

An Integrated Approach for Modeling, Analysis and Optimization of Systems whose Design Follows the EAST-ADL2/AUTOSAR Methodology

Ernest Wozniak, Sara Tucci-Piergiovanni, Chokri Mraidha and Sebastien Gerard
CEA LIST

ABSTRACT

Clear and comprehensive view on an automotive system design, as well as its analysis/optimization, are the key research topics in the automotive domain. To tackle the first problem, modeling languages such as the EAST-ADL2 or the AUTOSAR metamodel, have been defined. The second issue has found its solution in the numerous analysis and optimization techniques. Nevertheless consideration of these two problems simultaneously is not the case. This has resulted in many tools that deal either with modeling or analysis and optimization. As a consequence it is difficult to embed these tools in the standard methodologies such as the EAST-ADL2/ AUTOSAR methodology. This especially applies to the higher abstraction layers as those defined by the EAST-ADL2. It is a consequence of a high dependability of analysis/optimization tools on the platform specific information. Therefore this paper presents UML based framework that combines the modeling of the automotive architectures and support for an optimization at the early stage of a system design. Optimization technique uses genetic algorithms and the early stage optimization goals concern performance, safety and cost. We show how this framework complies with the EAST-ADL2/AUTOSAR methodology and present models expressing functional but also non-functional concerns related to the optimization goals.

CITATION: Wozniak, E., Tucci-Piergiovanni, S., Mraidha, C. and Gerard, S., "An Integrated Approach for Modeling, Analysis and Optimization of Systems whose Design Follows the EAST-ADL2/AUTOSAR Methodology," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 6(1):2013, doi:10.4271/2013-01-1225.

INTRODUCTION

Complexity of hardware/software based solutions in nowadays premium cars is well expressed by the amount of code which overreached 10 millions of lines, number of ECUs that can easily reach the number of 70 or its production cost which can state up to 40% of the overall production costs of the car. Introduction of the Model Driven Engineering (MDE) into the automotive industry provides promising perspectives in terms of lowering the development time and costs. Modeling assures a better, clearer view on the system and analysis/optimization performed on the abstract models can exclude unfeasible system architecture at the early stage of a system design. System modeling can be done with Architecture Description Languages like the EAST-ADL [1] (Electronics Architecture and Software Technology - Architecture Description Language) language, the SysML [2] (System Modeling Language) or can be based on the concepts introduced by the AUTOSAR [3] (AUTomotive Open System ARchitecture) standard. Both have a growing support in the modeling oriented tools. There are also

numerous approaches for analysis and optimization that can be used directly on the system models. For instance schedulability analysis techniques or evolutionary algorithms proved to be efficient for the design space exploration. Their implementation is delivered by tools such as SymTA/S [4] or HiP-HOPS [5].

Unfortunately most of the industrial and research tools serve either for a modeling or analysis/optimization. Therefore the optimization results have to be applied manually on the system models. Secondly, many of these tools use their own, specific notations to specify the constraints and requirements stating an input for the optimization engine. Moreover even for the architecture modeling some of the tools define their own, non-standardized metamodel. In addition, it is difficult to embed these tools into the overall design process following the EAST-ADL2 and AUTOSAR methodology. This is because only specific analysis/optimization can be used at the different abstraction levels of the EAST-ADL2 and AUTOSAR. This follows from the fact that particular abstraction levels contain restricted amount of information.

For instance it is not possible to do the schedulability analysis test at the Design Level of the EAST-ADL2 due to the lack of an OS task concept. Lastly, we are still missing a support for a functional/design level analysis/optimization. This is quite surprising as certain optimization decisions can be already made at this early development stage basing on the information that can be modeled with the EAST-ADL2 language.

In order to resolve these issues, this paper presents UML-based framework where UML [6] serves for the integration. This framework combines the modeling and analysis/optimization engine (called Qompass [7]) within one tool. For the modeling aspect we use SysML UML profile. In order to specify the input for the analysis/optimization (set of system non-functional properties and constraints) MARTE [8] (Modeling and Analysis of Real-Time and Embedded Systems) profile is used. Multidimensional optimization executed on functional models concerns system performance, cost and safety and is done using the genetic algorithms. We also show that it can be used within the EAST-ADL/AUTOSAR methodology at the functional/design abstraction level. This is possible thanks to the set of transformations of EAST-ADL models to our framework's input.

