FROM EAST-ADL TO AUTOSAR

TAHIR NASEER QUreshi, DE-JIu CHEn, HENRIK LÖNN AND MARTIN TÖRNGREN

Stockholm 2011

Mechatronics Lab
Department of Machine Design
School of Industrial Technology and Management
Kungliga Tekniska Högskolan- The Royal Institute of Technology

TRITA-MMK 2011:12, ISSN 1400-1179, ISRN/KTH/MMK/R-11/12-SE
ABSTRACT
The increasing complexity of embedded systems has led to a multitude of efforts in the direction of component and model-based development. The use of formalized (or semi-formal) system descriptions enable (early) analysis and synthesis, reuse and also support structured information management. Several challenges however face industrial adoption. One of the challenges is the gap between models describing system requirements, functions and architecture at a higher level of abstraction (such as SysML), with respect to software/hardware architecture description languages (such as the AADL and the means for system description provided by AUTOSAR). The presented work addresses the gap between EAST-ADL and AUTOSAR. EAST-ADL is an architecture description language which provides an extension and profiling of SysML dedicated to automotive embedded systems. AUTOSAR provides means to describe software and hardware based architectures.

The contributions include a mapping between different concepts in the two languages along with possible variation points, identification of both existing and required tool features and methodological aspects by considering a specific tool chain. A discussion on the possible decision support is also a part of the outcomes of the presented work. Two case studies, i.e. a position control and a fuel control system, have been used to support the work, and a third case study of a brake-by-wire system has been used to validate the results. The findings provide a basis for automated refinement of EAST-ADL based architecture models to AUTOSAR configurations.

KEYWORDS:
# Table of Contents

Abstract ......................................................................................................................................................... 2

Keywords: ..................................................................................................................................................... 2

Table of Contents .......................................................................................................................................... 3

Introduction ................................................................................................................................................... 5

Industrial Needs and Solutions ..................................................................................................................... 6

AUTOSAR .................................................................................................................................................... 7

EAST-ADL ................................................................................................................................................... 8

Case-Studies and Methodology .................................................................................................................. 10

  Position Control System ............................................................................................................................... 10

  Fuel Control System ................................................................................................................................... 13

  Brake-by-Wire System ................................................................................................................................. 17

EAST-ADL and AUTOSAR Mapping Scheme ............................................................................................... 22

  Functional and Behavioral Mapping .......................................................................................................... 22

  Hardware Modeling and Function Allocation ............................................................................................ 25

Additional Observations .............................................................................................................................. 26

  Mode and mode behavior ............................................................................................................................. 26

  Mode Switch Port and Event ......................................................................................................................... 26

  Data Types and Prototypes ............................................................................................................................ 27

  Design Function Type Realization .............................................................................................................. 27

  Timing and Events ...................................................................................................................................... 27

  Tool Features ............................................................................................................................................ 28

Methodological Investigation ........................................................................................................................ 29

  Model transformation Investigation .......................................................................................................... 31

Related Work ............................................................................................................................................... 33

Discussion and Conclusion ......................................................................................................................... 33
INTRODUCTION

Embedded systems development complexity has increased considerably during the last few decades. This complexity spans over three dimensions i.e. product (features and interactions of components), organization (companies and development teams) and technology (tools and process). It is required to have not only the separation of hardware and software but also the consideration and analysis of execution constraints and other quality attributes such as safety and dependability. Several formalisms have been proposed to handle such complexities each targeting different abstraction levels and do not provide all the required analysis and information management support. Therefore, there exists a need to bridge the gaps between different formalisms to increase the development efficiency. This can include but not limited to an automated tool support for architecture refinement and synthesis and methodological guidelines.

We have addressed the above mentioned need by considering two different formalisms i.e. EAST-ADL [1] and AUTOSAR [2] for automotive embedded system from control systems development viewpoint. EAST-ADL shares the same meta-model and complements AUTOSAR with additional levels of abstractions and concepts such as requirements engineering and safety [1]. AUTOSAR can also be considered as an implementation alternative for an architecture specified by EAST-ADL. However, there still exist a gap between the two formalisms such as lack of detailed mappings between the artifacts of the two formalisms, tool integration, and lack of formal guidelines to work with EAST-ADL etc. For the presented work, the following needs are considered:

- A detailed mapping between the artifacts of the two formalisms for enabling transformation of an architecture specified by EAST-ADL to AUTOSAR.
- A relation between the EAST-ADL and AUTOSAR methodologies from behavioral point of view as a basis for formal guidelines in using the two formalisms together.
- Tool integration providing a support for transformation and formal guidelines to utilize the two formalisms and their methodologies.

The main contribution of the work is a mapping scheme between the artifacts of the two languages related to the software architecture part of a system. A methodological relation and tool integration scenarios for the integration of EAST-ADL and AUTOSAR are also identified. A primitive effort for automated transformation by evaluation of two different transformation possibilities and a suggestion for a possible extension of EAST-ADL are also performed as a foundation for the continuation of the work.

Due to large span of the two languages, we have limited our work mainly to behavioral aspects and associated execution constraints from control systems development view. In particular, we have performed two investigations. The first investigation is carried out to find how an AUTOSAR compliant architecture can be generated from an EAST-ADL model. The second investigation is carried out to find how the plug-in [3,4] for model transformation between Simulink and EAST-ADL models (developed using an EAST-ADL profile in PapyrusUML[5]) can be utilized for developing an AUTOSAR compliant system in conjunction with the AUTOSAR tool chain from dSpace [6]. Three case studies of a brake-by-wire, position control and a fuel control system are used as case studies to support the work.
**INDUSTRIAL NEEDS AND SOLUTIONS**

The development processes for many embedded systems to a great extent is a variation of the V-cycle [7]. The core development steps include requirement specification, high level design and analysis, detailed design and implementation, verification and validation. For the development of automotive control functions the traditional method follows a model-based approach similar to the one shown in Figure 1[7].

