Evaluating Modeling Solutions on their Ability to Support the Partitioning of Automotive Embedded Systems

Augustin Kebemou[1] and Ina Schieferdecker[2]

[1] Fraunhofer Institute for
Software and Systems Engineering (ISST)
Mollstrasse 1, 10178 Berlin, Germany
[2] Fraunhofer Institute for
Open Communication Systems (FOKUS)
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany

Abstract. A pool of competing modeling solutions have been proposed to cope with the problems induced by the growing complexity of automotive embedded systems, i.e. the E/E (Electric/Electronic) systems of automobiles. As the principal features of these solutions are axed around modularization and high level of abstraction, it is necessary to investigate their ability to support the implementation. An objective evaluation will be helpful to define how each modeling technique should be enhanced for a better support of the implementation, whenever necessary. This paper defines a framework to evaluate the capacity of modeling techniques to support the implementation in the automotive engineering domain, particularly the partitioning. Following the state-of-the-art in the partitioning of automotive embedded systems, we present the evaluation framework. Then, we introduce the most common modeling solutions used in the automotive embedded systems design and we use the framework to evaluate and classify them.
Index Terms: Automotive, embedded systems, modeling, partitioning

1 Introduction

The development of Automotive Embedded Systems (AES) has incontestably experienced a great leap forward during the last decade. On the way to its maturity, the AES design has adopted the model-based development scheme. Model-based development offers an effective way to decrease the technical and financial risk of "try and error" and improves the economy (design time, material usage, etc.) and the quality (reliability, soundness, performance, EMC, etc.) of the system. Furthermore, model-based development has the potentiality to boost the innovation, afford collegiate work and simplify the product maintenance. All these concerns are quoted to be vital in the automobile industry. Unfortunately, the state of the art in modeling embedded systems in the context of the automotive engineering does not yet allow the designer to take the best possible advantages from model-based development. In fact, even if modeling is current practice for

today's automotive systems designers, models are still considered as simple description and communication media, although in the context of hard competition that rules the automotive industry, modeling can unacceptably continue to be a task that unnecessarily consumes time instead of being helpful and easy.

Hence, even though models are abstractions of the reality, useful specifications must highlight the system characteristics, motivate the design options and facilitate the design decisions. Therefore, a model should bear all necessary information needed for the subsequent design operations. In the context of embedded systems design, one of the most decisive design operations is the design of the system's architecture. We call this task the partitioning. This paper presents a framework for the evaluation and the classification of the modeling solutions that address the AES domain regarding their ability to support the activities of the implementation phase, in particular the partitioning. Even if a comparison might be expected to be supported by quantitative techniques, this paper limits the scope of the framework on the qualitative analysis of the involved modeling solutions. This is sufficient to enable suggestive evaluations with regard to the AES domain.

2 Problem definition and motivation

The overall goal of modeling is to build the system. With a model-based design approach, models are expected to guide the whole design process. That means that all the activities within the life cycle of an AES, from its conception to its destruction, must be supported by models as far as possible. Thus, here, models are the primary artifacts in the system development process. The electronics of AES consist of ECUs, sensors, actuators, gateways and several communication networks. Thanks to the global connectivity enabled by the gateways, these complex, modular and heterogeneous systems can work like a unified system. For example, a power-train member function can communicate with the infotainment system to order the emission of audio signals corresponding to a particular alert. During the design of such systems, decisions must be made about the composition and the topology of the platform on which the system's application will run as well as its implementation. An optimal material usage can considerably reduce the cost and enhance the reliability of AES. As the functionalities of an AES can be implemented on different architectures built each of different hardware components, the choice of the hardware and a goal-oriented partitioning are decisive for both the economy and the performance of the system. Design space exploration allows designers to find optimal implementations of the system by analyzing various alternatives of both the architecture and the topology.

