensor. These inputs are interpreted by the function exterior light control which controls the light units (front and rear). For the direction indication a direction indication switch can be used to signal the turning direction. With the hazard light switch, risky driving situations can be signalled to other road users. Therefore, the direction indication master control informs the direction indication front and rear controls about the designated status of the direction indication lights. These turn the direction indication lights on or off in the front and rear light units. Central door locking allows locking and unlocking all doors simultaneously by using the key in the lock or by radio transmission. A radio receiver signals the information to the central door locking control. This function flashes the direction indication lights for a feedback to the driver and controls the four door locks of the car. An additional feature to the un-/locking of an automobile is the keyless entry. A driver can approach his car with the key in his pocket and the doors will unlock automatically. It can be locked by simply pressing a button on the door handle. Antenna components detect the key in the surrounding and inform the central door locking function which in turn unlocks the doors. With respect to the interaction with exterior light (which gives feedback via the direction indication lights), it does not make any difference whether the doors have been unlocked in a standard way or via the keyless entry. In the following a realization of these

features at Analysis Level and Design Level of EAST-ADL is described with its corresponding SystemC implementation.

At Analysis Level this use case is modelled in EAST-ADL by the FunctionalDevices `KeylessEntryController`, `CentralDoorLockingController`, `DirectionIndicationMasterController`, `DirectionIndicationFrontController`, `DirectionIndicationRearController` and `ExteriorLightController` as can be derived from Figure 5. The behaviour of these functionalities is described as opaque behaviour of the components (C++ source code). Additionally, behaviour can be modelled with straight-forward UML Statecharts providing a UML based behaviour specification. Communication is designed as data flow represented by `FunctionFlowPorts` and `FunctionConnectors`.

A SystemC simulation generated from this level includes modules interconnected for each of the above mentioned FunctionalDevices. They implement the respective behaviour of these modelled components in a thread of the module. With this transformation a simulation based on the Analysis Level in EAST-ADL of the use case was realized. Thus, the interaction of the abstract modelled functionalities can be validated with a simulation-based analysis.

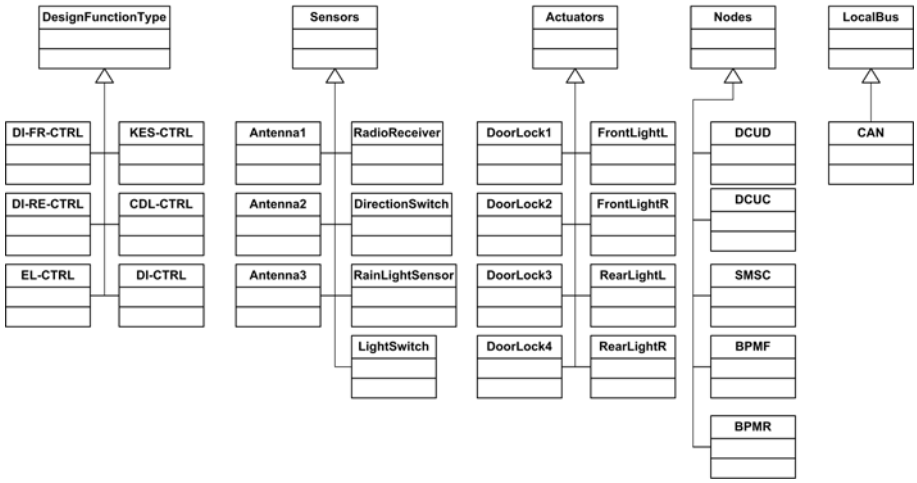

Fig. 6. EAST-ADL elements of the use case at Design Level

At Design Level the use case is modelled in a Functional Design Architecture (FDA) representing the software parts and a Hardware Design Architecture (HDA) representing the hardware parts of the use case realization. The FDA includes `DesignFunctionTypes` for the software functionalities of the use case and `LocalDeviceManagers` representing the software access to the modelled sensors and actuators. The latter are designed in the HDA together with the hardware platforms (`Nodes`) and the interconnecting `LocalBus`. Components in the FDA are interconnected with `FunctionConnectors` and in the HDA with

`HardwareConnectors`. An overview of the elements modelled at Design Level for the use case is shown in Figure 6. Additionally, `LocalDeviceManagers` exist for each depicted *Sensor* and *Actuator* in the Functional Design Architecture which are not explicitly displayed in this figure.