![Figure 1: An illustration of model-based development process for automotive systems](image)

As shown in the figure, the development process starts with the design of control functions or algorithms based on a set of given requirements. The initial design is a continuous time model of the controller and the plant (system to be controlled) which is then tested by rapid prototyping on a test platform. The result is a logical architecture with valid control algorithms. This is followed by the implementation of the software on real hardware platform and final integration with other devices in the vehicle. At the last stage, testing and calibrations are performed for fine tuning of controller parameters.

The model-based method described above has proven to be efficient for implementation of control applications on single ECUs (Electric Control Units) but problems related to timing, interface and communication occur during the integration of different systems [7]. In addition to the control system view, there exist other views such as safety, variability, requirement specification and compliance to standards which are required to be taken care of during the development. This results in increase of the overall complexity of the development process. The complexity has three dimensions i.e. product (related to system features, components and their interaction), organization (development teams and companies) and technology (tools, hardware platform etc.). A few of the current industrial challenges and needs are efficient management of information between different engineering teams, traceability between different development phases and design artifacts. Compliance with different regulations and standards to achieve increased safety and reliability and at the same time decrease cost and time etc. is also one of the challenges faced by the industry. In addition, due to longer life cycle of the whole vehicle and shorter life cycle of its components, it is also required to cope with the introduction of new components in the form of hardware or software.

Architectural description languages (ADL) and standardization are two possible solutions for the above mentioned problems and challenges. An ADL can be used for the specification of overall architecture of a system and the constraints including timing and behavior. SysML[8] and EAST-ADL are two examples of ADLs focusing on high levels of abstraction. SysML provides a sound basis for modeling requirements.
and system structure as well as behavior for generic systems and but needs to be tailored for modeling of domain specific artifacts. EAST-ADL on the other hand is developed specifically for the automotive industry. Standardization effort in the form of AUTOSAR is also being carried out for managing the ever growing complexity in automotive embedded systems. AUTOSAR and EAST-ADL are the main focus of this work and discussed in the following sections.

AUTOSAR

AUTOSAR (AUtomotive Open System ARchitecture) [2] is a standard for dealing with the ever growing complexity of automotive embedded systems. It provides a basis for modular software architecture with standardized interfaces and specifications of a run-time environment. Configuration management e.g. relocation of functionality from one computation node to another at development time is also supported by AUTOSAR standard. It is also possible to use commercial-off-the-shelf components and the components obtained from different suppliers.

![Interfaces and interfaces view (simplified)](image)

**Figure 2:** An overview of the AUTOSAR architecture

The result of the AUTOSAR effort is a framework and methodology [9] for standardized automotive software and hardware. Six-layered software architecture is a part of the AUTOSAR framework. The layers are Application, Run Time Environment (RTE) providing a middleware functionality, Service, ECU (Electronic Control Unit) Abstraction, Complex Drivers, and Micro-controller Abstraction [10]. While the ECU abstraction is the lowest layer, the Application is the highest one. These six layers

---

1 Figure extracted from the AUTOSAR website [2]
together comprise of more than 49 software modules ranging from modules for communication, drivers for different devices to services for memory and processor.

The AUTOSAR **atomic software components** are categorized into application and infrastructure. While the former is related to providing software functionalities such as control algorithm etc. the latter is used for different services related to an ECU. In addition a concept of virtual functional bus (VFB) is also introduced to resolve control related integration problems. Furthermore, AUTOSAR supports both client-server and sender-receiver kind of communication between its components.

As the work presented in this paper is focused on behavior, we would like to mention that the major components of the internal behavior of an AUTOSAR component are runnable entities (smallest code fragment provided by a component) capable or running under different modes, RTE events (e.g. reception of remote invocation, timeout etc.) exclusive areas (specifying constraints on scheduling policy) and inter-runnable variables. For more information, the readers are referred to the specifications available on the AUTOSAR website [2].

SystemDesk and TargetLink are the model-based development tools for AUTOSAR systems considered for this work. These tools are provided by dSpace [6]. SystemDesk is a standalone tool which is used for the specifications of the main architecture as well as integration of different parts of the system. It is complemented by TargetLink which is a Simulink based tool for modeling the behavior of the software components. The two tools can share the data with the help of the DataDictionary, a software which is capable of reading different files in the AUTOSAR XML format. For further description the readers are referred to the dSpace website [6].

**EAST-ADL**

EAST-ADL [11] is a domain specific architectural description language for automotive embedded systems which evolved from several European projects. The purpose of EAST-ADL is to specify automotive embedded systems at different levels of abstractions from different views, providing a domain specific approach for separation of concerns and information management between different engineering disciplines. The abstraction levels of EAST-ADL as shown in Figure 3 are:

- **Vehicle level** - the highest level covering vehicle features.
- **Analysis level** - related to the design of system functionality and interfaces e.g. control logics, algorithms and compositions. Physical resources like memory and processor are not considered at this level.
- **Design level** - takes into account the hardware architectures and the division and allocation of different functions to different hardware entities. It is further divided into Functional Design Architecture and Hardware Design Architecture referring to software architecture and hardware topology respectively.
- **Implementation level** - related to the specification of the actual hardware and software configuration. For automotive systems it corresponds to the AUTOSAR or similar specifications.
Independent extensions for modeling environment, behavior, dependability, and timing etc. are provided for modularity. These extensions are allowed at different levels of abstraction. This separates the definition of design function behaviors and the definition of related constraints for such behaviors like timing and safety.