The partitioning of an AES aims at founding the best platform (i.e. hardware components, system's architecture and topology) and distributing the system's working load within the available resources in a manner that the functioning of the system is optimized, by concurrently avoiding resource underutilization. This activity includes three operations: The allocation, i.e. the choice of the components of the infrastructure platform, the mapping, i.e. the assignment of the

elements of the functional specification to the components of the platform and the deployment, i.e. the distribution of the available computing power and the storage capacity of the platform among the elements of the functional model. As shown in figure 1, the mapping is achieved by a clustering that groups the elements of the functional specification of the AES system that should be implemented together in order to profitably share the allocated resources. It results into clusters of functions that represent the logical devices of the system. As these devices communicate through bus networks, the inter-cluster communication can concurrently be assigned to the communication channels of the buses.

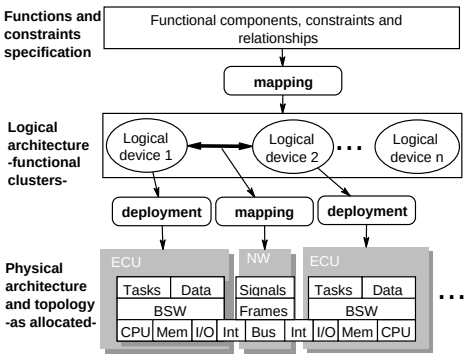

Fig. 1. The partitioning

The goal of the deployment is to assign the computation tasks to the processing units and the logical data to the physical memories in a way that the resource usage is optimized within the required performance and the system constraints (e.g. size, weight, power consumption, safety, speeding up, maintenance, etc.). This operation relies on the scheduling of the tasks on the processing units, the scheduling and the synchronization of the communication within each ECU and the data access procedures. A feasible mapping must allow executable scheduling of the tasks on the containing devices and enable smooth inter-device communications. Thus, in addition to the resource needs and the timing behaviors of the elements of the functional model, the mapping relies on the quality of the information about the inter-components communication and a wide range of relationships between the elements of the functional specification, such as those induced by the strategic concerns of the AES design. AES-desired input specifications must thus enable to clearly identify the boundaries and the interfaces between the components, identify the connection paths, extract the substance and the heaviness of the communications (e.g. throughputs, access rates, data density, timeliness, priorities, security levels, etc.), find out the dependences and causality relations such as sequentiality, concurrency and synchronization, and analyze the internal behaviors of the components (so that their elementary operations and critical paths can be identified) and the relationships resulting from

the strategic concerns of the design. However, the quality of the partitioning depends on the information that is available in the input models. The designer needs powerful and expressive models.

In the current practice, the partitioning is done manually by highly experienced designers, usually called system integrators. When partitioning the system, a system architect must take hundreds of often contradictory, opposite and competitive constraints into account. Keeping this information for a long time in mind is not easy for a human intelligence. A CAD-supported partitioning will be an effective contribution to the dream of model-based system design in the automobile industry. Automated partitioning will be time saving, deterministic and produce well-documented and optimal system architectures. The existing approaches for automated partitioning input very low-level, fine-granular specifications (e.g. logical and arithmetical operations or simple assignments). Unfortunately, because of the complexity of AES, this dimension of granularity is difficult to achieve when following a system-oriented design scheme. As a special domain of interest, important works have addressed the modeling of AES, producing appreciable results. Near general-purpose embedded systems-qualified tools (UML, MatLab, Simulink, SDL,...), more domain-specific modeling solutions have been proposed for the development of AES (e.g. EAST-EEA[1], AADL[2], AUTOSAR[3], etc.). But the most of the known solutions were merely focused on the definition of modeling languages, neglecting the substance of modeling itself and its potential methodological support for the design process.

As each of these solutions pretends to be optimized to support the implementation, it is necessary to investigate the level of support that they provide to the system architects in order to determine how they can be optimally used or how the can be enhanced. We resume this work with the following questions: Which information is needed in a specification to support the partitioning? Which modeling features are needed to provide this information? Do the actual modeling solutions provide these features? How capable are the usual modeling solutions? The rest of the paper is organized as follows: In section 3 we scan the most significant preceding efforts in the evaluation and the classification of embedded systems modeling solutions. In section 4, we define our framework for the classification of AES modeling solutions. Section 5 defines the criteria on which the level of support provided to AES architects will be evaluated. In section 6, we succinctly introduce the most common AES-used modeling solutions, including e.g. UML, SDL, SysML, EAST ADL and AUTOSAR, that are then evaluated and categorized following our framework.