This paper is organized in the following way. The next section will provide current status concerning methodology of the design and exploration, following the standard EAST-ADL2 and the AUTOSAR methodology. In addition to this we will present our optimization goals that can be already considered at the early stages of presented methodology, taking into account available information expressed with EAST-ADL2 models. Following section shows how this information can be modeled with SysML and MARTE. This is because our framework takes as an input models expressed in these languages. Then we formalize the optimization problem and describe in details our exploration algorithm. Lastly we will inspect our approach based on a use case combining ABS (Anti-lock Braking System) [7] and CCS (Cruise Control System) [9] to present some SysML/MARTE models and results of architecture exploration.

METHODOLOGY WITH EXPLORATION

One of the main assumptions of the MDE (Model Driven Engineering) approach for developing real-time systems is to specify both the system functional and non-functional requirements, and finally, to produce binary code that will fully respect the specification. In order to do this, different types of models are used. These models can be then grouped into views which in turn make up abstraction layers that represent the system architecture at different levels of refinement. Such layered system specification is highly dependent on designer experience and domain expertise. This has been one of the many factors that have triggered the foundation of the AUTOSAR standard. The development chain of the AUTOSAR methodology stretches from the

depiction of application software components to the runtime infrastructure, including the description of the hardware platform. Significant drawback of the AUTOSAR is the missing support for the specification of architectures at the functional level. Functional models provide higher abstraction and can be therefore easier to understand. For that reason they are commonly used as a communication medium between interested parties such as customer, for whom deeper technical details are not intelligible. This has prompted the definition of the EAST-ADL language. Its current state and version, EAST-ADL2, is a result of consecutive projects: EAST-EEA [10], ATTEST I and ATTEST II [11]. Further refinement of this language is in progress under the MAENAD [12] project and the new release is expected at the end of 2013. Its constant improvements prove the high interest of automotive OEMs (Original Equipment Manufacturer) in the model-based design encompassing higher levels of abstraction.

EAST-ADL/AUTOSAR Methodology

Adoption of EAST-ADL and AUTOSAR forces the observance of a specific methodology depicted on the [Figure 1](#). This methodology can be divided into three main abstraction levels: Functional Level, Design Level and Implementation Level. The first one provides the highest architectural level of abstraction, and initial translation of the system functional and non-functional requirements. At the design level functions from the higher level are refined into sub-functions which can be either composite (can aggregate other functions) or atomic (non-concurrent entities). In addition abstract hardware platform is provided at this stage, together with an allocation of functions into it. These first two levels are fully covered by the EAST-ADL2. The third, implementation level is represented by the AUTOSAR. AUTOSAR defines a metamodel that specifies an automotive system. This specification is divided into different stages, according to the AUTOSAR methodology: the system configuration input, the system configuration description, ECU specific information, the ECU configuration description, and the final result which is an ECU executable. The highest AUTOSAR level, this is the definition of software architecture (set of software components) and their internal behavior represented by the set of runnable entities. Runnable entities are non-concurrent entities that are the allocation units of the operating system (OS) tasks.

Transformation from the design (EAST-ADL2) to the implementation (AUTOSAR) level is a crucial step, however so far there is no common practice for this. The transformation in which atomic functions from the design level are transformed into runnable entities seems intuitive. Using such approach, together with information about allocation of functions into the nodes, the designer can provide initial configuration at the implementation level. It can be then further refined and evolved by applying design space exploration that will partition runnable entities in tasks and schedule them in a way that will optimize system end-to-

end responses. The only works we have identified, that relate to the transformation problem are [13] and [14].