In terms of EAST-ADL a behavior is “either a function performing some computation on provided data (FlowPort interaction) or the execution of a service called upon by another function (in a ClientServer interaction)” [11]. The execution can have either time- or event-based triggering. This means that a mapping from EAST-ADL behavior to AUTOSAR behavior (InternalBehavior) needs to consider not only the EAST-ADL behavior models, but also the related models describing the constraints in regards to timing, errors, communication, data and control properties (e.g., states and state-transitions) when mapping the EAST-ADL behavior definition to AUTOSAR InternalBehavior.

It is also possible to use external tools such as Simulink. The external models can be specified and linked by the behavior extension [13] of EAST-ADL. The EAST-ADL methodology [12] provides guidelines for the activities at all the abstraction levels. For more details the readers are referred to theATESST2 website [1] and the EAST-ADL specifications [11].

PapyrusUML is an eclipse based UML modeling tool. This tool which uses a UML profile for EAST-ADL has been the main tool for modeling systems by the ATESST project. It has also been incorporated in the standalone workbench together with the plug-ins for tools like Hip-Hops and Simulink. We consider the Simulink plug-in [3,4] for transformation of models from EAST-ADL to Simulink. The reason for choosing this plug-in for our work is the fact that Simulink can be used for specifying behavior of both EAST-ADL and AUTOSAR components.
CASE-STUDIES AND METHODOLOGY

A two-step approach is adopted to meet our objectives. First, a position control system is modelled in EAST-ADL using PapyrusUML modeller for basic structural and behavioural artefacts. This is followed by repeating the same procedure for a fuel control system for additional behavioural aspects especially mode related behaviour. As a second step, a brake-by-wire (BBW) system provided by Volvo Technology is modelled in SystemDesk to validate the results (i.e. refined mapping scheme between EAST-ADL and AUTOSAR) from the first step. The fuel control and position control systems are example cases from dSpace providing coverage of AUTOSAR artefacts sufficient to meet our objectives. The brake-by-wire (BBW) system has also been used in the ATESSST2 [1] and TIMMO [14] projects.

The three case studies are described below²:

POSITION CONTROL SYSTEM

The position control system comprises of the following three main components:

- Plant – represented a mass-spring-damper system whose position is to be controlled.
- Reference Generator – For generating reference values.
- Controller – Implementing a linearization algorithm and a PI algorithm to control the position of the motor.

While controller and the plant execute every 1ms, the reference generator updates the reference value every 100ms. The system consists of two ECUs namely plant and controller connected to each other via a controller area network (CAN). While the plant and reference generator are allocated to the former, the controller functionality is allocated to the controller ECU.

From AUTOSAR perspective, the plant and the reference generator comprises of one runnable each triggered periodically. The controller consists of two runnables; one for linearization triggered on the reception of sensor data and the other one implementing the PI controller.

From behavioral perspective, the internal behavior is specified using Simulink. The top level Simulink diagram is show in Figure 4. It can be seen from the figure that dSpace TargetLink blockset is used for the modeling purpose. The same blockset is also used for C code-generation. This code is directly incorporated as the internal behavior of each runnable.

² As the position and fuel control system are already documented in SystemDesk documentations, we will focus on the EAST-ADL implementation. The same applies for the brake-by-wire case for which we focus on SystemDesk implementation.
This case study was modeled using EAST-ADL at the design level of abstraction. The following three figures show the FDA, HDA and allocation views of the model.

**Figure 4:** Top level Simulink model for position control system

**Figure 5:** Functional design architecture for position control system
In Figure 8, implementation of two different types of triggering mechanisms and related timing constraints using EAST-ADL is illustrated. The classes with grey color represent the design function types in terms of EAST-ADL or a runnable with prefix “Run_” in the name. Artifacts belonging to the
behavior extension of EAST-ADL are colored in blue. A function trigger specifies the type of triggering for a particular function type. Artifacts with the green color represent the timing constraints on triggering. It should be noted that the artifacts from behavior and timing extensions do not point to each other. This implies that, one has to look for artifacts referring to a particular class to get complete information about both behavior and its constraints.

**Figure 8:** An illustration of periodic and data received triggers

**FUEL CONTROL SYSTEM**

The fuel rate controller is similar to the position control with an addition of mode control logic. It consists of the following three major software components:

- EngineModel representing the system being controlled. The main purpose of this component is closed-loop simulations.

- FuelsysSensors for detecting sensor failures and correct the signal value. The corrected value is send to the FuelSysController component. This software component also serves as the mode manager / provider. The composition of the FuelsysSensors modeled in EAST-ADL is shown in Figure 9:
• FuelsysController for calculating the fuel rate based on the corrected sensor readings from the FuelsysSensors. FuelsysController changes mode based on the mode provided by the FuelsysSensors. The composition of the FuelsysController as implemented in EAST-ADL is shown in Figure 10:

The triggering and mode disabling dependencies of the runnables are listed in the following table:
<table>
<thead>
<tr>
<th>Runnable</th>
<th>Description</th>
<th>Triggering</th>
<th>Mode Disabling Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run_AirflowCalculation</td>
<td>To calculate the intake airflow regardless of the currently active fuel mode</td>
<td>Periodic with 10ms period.</td>
<td>None</td>
</tr>
<tr>
<td>Run_FuelRateCalcNormal</td>
<td>To calculate the fuel rate for a normal low mixture</td>
<td>Periodic with 10ms period.</td>
<td>LowWarmup, Rich, or Disabled</td>
</tr>
<tr>
<td>Run_AirflowCorrection</td>
<td>To calculate the intake airflow correction for a normal low mixture</td>
<td>Periodic with 10ms period.</td>
<td>LowWarmup, Rich, or Disabled</td>
</tr>
<tr>
<td>Run_FuelRateCalcRich</td>
<td>To calculate the fuel rate for a rich mixture</td>
<td>Periodic with 10ms period.</td>
<td>LowNormal or Disabled</td>
</tr>
<tr>
<td>Run_OnEntryDisable</td>
<td>To set a failure counter if there is a faulty operating state and the system is running in the Disabled mode</td>
<td>Triggered a Mode switch event i.e. the entry of the Disabled mode.</td>
<td>None</td>
</tr>
</tbody>
</table>

The modes in the above table are as follows:

- The **LowWarmup** mode, to represent low emissions or a stoichiometric fuel mixture when the lambda sensor is in the warmup state after a cold start. This is the initial mode. It is also the active mode when a simulation is started.
- The **LowNormal** mode, to represent low emissions or a stoichiometric fuel mixture when the lambda sensor is in the normal operating state.
- The **Rich** mode, to represent an operating state using a rich fuel mixture.
- The **Disabled** mode, to represent a faulty operating state if more than one sensor failure occurs.

While the first two components i.e. FuelsysSensors and FuelsysController form a FureRateController composition, the last component is contained in EngineModel composition. The two compositions are allocated on two separate ECUs. The hardware design architecture in EAST-ADL corresponding to the hardware topology in AUTOSAR is shown in the following figure:
Figure 11: Hardware Design Architecture corresponding to the hardware topology of fuel control system

As the implementation of both position and fuel control systems is similar, the time triggered runnables are not shown in the report. In addition, the allocation of software components to the hardware is also omitted from the report for simplicity reasons.

For the fuel control system, we focus only on the mode related behavior which distinguishes it with the position control system. The following figures illustrate the implementation of mode behavior using the EAST-ADL behavior annex [13].

Figure 12: Mode and mode Group of fuel control system in EAST-ADL2
The above figure shows the behavior of running state (a combination of all the modes excluding the disabled mode). Similar to the position control system, the internal behavior of each runnable is modeled in Simulink. The Stateflow implementation corresponding to Figure 13 is shown in Figure 14:\(^3\):

![Diagram](image)

**Figure 14:** Mode behavior model in Stateflow

**BRAKE-BY-WIRE SYSTEM**

The brake-by-wire (BBW) system is an updated version of the case study developed in the ATESST2 [1] and TIMMO [14] projects. This model covers a larger number of EAST-ADL2 concepts. The updates

---

\(^3\) Note that the stateflow implementation is from the already available dSpace models.
include refinement with respect to the new language constructs, and functional allocation. However, the same Simulink model is used for our work which has been used in the ATESST2 and TIMMO projects. The different components of the system allocated to two different ECUs communicating via a CAN network have the following functionalities:

- Brake Pedal Sensor – To detect the pressure on brake pedal.
- Wheel Speed Sensor – To measure the angular velocity of the vehicle (rpm).
- Vehicle Speed Sensor – To measure the vehicle speed (Km/h).
- Brake Calculator – To measure the requested torque by the driver and send the value to the Brake Controller. (Period = 10ms)
- Brake Controller – To calculate the torque distribution for each wheel and send the value to the ABS of each wheel. (Periodicity = 20ms)
- ABS – To calculate the actual torque to be provided at each wheel based on the inputs from the Brake Controller. It sends an actuation signal to the brake actuator. (Period =10ms)
- Brake Actuator - The component responsible for the actuation of the physical brakes (Triggered on the reception of actuation signal from the ABS)

A simplified block diagram of the BBW system is shown in Figure 15. Note that “≤300” represents an end-to-end timing constraint i.e. the maximum delay from the sensing of drivers pedal to the final actuation.

![Simplified block diagram for Brake-by-Wire system](image)

**Figure 15:** Simplified block diagram for Brake-by-Wire system

The BBW model at the design level including the allocation to different ECUs is shown in Figure 16. The unconnected ports in the above model are for simulation purpose to be connected with the environment model.
The software architecture part of the system modeled in SystemDesk and a snapshot for the allocation of software components on one ECU are shown in Figure 17 and Figure 18. For our AUTOSAR implementation we assume a single runnable for each software component for simplicity. Figure 18 illustrates not only the allocation but also the relevant data elements and communication signals.
Figure 17: Brake-by-wire system software architecture in SystemDesk

<table>
<thead>
<tr>
<th>Component</th>
<th>Port</th>
<th>Interface</th>
<th>Data Element</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>VehicleSpeedSensor</td>
<td>VehicleSpeed</td>
<td>VehicleSpeed</td>
<td>VehicleVelocity</td>
<td>Tx</td>
</tr>
<tr>
<td>BrakeController</td>
<td>TorqueRequest</td>
<td>TorqueRequest</td>
<td>Torque</td>
<td>Tx</td>
</tr>
</tbody>
</table>

Figure 18: Component allocation for Central ECU of the brake-by-wire controller

The Simulink implementation of the brake-by-wire model is shown in the following figure.
The model is obtained from the TIMMO consortium and is used in the work without any changes.

Figure 19: Simulink model for Brake-by-Wire System\textsuperscript{4}
EAST-ADL AND AUTOSAR MAPPING SCHEME

The following mapping scheme is proposed based on the experiences from the case studies.