3 Related work

The aim of the evaluation or the classification of modeling techniques is to measure and compare their potential level of support, their adequacy and their usefulness regarding the requirements of the intended design activity, in our case the partitioning. During the partitioning, the decision making is based on the at-

tributes of the model elements like their resource consumptions, their sizes, their need for computation power, their consumption of energy, the magnitude of their collaborations with each other and a lot of other significant interdependencies. To enable the incorporation of the necessary information in the models, a modeling technique must provide a certain level of precision for structuring paradigms, computation paradigms, control paradigms, communication paradigms and for the specification of the constraints and the non-functional requirements.

Several frameworks have been proposed for the evaluation and the classification of embedded systems specification tools. The authors of [4] proposed a classification framework based on five specification styles: State-oriented (using state machines), activity-oriented (using transformations), structure-oriented (concentrating on structural architectures), data-oriented (based on information modeling) and heterogeneous. This classification is mainly based on syntactic criteria. However, it can be used to select a specification style depending on the nature of the behavior that needs to be captured. In contrast, the authors of [5] argue for a classification based on the model of computation (MOC) of embedded systems modeling solutions. Using the Tagged-Signal Model (TSM) [6], a formalism for the description of MOCs aspects, they focus on timing, concurrency and communication aspects to analyze and classify several MOCs. Since a MOC formalizes the execution model of a modeling solution rather than the style in which the specifications are written, this orientation is more objective than the syntax-based classification and is also better adapted to estimate the usefulness of a model. But, generally, as the user is not aware of the MOC of a solution, he also cannot be a priori aware of its quality. A good implementation (i.e. easy and clear syntax, powerful tool support) of a poor MOC is generally far more easier accepted by the user than a poor implementation of a good MOC. MOCs are important characteristics of modeling solutions that can not be ignored when evaluating them. But, an exclusive orientation on the MOCs is not sufficient for our purpose.

Considering the characteristics of the modeling tools from a very different perspective, Hartenstein [7] used four high-level criteria to classify hardware description languages (HDL): The abstraction level, the application area, the dimension of notation and the source medium of the language. Following this author, the abstraction level characterizes the methodological level for which the language has been designed. The area of application is the type of behavior for which the modeling technique has been defined. The dimension of notation is the general class of information supported, e.g. behavioral, structural or morphological information. The source medium is the presentation medium, e.g. graphic or textual presentation. This framework is right in our target. Its main advantage is its simplicity. But, in order to evaluate AES modeling techniques, we need supplementary dimensions of criteria.

The authors of [8] first identified four main classes of computation models defined on the vectorial cross product of concurrency (control-driven, data-driven) and synchronization (single-thread, distributed). Then they defined three high-level criteria to compare embedded systems specification languages, i.e. the ex-

pressive power, the analytical power and the cost of use. The expressive power determines the level of efforts invested when describing a given behavior. The analytical power measures the level of analysis, transformation and verification facilities offered by the language. The cost of use is composed of aspects like the clarity of the models, the quality of the related existing tools, etc. Even though these criteria are very realistic for the evaluation of embedded system modeling languages, this taxonomy is very abstract and limited regarding the characteristics of AES. Furthermore, although it allows to consider important AES modeling features such as timing and concurrency as first class quality criteria of specification languages, the components-based character of AES is not fungible in this taxonomy. A look into a far different research community lets us discover a framework for the classification and the comparison of architecture description languages (ADL) [9] that can efficiently enhance the taxonomies mentioned in [4, 7] and [8] with regard to the AES modeling requirements.