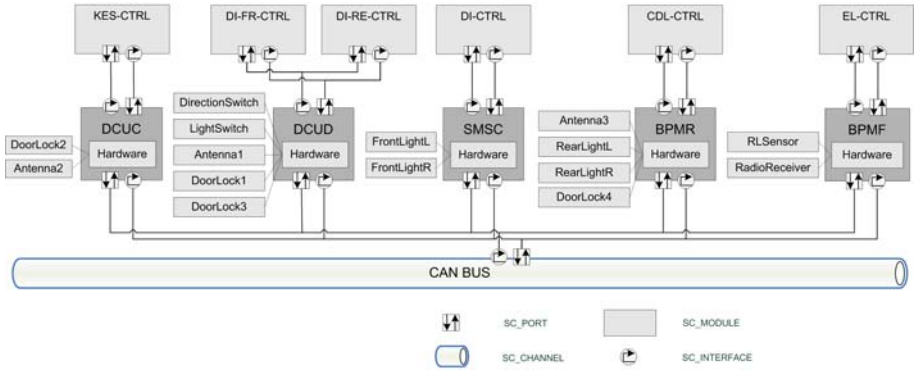

Fig. 7. Overview of the generated SystemC PVT use case at Design Level

The generated SystemC implementation of the use case at Design Level is depicted in Figure 7. It includes the use of a framework for automotive-specific modules. For example, ECUs and software functions can be included out of a library as specific `sc_module` implementations (cp. Section 5.1). As can be derived from Figure 7 the EAST-ADL Design Level components are generated as `sc_modules` representing software functions. These modules are included in another SystemC module which realizes a hardware platform with attached sensors and actuators in form of `sc_modules`. These hardware platforms are interconnected by a module implementation of the defined `LocalBus`. SystemC interfaces and channels realize the concrete interconnections of the modules. For example, a specialized type of `sc_interface` (*EcuSw_If*) realizes the communication between software functions and ECU modules.

The introduced transformation is realized in a prototypical *toolchain* which integrates into the Eclipse environment as a plug-in. By this, it can easily be used with EAST-ADL models based on UML in Eclipse (e.g. with the Papyrus UML modelling tool which supports EAST-ADL). The transformations itself are implemented as templates of the *Xpand* model-to-text transformation language. They use EAST-ADL models as input and generate the particular SystemC files with respect to the previously introduced mapping of the languages. Currently, simulations can be generated from the Analysis Level or Design Level. Simple checks allow to check the conformity for a simulation. Because a generation of incomplete models in early design stages should be possible, the checks are only as strict as needed for generating correct SystemC simulations. This supports the iterative simulation of ADL models in the design process. For the simulation at Design Level we utilize a self-developed framework (cp. Section 5.1) called

DynaSim which allows the modelling of an automotive in-vehicle network in SystemC. The generated files refer to SystemC models in the DynaSim library (e.g. ECUs or software functions). By this, a simulation can be performed considering the automotive-specific system environment. Future work will be the automatic feedback to the model as well as the integration and analysis of timing definition semantics.

7 Conclusion

Architecture Description Languages capture design information in architecture models. A simulation of these models in the development process allows an early validation. In this work we have briefly described the automotive-specific EAST-ADL and system modelling language SystemC. We showed that simulations from EAST-ADL can be generated automatically by a transformation to SystemC. Therefore, the EAST-ADL layers of abstraction were compared and mapped to corresponding layers of SystemC TLM. Also, transformation rules for the modelling artefacts of EAST-ADL with their concrete target elements in SystemC were presented. The approach was evaluated with an automobile case study with respect to the generation of simulations from EAST-ADL models on two different layers of abstraction. For this purpose a prototypical toolchain was built which allows the automatic generation of SystemC simulations from EAST-ADL models. By this, we showed that our approach allows the iterative simulation-based validation of automobile functions at different layers of abstraction.

In future work we plan to refine this approach by focusing on the simulation and preservation of non-functional requirements (e.g. timing) and integrating externally defined models. Additionally, more detailed automotive-specific SystemC models will be integrated in the simulation for a more precise analysis. A special emphasis will be taken on the design and simulation-based validation of adaptive embedded systems.