FUNCTIONAL AND BEHAVIORAL MAPPING

Tables 1 and 2 summarize the mapping between the functional and behavioural entities of EAST-ADL and AUTOSAR. For semantics the readers are referred to the EAST-ADL and AUTOSAR specifications.

Table 1. A functional (structural) mapping between EAST-ADL and AUTOSAR

<table>
<thead>
<tr>
<th>EAST-ADL</th>
<th>AUTOSAR</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>FunctionalDesignArchitecture</td>
<td>Software architecture</td>
<td>It is assumed that a design function type with name “FunctionalDesignArchitecture” is the top most function in the hierarchy of design function types and prototypes.</td>
</tr>
<tr>
<td>FunctionModeling::DesignFunctionType</td>
<td>Runnable and Atomic Software Component</td>
<td>The property isElementary determines if a design function prototype (or several) is conveniently realized by a runnable. An atomic software component contains at least one runnable through its internal behaviour definition. isElementary = true implies a single runnable can be used. isElementary = false implies an atomic software component with one or more Runnables in its InternalBehavior or a composite component.</td>
</tr>
<tr>
<td>FunctionModeling::BasicSoftwareFunctionType</td>
<td>Basic Software Component</td>
<td>A basic software function type is a specialization of DesignFunctionType and a middleware abstraction. This corresponds to basic software component in AUTOSAR</td>
</tr>
<tr>
<td>FunctionModeling::LocalDeviceManager</td>
<td>Sensor Actuator Software Component</td>
<td>The LocalDeviceManager encapsulates the device-specific or functional parts of a Sensor or Actuator, device, interface etc.</td>
</tr>
<tr>
<td>FunctionModeling::FunctionFlowPort</td>
<td>A port with a Sender/Receiver interface or an interrunnable variable.</td>
<td>The direction of the EAST-ADL flow port determines if the corresponding AUTOSAR port has a provided or required interface. direction = OUT corresponds to provided port direction = IN corresponds to a required port. If the associated EAST-ADL DesignFunctionType is realized as an AUTOSAR runnable then the port is realized as an inter-runnable variable.</td>
</tr>
<tr>
<td>FunctionModeling::FunctionClientServerPort</td>
<td>A port with Client or Server Interface</td>
<td>The role as a client or server is defined by the property ClientServerType. This corresponds to a client and server interface respectively in AUTOSAR</td>
</tr>
<tr>
<td>FunctionModeling::ClientServerInterface</td>
<td>Client-Server Interface</td>
<td>Same Concept</td>
</tr>
<tr>
<td>EAST-ADL</td>
<td>AUTOSAR</td>
<td>Remarks</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FunctionModeling::Operation</td>
<td>Operation</td>
<td>Same concept and related to a client-server interface.</td>
</tr>
</tbody>
</table>

AUTOSAR software components and runnables mapped at EAST-ADL are illustrated for fuel control system case study in

![Figure 20](image-url)

Figure 20.
Figure 20: An atomic software component with its runnable modeled in EAST-ADL2

Table 2. Mapping of EAST-ADL artifacts corresponding to AUTOSAR behavior

<table>
<thead>
<tr>
<th>EAST-ADL</th>
<th>AUTOSAR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavior::Mode</td>
<td>Mode</td>
<td>The concept is similar in both formalisms. A mode control can lead to the switches between different configurations or execution schemes</td>
</tr>
<tr>
<td>Behavior::ModeGroup</td>
<td>Mode Group</td>
<td>Same concept in both formalisms to organize a set of modes in a mutually exclusive group.</td>
</tr>
<tr>
<td>Behavior::FunctionTrigger</td>
<td>RTE Event</td>
<td>The execution behaviour of an EAST-ADL function is declared by function triggers. The type of corresponding AUTOSAR RTE Event is determined by the TriggerPolicy and the associated Event Function and constraint for the function trigger. TriggerPolicy=Time implies a periodic event. (Timing event in AUTOSAR) TriggerPolicy=Event implies other events.</td>
</tr>
<tr>
<td>Timing::EventFunctionClientServerPort</td>
<td>Op Inv</td>
<td>An event function is associated with a function type in EAST-ADL. The type of event function determines the type of RTE Event in AUTOSAR. eventKind = receiveRequest implies Operation Invoked Event eventKind = receivedResponse implies Asynchronous</td>
</tr>
<tr>
<td></td>
<td>Event and Asynchronous Server Call Returns</td>
<td></td>
</tr>
</tbody>
</table>
The mappings are further illustrated in Figure 21

**Figure 21**: An illustration of EAST-ADL and AUTOSAR mapping

**HARDWARE MODELING AND FUNCTION ALLOCATION**

Hardware modeling is similar to functional modeling in EAST-ADL2. A hardware design architecture in EAST-ADL2 corresponds to a hardware topology in AUTOSAR. This is shown in Figure 6 for the position control system which comprises of two ECUs and a CAN (Controller Area Network) bus. The plant ECU is used for the sensing and actuating functions while the controller ECU is for the main controller.

The allocation of different functions is done by using the “FunctionAllocation” entity in EAST-ADL2 which corresponds to software to ECU mapping in AUTOSAR. A summary of mapping hardware topology and function allocation is shown in Table 3.