4 The classification framework

The modeling solutions that are used in the AES design can be distinguished following their originating specialization, i.e. the fields of activity for which the solution has been developed, e.g. general purpose, automotive-specific solution, etc. Independently of its specialization, a modeling solution is conceived with focus on a particular domain of application or to address some problems that are specific to some abstraction/conceptual levels, for example some modeling techniques are optimized for abstract descriptions while others are more effective for more detailed, fine-grained descriptions. Also, a technique may be optimized to specify only the interactions between the system's modules, but not the computation performed in the modules while another one is designed only to specify the causality and constraints of the interactions without detailing the interactions themselves. We retain 5 domains of application to classify AES modeling solutions: The modeling of the requirements, the modeling of the architectures, the modeling of the computations, the modeling of the communications and the modeling of the constraints and the non-functional requirements. The most modeling solutions cover a scope of several domains of application. However, for each domain, the modeling techniques differ in the modeling style, the expressiveness, the granularity and the cost of use.

The modeling style indicates the style of writing the models when using a modeling technique, e.g. architecture-oriented models may use object- or component-based techniques while behavior descriptions may vary between algorithmic descriptions, differential equations, state- or activity-based models, etc. A classification based on the modeling style can be used to localize the most adequate modeling techniques according to the nature of the system under construction.

The expressiveness of a modeling technique determines its appropriateness and its usefulness when capturing the characteristics of a specific system. A modeling technique that is not expressive enough to specify a particular item is evidently unsuitable. On the other side, a modeling technique in which the item

of interest cannot be described succinctly is also problematic. The expressiveness of a modeling technique is evaluated based on the suitability of the concepts it supports regarding the nature of the information that is to be captured. The suitability is determined by the ease to describe the system and the clarity that can be achieved. The components of the expressiveness include for example the ability to model the system structures, the support for the modeling of non-software components, the ability to model the computations and the communications, the handling of time and data, the ability to describe concurrency, synchronization and non-functional requirements, etc.

The granularity determines the dimension of the objects described. In other words, it is the (mean) size of manageable information contained in the elements of the models. The size of the objects it manipulates has great influence on the accuracy that a modeling technique can provide. The granularity is measured on the resolution and the level of precision that are achievable with a modeling technique. Coarse granular solutions are efficient for high-level abstract modeling while fine granular ones are more adequate for detail and low-level modeling.

The AES development is a "team-sport" in which different actors coming from different technical domains act in synergy across different OEMs and suppliers with different points of interest. A modeling technique must be easy to learn and use, intuitive, capable to capture and visualize domain-specific items, but related to standards and at the best leaning onto formal notations. These features determine the cost of use of a modeling technique. The cost of use may include further components like the support of CAD tools, e.g. for edition, syntax check, etc., the executability, the synthesizability, the interoperability with other modeling tools, the affinity with the standards and the visualization medium.

However, even modeling solutions that address the same domain of application and that are deemed adequate for the same conceptual level would differently support the partitioning. The following section presents our framework to capture such differences.

5 Evaluating the level of support

The four dimensions of the domains of application mentioned in section 4 might be sufficient to classify the AES candidate modeling solutions, but they are still very abstract to enable an evaluation of the level of support that may be provided. The following taxonomy defines the criteria that indicate the value of a given modeling technique with regard to the partitioning. Depending on the goal of the evaluation, e.g. finding the most adequate, the most useful solution or the one with the best support, a particular combination of these criteria will give the orientation to chose the most appropriate technique.

*Modularity: Independently of the domain of application, the modeling style and the expressiveness, each AES modeling solution should provide some modularization features. The modularity support measures the ability to model the structure and the composition of the system. The modularity is independent of the granularity. But, it is an important characteristic of the expressiveness