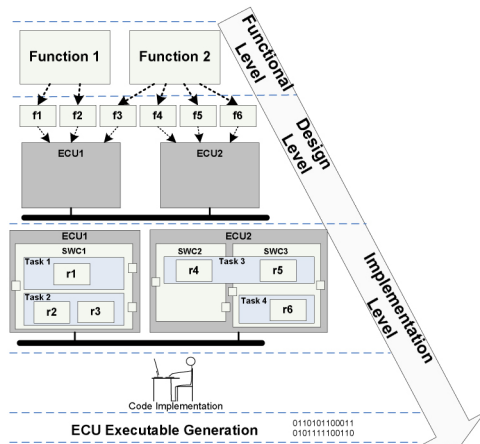


Figure 1. EAST-ADL/AUTOSAR Methodology

Early Optimization Goals

Throughout the overall design, system engineer has to make certain design decisions where most of them have a high impact on system functional as well as non-functional properties. Multitude of decisions to make and complexity of the system resulted in a wide range of algorithms and tools that support a designer. Most of them relate to the implementation level as their input in many cases is platform dependent and requires detailed information. However already at the EAST-ADL level certain design decisions have a tremendous influence on non-functional characteristics such as performance, cost and safety.

Safety

ISO 26262 [15] safety standard imposes consideration of a safety aspect at the functional level. To comply with it, it is crucial to perform Preliminary Hazard Analysis (PHA) in order to reveal the possible effects and consequences of failures [16]. Multitude of techniques exists for the identification of hazards and their causes: FFA, HAZOP, FMEA, FTA, etc. Such analysis should be followed by the planning of necessary safety engineering tasks that will help to decrease the risk assessed with PHA. ISO 26262 provides recommendations on risk prevention or risk mitigation solutions to implement. According to it, safety architecture concept includes the redundancy for the elements. EAST-ADL2 provides a solution to model the redundancy with the concept of the *GenericConstraint*. This stereotype contains property called *target* that defines the subject of the constraint. Kind of a constraint can be further specified with the *GenericConstraintKind* enumeration. The constraint kind that interests us is called *spaceRedundancy*. Textual semantic of a constraint of this kind says that *target* is replicated for redundancy, *genericConstraintValue* times, where the last is the *String* property of the *GenericConstraint*. Figure 2 which represents simple model of the ElectronicBrakeControl

function contains an example of the *GenericConstraint* called *AntilockControlRedundancy*. In this case function prototype called *alc* can have at the most 2 replicas.

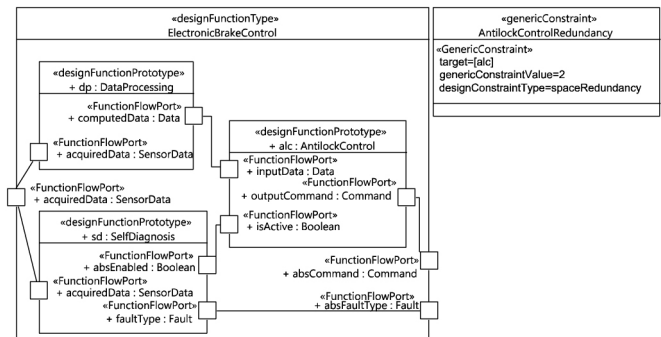


Figure 2. Redundancy Specification

Certain assumptions hold concerning the replication strategy. These are:

1. Active replication style is considered: in this case all the replicas process the same input.
2. Replicas are assumed to be deterministic: the result of a behavior execution depends only on replicas initial state and of the sequence of executions already performed.
3. Replicas behave independently and mechanisms should be provided to ensure that all replicas receive the inputs in the same order to keep consistent the state of replicas.
4. Failed components do not damage the system, and are not repaired.

Cost

The second optimization parameter is the monetary cost of the system. It is computed as a weighted sum of used hardware resources (BUS-es and ECUs) where weight expresses the price of a particular hardware component. This is an intuitive way to analyze the system cost which when specified this way, stays in conflict with a performance of a distributed system [17]. Cheaper components or usage of a smaller number of hardware resources increase the workload of other hardware elements. Naturally there might be other ways to express the overall cost. They can relate to the overall size of on-chip local memory [18], which optimization might cause the increase of a system timing responses [19]. Next, cost of a logical component (in our case system functions) might also be considered. Usually there is relation between the cost of a component and its reliability, namely the higher the cost the higher the reliability of a component [20] [21].

EAST-ADL2 enables to specify the abstract hardware platform at the design level. In addition, for each hardware element we can specify *GenericConstraint* where the *target* property is set to the hardware element for which the constraint applies. The kind of a *GenericConstraint* is now

set to *pieceCost* hence a value of the *GenericConstraint* represents a price of a hardware element.