References

1. Medvidovic, N., Taylor, R.N.: A Classification and Comparison Framework for Software Architecture Description Languages. IEEE Transactions on Software Engineering **26**(1) (2000) 70–93
2. Cuenot, P., Frey, P., Johansson, R., Lönn, H., Reiser, M., Servat, D., Koligari, R., Chen, D.: Developing Automotive Products Using the EASTADL2, an AUTOSAR Compliant Architecture Description Language. In: Embedded Real-Time Software Conference, Toulouse, France. (2008)
3. Automotive Open Sytem Architecture (AUTOSAR): http://www.autosar.org
4. IEEE: IEEE Standard 1666-2005 - System C Language Reference Manual. (2005)
5. Cai, L., Gajski, D.: Transaction level modeling: an overview. In: Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software Codesign and system synthesis (CODES+ISSS '03). (2003) 19–24
6. SAE: Architecture Analysis and Design Language (AADL), document AS5506/1. http://www.sae.org/technical/standards/AS5506/1 (June 2006)

7. Chkouri, M.Y., Robert, A., Bozga, M., Sifakis, J.: Translating AADL into BIP - Application to the Verification of Real-Time Systems. In: Models in Software Engineering: Workshops and Symposia at MODELS 2008, Berlin, Heidelberg, Springer-Verlag (2009) 5–19
8. Varona-Gomez, R., Villar, E.: AADL Simulation and Performance Analysis in SystemC. In: Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, Los Alamitos, CA, USA, IEEE Computer Society (2009) 323–328
9. Kugele, S., Tautschnig, M., Bauer, A., Schallhart, C., Merenda, S., Haberl, W., Kühnel, C., Müller, F., Wang, Z., Wild, D., Rittmann, S., Wechs, M.: COLA – The component language. Technical Report TUM-I0714, Institut für Informatik, Technische Universität München (September 2007)
10. Wang, Z., Haberl, W., Kugele, S., Tautschnig, M.: Automatic generation of systemc models from component-based designs for early design validation and performance analysis. In: WOSP '08: Proceedings of the 7th international workshop on Software and performance, New York, NY, USA, ACM (2008) 139–144
11. Krause, M., Bringmann, O., Hergenhan, A., Tabanoglu, G., Rosentiel, W.: Timing simulation of interconnected AUTOSAR software-components. In: Proceedings of the conference on Design, Automation and Test in Europe (DATE '07). (2007) 474–479
12. Khlif, M., Shawky, M.: Enhancing Diagnosis Ability for Embedded Electronic Systems Using Co-Modeling. In: Proceedings of the International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering. (2007) 3–12
13. Khlif, M., Shawky, M.: Co-modelling and Simulation with Multilevel of Granularity for Real Time Electronic Systems Supervision. In: Proceedings of the 10th International Conference on Computer Modeling and Simulation (UKSIM 08). (2008) 348–353
14. Krause, M., Bringmann, O., Rosenstiel, W.: A SystemC-based Software and Communication Refinement Framework for Distributed Embedded Systems. In: Proceedings of the 13th Workshop on Synthesis And System Integration of Mixed Information Technologies. (2006)
15. Liang, L., Zhou, B., Zhou, X.G., Peng, C.L.: System Prototyping Based on SystemC Transaction-Level Modeling. In: Proceedings of the 1st International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06). (2006) 764–770
16. EAST-ADL2: Profile Specification 2.1 RC3. http://www.atesst.org/home/liblocal/docs/ATESST2_D4.1.1_EAST-ADL2-Specification_2010-06-02.pdf (June 2010)
17. Weilkiens, T.: Systems engineering with SysML/UML: modeling, analysis, design. Morgan Kaufmann (2007)
18. Open SystemC Initiative (OSCI): SystemC. http://www.systemc.org
19. Donlin, A.: Transaction level modeling: flows and use models. In: Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software Codesign and system synthesis(CODES+ISSS 04). (2004) 7580
20. TADL: Timing Augmented Description Language Version 2. http://timmo.org/pdf/D6_TIMMO_TADL_Version_2_v12.pdf (2009)
21. Hardung, B., Kölzow, T., Krüger, A.: Reuse of Software in Distributed Embedded Automotive Systems. Proceedings of the 4th ACM international conference on Embedded software (2004) 203 – 210