and the modeling style of components-based modeling techniques. Concerning the AES design, clear structuring is provided when the system components are clearly identifiable as detachable building blocks with clear boundaries and interfaces. Fuzzy structuring in contrast denotes the difficulty to identify the components and their boundaries. We evaluate the modularity of AES modeling solutions based on the features provided to specify the system's modules and the connections between them. This includes the substance, the encapsulations and the interfaces of the system modules and their connections.
- The substance: A component is normally designed to fulfill a given functionality/service. Both the achievement of a component and its contents must be modeled. The substance of a component defines its role and its composition. A component may be an atomic or a composite structure.
- The interfaces: The interface of a component depends on the way it is encapsulated. Modeling the interface of a component includes the definition of its points of interaction, what it consumes, what it produces, the constraints on these items and the commitments that are necessary to access them, i.e. the type of information that can be consumed and the protocols allowed to be used for the information exchange. Some techniques encapsulate the components using wrappers and virtual interfaces that adapt the communication semantics of the components to the needs of its accessors [10]. Other techniques use special connection components like in [11] where an object-oriented approach is presented with an elegant coordinator concept in which the communication of a composite function is controlled by a coordinator. With this method, the coordinator of a component acts like its communication intelligence. Indeed the coordinator is an intrinsic part of the component. Thus the component's behavior and its communication are always intertwined, making it particularly difficult to separate them and thus to reuse the component since any instantiation of such a component will require to adapt either the accessing components in the destination model or the coordinator, that means the component itself. In the worst case, both must be redesigned. To separate the communication from the behavior, the most methods propose (in- and output) ports. These methods differ in the power they give to ports and the precision of ports descriptions. Some ports are able to transform data, thus holding complex functionalities. Ports can receive directions, types, etc., that simplify the analysis and the synthesis.

*Resolution of the components: The resolution of a component refers to the granularity of the leaves in its hierarchical structure. A leaf component can be as large as the entire system or as small as a logical operation, an arithmetical operation or a simple assignment.

*Computation modeling: A partitioning process will cluster the functions depending on their cost (i.e. computation time, response time), their size, and further attributes like those considering the environment they need to run efficiently (e.g. type of hardware, shareable data, etc.). Therefore, detailed internal behaviors of components are first-class information for the partitioning, that must be precisely specified. The computation description facilities of a modeling solution are characterized by the type of description used to specify the compu-

tations and the provided level of detail. This encompasses the modeling style, the granularity and the expressiveness of these models.

*Communication and data modeling: The attributes of the information exchanged and the protocol governing the communication strongly constrain the partitioning. AES communication may be synchronous or asynchronous, realized by direct information passing or shared memory, in P2P or multi-cast schemes. At different levels of abstraction, the substance of a communication may be specified in terms of services or operations invocations, messages or data block passing, signals or bits flows, etc., and the communication primitives may variate between call/request, send/receive, read/write, set/reset, load/save, etc. The evaluation of the capability of a modeling technique to model the communication is based on the types of communication supported (cf. expressiveness and cost of use), the tools used to capture the communication (cf. modeling style and cost of use) and the resolution (cf. granularity) of the information exchanged.

*Time modeling: Timing information modeling is crucial for embedded systems. The ability of a modeling technique to model time is evaluated through its conceptualization of the notion of time and the resolution of time expression. Time can be expressed through ordering of the activities in the processing (i.e. the order in which things happen induces a notion of time), or as absolute values measured by a clock, this at different resolutions. A modeling technique that can achieve high resolution in modeling timing behaviors is suitable for the partitioning.

*Concurrency and synchronization: Embedded systems behave inherently concurrently. Concurrency has two forms: parallelism and interleaving. Parallel processes run at the same time. They may need to communicate and synchronize, for example to publish their beginnings and ends. Interleaving processes must compete for resources. In order to coordinate the interaction of concurrent processes, some intelligent synchronization mechanisms such as schedulers, message queues (buffers), rendez-vous (for message passing), semaphores or read/write blocking in the case of shared memory are needed. The evaluation of the ability of a modeling solution to model concurrency and synchronization is based on the number of concurrency schemes and synchronization mechanisms that are supported and the quality of the concepts that are proposed to capture them.

*Relation to standards: The distance between a given modeling solution and the nearest standards is an important factor for it acceptance. The relation to standards determines the intuitiveness of a solution, the facility to learn and to communicate it and the possibility to integrate it with other solutions.