**Table 3**: Mapping of hardware entities

<table>
<thead>
<tr>
<th>EAST-ADL2</th>
<th>AUTOSAR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FunctionModeling::FunctionAllocation</td>
<td>SwcToEcuMapping</td>
<td>This mapping implies that all the AUTOSAR runnables belonging to an atomic software component should be mapped to the same ECU.</td>
</tr>
<tr>
<td>EAST-ADL2</td>
<td>AUTOSAR</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>HardwareModeling::LogicalBus</td>
<td>Bus</td>
<td>Same concept</td>
</tr>
<tr>
<td>HardwareModeling::Node</td>
<td>ECU</td>
<td>Same concept</td>
</tr>
</tbody>
</table>

**ADDITIONAL OBSERVATIONS**

The following text will throw light on the factors which may affect the decisions required for refinement of software architecture from EAST-ADL to AUTOSAR.

**MODE AND MODE BEHAVIOR**

Mode is one of the fundamental behavior entities. It is used to represent different configurations of a system. Two or more modes can be organized in a “Mode Group” as mutually exclusive entities. Both AUTOSAR and EAST-ADL2 have the same concept of mode and mode group. A design function type in EAST-ADL2 model or a runnable in AUTOSAR can be active or disabled based on the selection of mode. This will be discussed in the next subsection on function trigger. A model of mode and mode group for the fuel control system is shown in Figure 12.

As mentioned earlier, behavior can be modeled using external tools or by using behavior extensions [13]. The behavior extension can be used to model state based behavior of a component or in AUTOSAR terms a dedicated mode manager. An illustration of the use of the extension is shown in Figure 13. For simplicity, the figure only shows one transition in detail i.e. its properties, related parameter and parameter (guard in terms of state machine) condition in detail. The initial mode in a mode group can be identified form the property ‘init_State’ of a state. As shown in figure the state ‘Warmup’ is the initial state for the modeled behavior.

**MODE SWITCH PORT AND EVENT**

Modes are declarative in EAST-ADL. Mode change and related communications e.g. mode switch event are not part of the EAST-ADL specifications. To handle this we can assume an EAST-ADL FunctionFlowPort to be equivalent to a mode switch port provided that it is referred in the function trigger properties in addition to the event condition for activation.

![Figure 22: A function trigger illustration in EAST-ADL](image)
An illustration is shown in the left part of Figure 22 for a runnable which is activated on the entry of the mode called ‘Disabled’. The policy selected for this trigger is ‘EVENT’. The right side of Figure 22 refers to a periodic trigger with the value ‘TIME’ selected for the property ‘TriggerPolicy’. Here the mode property specifies the modes during which this trigger is active.

**DATA TYPES AND PROTOTYPES**

EAST-ADL only supports basic data types including Boolean, float, integer, string as well as composite data types. As a part of defining an AUTOSAR software architecture, these abstract data types are required to be mapped to concrete implementation data types with signedness, number of bits, coding, etc. Furthermore, in contrast with AUTOSAR, only one data type can be assigned to a single port in EAST-ADL. This is handled by the use of port groups. Depending on the type of realization i.e. an atomic software component or a runnable, a port specifies an inter (RTE) or intra (Variables) software interaction. This is described in Table 1 and illustrated in Figure 1. It should also be noted that there is a DataElementPrototype for every EAST-ADL port. The data types should have matching specifications. Several EAST-ADL ports may be aggregated in a single AUTOSAR interface. EAST-ADL port groups are candidates for aggregation depending on connection patterns.

**DESIGN FUNCTION TYPE REALIZATION**

For simplicity we proposed a one-to-one mapping between a design function type of EAST-ADL and an AUTOSAR runnable or software component. However, it is a n-to-m mapping as shown in Figure 1. This implies that several design function types can be realized by one runnable or vice versa. The same applies for the realization of design function type by AUTOSAR atomic software components.

**TIMING AND EVENTS**

As mentioned in the last subsection that the type of an event is determined by an event function and the associated constraints. This is illustrated by the five classes on the left of Figure 8 for a periodic event which corresponds to a timing event of 1ms for a runnable named ‘Run_DetectSensorFailures’ in AUTOSAR. It can be noted from the function trigger in Figure 23 that the runnable is active in all the four modes shown in Figure 12 as compared to the runnable on the right side of Figure 22 which is only active in two modes.

---

5 The string ‘ModeSwitchEvent’ for the trigger condition in the figure is for illustration purpose only. This means that other constructs e.g. OCL (Object Constraint Language) can also be used depending on the user choice.
TOOL FEATURES

As mentioned earlier, the presented work is based on a specific tool chain. The tool chain comprises of AUTOSAR tools from dSpace, PapyrusUML for modeling EAST-ADL, Simulink for behavior modeling of both EAST-ADL and AUTOSAR components, and Simulink plug-in for the transformation of the models between PapyrusUML and Simulink. There exist two possible links between EAST-ADL and AUTOSAR as shown in Figure 24.

The first link is through Simulink for behavior specification and C/C++ code generation for an AUTOSAR runnable. The second link is through model transformation from EAST-ADL to AUTOSAR software architecture. The following are the results from the investigations.

- The dSpace tool chain provides complete support for modeling an AUTOSAR software architecture, behavior modeling and code generation for atomic software components.
The current Simulink plug-in only supports basic subsystems and their ports from the Simulink library. It also lacks support for integrating Simulink subsystems as Function Behavior. This means that only a structural transformation is possible where the Simulink model (i.e. behavior specification) is abandoned after import to EAST-ADL. This limits the use of Simulink.

There also exists a need for a model transformation tool which can transform models from EAST-ADL to AUTOSAR. For this purpose the mapping scheme presented in this paper can be used. Specifically for the tool chain considered for this work, a transformation tool to transform models from PayrusUML and SystemDesk is required. An important feature of the required tool is the traceability of information. This is due the fact that EAST-ADL artifacts corresponding to the internal behavior of an AUTOSAR software component are obtained from different extensions especially behavior and timing.