Performance

The last objective is the performance characterized by the ECU and BUS utilization, i.e. workload. ECU utilization is computed based on the WCET (Worst Case Execution Time) values of the functions that are allocated on the ECU and their execution periods. Similarly, to compute BUS utilization for the event triggered bus such as a CAN (Controller Area Network) all the messages transmitted through this bus, together with their transmission time and period are considered. Period of a message is the same as a period of a function that sends the message. Figure 3 presents simple specification of an event, its triggering period, and execution time of a function expressed with the EAST-ADL2 timing extension.

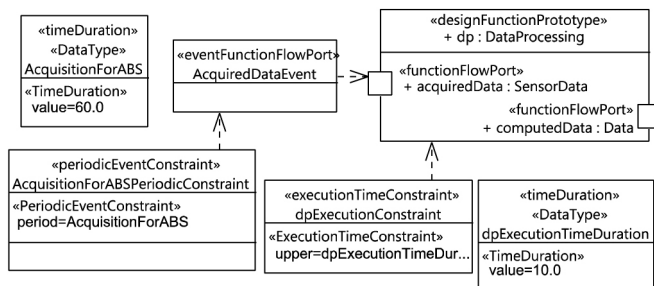


Figure 3. Timing Specification used for utilization computation

In general reasoning about performance might involve many different metrics. Timing performance metric that regards the worst case response times of system end-to-end flows [22] is commonly used. However as stated before, due to the missing task concept in the EAST-ADL2, this metric cannot be considered at the early development stage. Therefore this work accounts for the workload as a performance measure.

SYSMML/MARTE MODELING

Input for our analysis/optimization framework is stated by the SysML & UML4SysML/MARTE models. SysML is a general purpose modeling language for systems engineering applications. SysML as described in the specification is the combination of UML4SysML and the SysML profile. SysML is technology agnostic. Hence it fits well as a modeling language for a generic framework unlike the pure UML which is software oriented. MARTE is used to complement SysML in formal modeling of non-functional requirements and properties related to timing. It is also used to provide additional semantics to the SysML Blocks that enable to differentiate the hardware entities and specify their properties such as a BUS speed.

Functional Structure and Behavior Modeling

Modeling of a system functional structure within the framework is done with the SysML concept of the Block and UML4SysML Property, hence for this, Block and Internal Block Diagram are used (see Figure 4). The same applies to the modeling of a hardware architecture where hardware elements are typed with blocks stereotyped with the MARTE concepts such as *HwComputingResource*, *HwBus*, etc. Behavioral aspects are modeled with the UML4SysML Activity Diagrams. They depict control flows and events. Timing annotations of behavioral diagrams come from MARTE and serve to enrich the models with information about end-to-end deadlines, function time budgets and event periods (see Figure 5). Table 1 highlights the SysML/MARTE concepts used as an input for our analysis tool. Usage of these concepts allows to reason about the utilization of an ECU and a BUS as well as about the monetary cost of the system based on the hardware platform specification.

Table 1. Main modeling concepts used

Modeling Concept	Purpose
SysML::Block	Representation of a system functions and hardware elements. In addition, to differentiate between multiple hardware element types, MARTE annotations are used.
UML4SysML::Property	Instances of functions/hardware elements
UML4SysML::Port UML4SysML::Interface UML4SysML::Operation	Modeling of a client server communication
SysML::FlowPort	Modeling of a sender receiver communication
UML4SysML::AcceptEventAction	Modeling of events triggering end to end flows
MARTE::HwComputingResource	ECU (Electronical Control Unit)
MARTE::HwSensorMA RTE::HwActuator	Sensors and actuators
MARTE::HwBus	CAN or FlexRay bus
MARTE::SaStep	Used to specify the worst case execution time needed for the timing analysis – property <i>execTime</i>
MARTE::GaWorkloadEvent	Used to specify properties related to events (periodicity, period)
MARTE::SaEndToEndFlow	Represents end-to-end flows and enables to specify end-to-end deadlines and store the results of a schedulability analysis
MARTE::Assign	Used to map functions to hardware nodes