*Executability and synthesizability: The executability of a modeling solution refers to the existence of a tool that can be used to simulate the behavior of a system described with this solution. Synthesizable means that there exists a tool that can translate a specified behavior into a machine code or a netlist level model from which properties like memory consumption, hardware size, execution time, etc. can be directly measured. Low-level modeling techniques generally have efficient compilers or synthesis tools that allow rapid prototyping. Some sophisticated high-level models may be executable, particularly when

based on formal definitions. Executable and synthesizable modeling solutions have advantageous cost of use.

*Abstraction levels: Measures the ability to support different, AES domain-established methodological and conceptual abstraction levels.

*Support for variance handling: Depends on the quality of the features provided to support the design of product lines. This includes the modeling of varying elements and of the configuration information.

6 Evaluation and classification of AES modeling languages

Besides the modeling techniques, modeling languages are needed to express the contents of the models. AES modeling solutions generally incorporate each a language that in the reality becomes such a prominence that the whole modeling solution is generally called modeling language. The spectrum of AES-usable modeling languages is very wide, going from general-purpose programming languages and hardware description languages (HDL) to architecture description languages (ADL) and other more promoted languages such as UML, SDL, SysML, EAST ADL, AUTOSAR. General-purpose programming languages, e.g. *C, Assembler, C++, Pascal, Fortran, Java, etc.* are widely used to specify embedded systems. Similar to (HDL) (most known in the area of HW/SW co-design, e.g. *VHDL*, *Verilog*, *System Verilog*, *System C* and *Esterel*), they are optimized for fine-granular design. In contrast to ADLs, both programming languages and HDLs provide executable models and possess proved stable compilers, but they are too close to the implementation and they provide poor abstraction capabilities. SDL, usually used in combination with MSC, ASN.1 and TTCN (ITU standards Z.105, Z.107), provides good message communication and time modeling features as well as an appreciable support of synchronization and good abstraction possibilities. But, since it is OO-based, SDL offers very poor modularization of the system under design.

Since the OMG adopted real-time and embedded systems optimized concepts, e.g. components, events, actions, resources, schedules and timing to enable the high-level design of embedded systems, the UML is becoming popular in the field of the embedded systems software design. However, although UML may provide a modeling power that is suitable to capture the behavior of AES, it does provide neither synthesizable models nor meaningful support for model analysis. Furthermore, UML does not support the AES domain-specific concepts such as the specification and the management of the requirements, the product lines, the configurations, the transitions and the hardware resources. Inspired from UML, SysML adds a requirements diagram to the structure, allocation and behavior diagrams existing in UML. Parametric diagrams are used in SysML to specify the performance, reliability, safety, cost properties of the system under construction, etc. that can support the engineering and trade-offs analysis, thus the partitioning. In addition to these features, EAST ADL defines several conceptual levels while AUTOSAR promotes the standardization of AES' software components

and their interfaces. But, although these languages (i.e. based on UML) provide strong modeling power that can be sufficient to describe the most artifacts of the AES at the high-level, they may not always match the AES architects' ideas as faithfully as desired.

High resolution, clear encapsulation, execution and synthesis tools are needed in both the high- and the low-level design while clear modularity is essential in the higher levels. When the design follows a top-down strategy, none of the above languages can be expressive enough to be used efficiently for all purposes along the design process, since each of them offers in reality only a limited set of features. Otherwise, we are not aware of the existence of such an all-rounder general-purpose modeling language. However, the partitioning of AES needs clearly-framed functional components with their resource consumption properties, their communication paths, their timing behaviors as well as those of the communication between them. Following these requirements for the partitioning, we can classify the studied languages into two groups: Those that are more adequate for the high-level modeling and those that are more adequate for the low-level modeling. The first group contains UML, SysML, EAST ADL, AUTOSAR and the other ADLs, well-suited for the needs of the high-level design. These languages provide powerful architectural modeling features enabling components detachability, but they lack synthesizability. The second group is populated with the programing languages, the HDLs and all kinds of languages that are similar to those used in Matlab, Simulink, Statemate, ASCET-MD, etc. These languages with high resolution, execution and synthesis tools easily fulfill the requirements related with the executability and the synthesizability required for partitioning-compliant languages, but they are unfortunately too fine-grained and only provide fuzzy structuring capabilities, thus being less efficient in the high-level design. Consequently, at each step of the development process, the most adequate language must actually be selected depending on the current conceptual layer, the level of abstraction and the intended use of the model.