METHODOLOGICAL INVESTIGATION

EAST-ADL can support both top-down and bottom-up approach for designing a system. Figure 25 shows one such approach in the form a UML activity diagram.

The decision points marked from 1-5 are dependent on the application type. For our considered perspective i.e. control systems development and behavior modeling using the available tool chain, the decision points corresponds to the following:

1 - All the features and requirements are specified and no requirement is left unsatisfied. This means that every requirement should have at least one EAST-ADL "satisfy" link connecting it to a feature.

2 - There is a need to update the analysis model based on the results from the analysis carried out using Simulink. Addition of new function types, change in the environment model are a couple of examples of the possible updates. The major reason can be a control algorithm which is not valid (Compare with the transition between the second and third stages in Figure 1). A few of the reasons for this update are need for a more accurate environment model, basic timing requirements such as sampling or execution time does not satisfy a required behavior constraint.

3 - It is required to update or redefine features and high level (non-technical) requirements. This may be necessary if it is not possible to fulfill any analysis level requirement. The updates may include omission of a feature, addition of new requirements etc.

4 - If there is a need to update the design model based on the results from the analysis carried out using Simulink. Addition / removal of design functions or timing requirements on the execution of a function based on the emergent behavior are a couple of example of the updates possible at design level. The reasons can be similar those in analysis level, change in characteristics of the hardware platform e.g. processor speed or non-schedulability of components etc.

5 - If it is required to update or redefine analysis level requirements and functions. This may be necessary if the emerging behavior based on the available resources e.g. processor and memory
does not satisfy the one or many vital design requirements derived from the analysis requirements.

Figure 25: Illustration of a possible combined EAST-ADL and AUTOSAR methodology. While the actual methodology specifies only 'Analysis' we have used action 'Transformation and Analysis'. The reason is the fact that EAST-ADL relies on external tools for analysis. For behavior the
activities included in this action are shown in part c of the figure where the first and second model transformations correspond to the export from EAST-ADL model to Simulink and vice versa.

In part d of the figure, we have shown the actions required to generate AUTOSAR specific behavior (through code generation from Simulink using TargetLink) and the rest of the architecture (using model transformation to SystemDesk) from EAST-ADL. The action 'AUTOSAR System Configuration' refers to the modeling and specification carried out using SystemDesk or similar tools. These activities are completely aligned with the top-down approach used for the dSpace tools where the initial configuration is obtained from the EAST-ADL. The transformation from EAST-ADL to System Desk can be either model-to-model (M2M) or model-to-text (M2T). In case of former an approach similar to [15] can be adopted. In the latter case a python script which can run on SystemDesk can be generated directly from the EAST-ADL design level model.

The 'Implementation' and 'Verification, Validation and Testing' activities are out of scope of the presented work. Part b,e and f of the figure illustrate simplified steps of the EAST-ADL methodology. However, we would like to mention that the 'Implementation' action is the phase where the AUTOSAR methodology [9] can be applied. For the verification, validation (V&V) and testing phase, the V&V cases (usually defined with the requirements at each phase of EAST-ADL using Verification and Validation extension) can be used.

The following text describes how the existing design-flow shown in Figure 1 can be aligned with EAST-ADL.

- The first two steps of Figure 1 corresponds to the 'Analysis Level Modeling' and 'Transformation and Analysis' actions of Figure 25a. In order to align the current practice with EAST-ADL, it is required to do the higher level modeling in EAST-ADL and then perform the analysis. Rapid prototyping is also a part of analysis.

- In contrast to the design process shown in Figure 1, the integration is a multiple steps process instead of a single step. First a basic integration is carried out at the 'Design Level Modeling' which gives an advantage of an early analysis, verification and validation. The second integration is carried out during 'AUTOSAR System Configuration' where the actual code generated from Simulink is integrated with RTE code.

- The last two actions of Figure 25a correspond to a part of the fourth and complete last steps in Figure 1.

MODEL TRANSFORMATION INVESTIGATION

Two different approaches for model transformation from EAST-ADL2 to AUTOSAR were evaluated. The first approach was based only on the existing EAST-ADL2 constructs and similar to the ARGateway in the EDONA project [16] to the extent of modeling basic AUTOSAR artifacts in UML. A light-weighted AUTOSAR profile is developed in PapyrusUML. The reason for choosing this tool is its support for EAST-ADL2. With the main focus on the software architecture and hardware topology, the profile mainly includes data, structure and behavior. To avoid duplication and feasibility reasons, features
available in the standard tools, the behavioral aspects such as wait points, exclusive areas etc. are not covered in the profile.

In the above mentioned approach, an AUTOSAR configuration is developed first using the profile and adding realization links between AUTOSAR artifacts and the EAST-ADL artifacts realized. This approach is simpler in the sense that the variation points such as the many-to-many relation between and EAST-ADL design function type and AUTOSAR software component and runnables.

The second approach is based on the direct transformation from EAST-ADL2 “Design Level” to AUTOSAR. A Python script is generated from the AUTOSAR model using model-to-text transformation. For this purpose Acceleo\(^6\) which is an implementation of OMGs MOF Model-to-text Transformation Language is used. This approach requires extra effort to model AUTOSAR components. The following code illustrates the generation of a software composition from a Design Function Type.