However, in addition to the concepts related with the components orientation, the behavior and the interaction description tools borrowed from the UML, the AES domain-specific languages (e.g. EAST ADL and AUTOSAR) and SysML commonly address the domain issues like variants handling, configurations management, hardware platforms modeling, support for non-software components and definition of methodological abstraction levels. This allows them to perform better than the general-purpose ADLs in modeling AES at the high level. They therefore provide a good basis for an efficient AES modeling solution, even if the semantics provided for specifying the ports and the interfaces are not precise enough to support a CAD-supported partitioning. Particularly, the standardization of AUTOSAR interfaces will allow the designer to shift a function from a device to another one, enabling high-level mapping, i.e. partitioning. However, if enhanced with some features allowing for example clear tracing of the inter-components communication paths and the data flow screening, these high-level languages can be very useful for the design of CAD-supported mapping tools.

7 Conclusion

The design of AES begins with high levels of abstraction for which the modeling languages like UML, SDL, SysML, EAST ADL or AUTOSAR are adequate. Even if the syntax is different from one language to another one, most of these languages are based on the idea of components-based systems, i.e. they conceive a system as a set of components communicating through interfaces and ports. All these languages claim sufficient orientation to the implementation, but they still remain very abstract and lack synthesis and execution tools. As domain-dedicated languages, EAST ADL and AUTOSAR provide the most convenient features and the best precision needed to model AES, but they remain very insufficient to support the partitioning of the system. Firstly, because they are not synthesizable. Secondly, the semantics of ports, interfaces and connectors are fuzzy. A promising solution to the first drawback is to combine these languages with low-resolution languages such as programming languages, HDLs, etc. But this will not be the ultimate solution for supporting the partitioning of system specifications at high level. However, if these languages are enhanced with precise computations and communication modeling tools, accurate time and data handling, etc. so that the QoS of the model elements can be extracted and analyzed, then they will represent appreciable solutions to build partitioning-compliant models of AES.

References

[1] EAST-EEA, " Embedded Electronic Architecture. Definition of Language for Automotive Embedded Electronic Architecture v. 1.02," ITEA, Tech. Rep., 30.06.2006.
[2] AADL, "http://www.aadl.info/."
[3] AUTOSAR, "www.autosar.org."
[4] D. Gajski, F. Vahid, S. Narayan, and J. Gong, *Specification and Design of Embedded Systems*. Prentice Hall, 1994.
[5] L. Lavagno, A. Sangiovani-Vincetelli, and E. Sentovitch, *Models of Computation for Embedded Systems Design*. Kluwer Academics Publishers, 1999, ch. 2 in System-Level Synthesis, pp. 45–102.
[6] E. A. Lee and A. Sangiovanni-Vincentelli, "Comparing Models of Computation," in *International Conference on Computer-Aided Design pp. 234-241*, 1996.
[7] R. W. Hartenstein, *A Comparison of Hardware Description Languages*. Elsevier Science Publishers B.V., 1987, ch. 2 in Advances in CAD for VLSI, vol 7.
[8] A. A. Jerraya and al., *Multilanguage Specification for System Design*. Kluwer Academics Publishers, 1999, ch. 3 in System-Level Synthesis, pp. 103–135.
[9] N. Medvidovic and R. N. Taylor, "A Classification and Comparison Framework for Software Architecture Description Languages," UCI and USC, Tech. Rep.
[10] G. Nicolescu, S. Yoo, A. Bouchhima, and J. A. A., "Validation in a Component-Based Design Flow for Multicore SoCs," in *ISSS'02, Kyoto, Japan*, Oct 2002.
[11] M. Mutz, M. Huhn, U. Goltz, and C. Kroemke, "Model Based System Development in Automotive," *SAE*, 2002.