**Listing 1:** Code for generating software composition from a design function type

```plaintext
# Create a software architecture.
softwareArchitecture = MyNewProject.SoftwareArchitectures.Item("SoftwareArchitecture")
if not softwareArchitecture:
    softwareArchitecture = MyNewProject.SoftwareArchitectures.Add("SoftwareArchitecture")

    # Create a composition with a port.
    MyComposition = softwareArchitecture.Compositions.Add("[c.name]")
    for (a:Property |c.attribute|
        MyComposition.Ports.Add("[a.name]")
    
    # Create a Design Function Type.
    DesignFunctionType = MyNewProject/designFunctionType/
    DesignFunctionType = DesignFunctionType.Add("ABSController")
    DesignFunctionType = DesignFunctionType.Add("+1InputData: SensorReading [1]
        + PpDesiredActuation: DesiredActuation [1]"
```

The above code is only valid for the condition when a composition does not have any atomic software component and the assumption that a Design Function Type corresponds only to an AUTOSAR composition. A design function type and the resulting composition are shown in Figure 26.

![Figure 26: Model transformation illustration for an ABS controller of the Brake-by-wire System](image)

From the above work, it can be concluded that the artifacts like hardware topology can also be generated with the dSpace commands such as hardTopology.Ecus.Add("ECUName") for adding an ECU, mapping

\[^6\text{http://eclipse.org/acceleo/}\]
to nodes to ECUs etc. Some experiences from the plug-in work here to show that it will work and increase the confidence level of the transformations.

RELATED WORK

A few efforts have been made to develop AUTOSAR system specification from different ADLs. In [17] a mapping between different EAST-ADL artefacts and AUTOSAR is presented for both structure and behaviour. However, the mapping focuses on only a few structural entities and a few events. Furthermore, due to the changes in the language regarding the inclusion of concepts such as mode, restructuring for modularity and renaming of a few artefacts, some parts of the mapping are no longer valid. Our work aims to provide a more detailed and refined mapping scheme with the current version of EAST-ADL.

In the EDONA project [16] an Eclipse based platform is developed for the integration of different tools used in the automotive industry. The approach is based on an AUTOSAR meta-model developed using EMF (Eclipse Modeling Framework). The ARGateway (AUTOSAR Gateway) is used as a means for model transformation from EAST-ADL design architecture model to an AUTOSAR model. The gateway is mainly based on the mapping provided in [17]. Therefore, it needs to be updated. The required updates can utilize the results of the work presented in this paper.

In [15] bidirectional transformations between SysML and AUTOSAR using graph transformations are presented. The transformation uses a SysML profile covering basic embedded system entities such as hardware, software, port etc. and an AUTOSAR meta-model. SystemDesk API is used to transfer information to and from SystemDesk. The SysML profile in [15] is too generic and cover only a few automotive systems aspect as compared to EAST-ADL. SystemDesk is the common tool in [15] and our work. Therefore, a tool specific transformation can be performed by utilizing the results of the presented work and the SystemDesk API part from [15].

In the TIMMO [14] project, a language for timing design called TADL (Timing Augmented Description Language) is developed. The language and its methodology [18] are aligned with EAST-ADL and AUTOSAR. Especially the timing aspects of AUTOSAR are actually fully harmonized with the TIMMO concepts. As the name indicates, the TIMMO methodology is focused on timing analysis aspects. In contrast to the TIMMO effort focused on timing aspects, our effort focuses the behavioural aspects.

DISCUSSION AND CONCLUSION

We have investigated the support of EAST-ADL for AUTOSAR based system definition as a means to capture the early phase information and provide abstract representation of system. The results of the work include an investigation of the mapping between the two approaches. The work mainly covers the artifacts related to software architecture. A few tooling issues and methodological aspects are also investigated. Although the work is focusing specific tool chain, the mapping scheme is generic addressing both the updated and new concepts of EAST-ADL, therefore, it can be used for any AUTOSAR tool. We envision a tool support i.e. a complete and automated transformations. These transformations can be between Simulink and EAST-ADL for behavior specifications or between EAST-ADL to AUTOSAR for structural transformation. For this there is a need to automate design decisions such as realization of design function types to runnables or atomic software components. An automated tooling is also
necessary to manually trace different artifacts e.g. a periodic constraint traced to a design function type etc. From methodological perspective, EAST-ADL is an add-on to the current development process providing early verification and validation resulting in reduction of the integration problems.

There exist a few pertaining issues. The first issue is the improvement of current tools and the need for new tools. For example, the tool support for transforming EAST-ADL models into an AUTOSAR architecture model. The only tool for this purpose is the one developed in the EDONA project [16] which needs to be updated with the new EAST-ADL concepts and with more flexible mapping strategies, including manual intervention or use of constraints that control the mapping. For transformation, the mapping proposed in this paper can be used. Similarly, the current Simulink plug-in only provides limited support which is not sufficient for a coherent coupling or model continuity [19] with the available tool chain.

Another issue is the choice between the extension of EAST-ADL behavior annex [13] for direct code generation for runnables and relying on the code generation capabilities of tools like Simulink. The former approach will require new tools and extension of the language and the other option will require the improvement of the tool plug-ins. Both the options are non-trivial.

The presented work can be extended by providing the required tool support for automated model transformations between EAST-ADL and AUTOSAR tools, improvement of the existing Simulink plug-in for behavior modeling and improved integration with the available tools. The possibility of the mapping of runnables to OS (Operating System) tasks and relation with other analysis tool for the type of analyses not supported by Simulink also needs to be addressed in future. In addition to the presented work or the TIMMO effort which focus on behavior and timing respectively, a detailed methodology taking into account all the available aspects e.g. behavior, timing, safety and variability is also required.

REFERENCES


