oriented design approach. As demonstrated in [7], this approach must rely on a global view of the system specification in which the whole E/E system is perceived as a single, unified system of hardware and software components rather than a simple collection of parts (see figure 5).

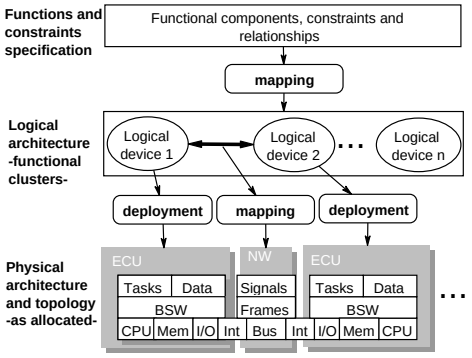

Figure 1. The partitioning

Such a system can become very complex. However, within a model-driven development scheme, a CAD-supported partitioning tool needs highly expressive specifications. A good model must enable to extract the attributes of the elements of the system specification that are necessary for the partitioning as well as their inter-connections. For example, their runtime resource consumptions such as the space needed to store the code, the heaps, the software data and the stacks of a functional component, the computation time required for its execution, its consumption of energy, the magnitude of its collaboration with its environment and a full range of other relationships and constraints. This type of information can be easily provided with the low-level modeling languages such as C, C++, Java, Assembler, etc. But, at the system-level, automotive systems become so complex that they cannot be described with these low-level languages without experiencing an order of explosion of the model that leads to the loss of visibility and makes the navigation within the model impossible.

In order to address this problem, several high-level modeling languages, e.g. SDL[1], SysML[2], AADL[4], EAST ADL[6] and AUTOSAR[3], have been developed and optimized for the modeling of automotive embedded systems. These languages are based on modularization and abstraction techniques, mostly using the principles of separation of concerns found in UML[5], to manage the complexity and the heterogeneity of system-level automotive specifications. But, as high level of abstraction goes together with coarse granularity and low resolutions, the complexity management capabilities of these languages are generally correlated with some shortcomings with respect to the partitioning. Their expressiveness can be fatally very poor compared with the requirements of the partitioning. However, as these solutions are en vogue, it is necessary to investigate the level of support that they provide for the partitioning and, if necessary, identify the missing features that would make them more partitioning-compliant so that we can provide the guidelines for their enhancement or for the development of new solutions. Nevertheless, as the perfect modeling solution is not actually around the next corner, efforts must be made to provide efficient CAD-supported design tools with the existing modeling solutions. This paper gives an insight into our work concerning these questions. In the next section, we discuss the features required from a modeling solution to support the partitioning of automotive E/E systems. The following section presents the analysis of the capabilities of the most popular E/E modeling languages, i.e. SysML, EAST ADL, UML, AUTOSAR, UML, with regard to these features. Then, in section 4, we outline our method to operate the partitioning with the existing modeling solutions.

2 Modeling features required for the partitioning

The goal of the partitioning is to find the most cost-sensitive feasible configuration of the system's architecture, i.e. a partition that minimizes the amount of hardware resources needed to achieve the required performance with respect to the system constraints (e.g. safety, flexibility, maintainability, power consumption, cost, speeding-up, etc.). Depending on the constellation of the design, the mapping can be done before or after the allocation. In the first case, the logical devices are determined and then the best physical devices are allocated to implement each of them. In the second case, the physical devices are given and the logical devices are determined to fit into the available resources. In this case (see figure 1), the mapping is constrained by the allocation. A feasible mapping is one that respects the individual storage and computing capacities of the allocated devices. It must result in an executable scheduling of the system's processes and enable smooth inter-device communication on the allocated communication media. This necessitates precise information about the behavior of the system, its architecture, the magnitude of the communication between the components of the system and the scheduling of their interactions as well as the related constraints. For instance, the scheduling of the communication is determined by the timeliness of the data exchange, that is determined itself by its attributes like the latency, the activation time, the transmission time of the system's components interconnections, etc., while the magnitude of the communication between two components is given by the frequency of the communication and the amount of data exchanged.

As each component of the system's specification must be assigned individually to a device independently of the

these concepts commonly provide the abstraction, composition and decomposition mechanisms needed to manage the complexity, the related semantics remain very fuzzy regarding the partitioning. For example, a connector is used to indicate that two components exchange a given information in some way, although when partitioning an E/E system specification, it is important to know if a connector represents a communication channel, a communication path or simply a connection. Furthermore, the allowed 1 to n links between client and server components or between a sender port and several receiver ports of the same component make the screening of the communication paths, thus the mapping, very difficult.

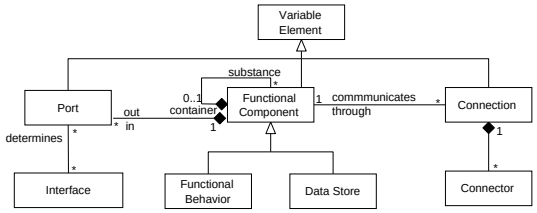

Figure 2. Basic modeling concepts for E/E systems logical architectures

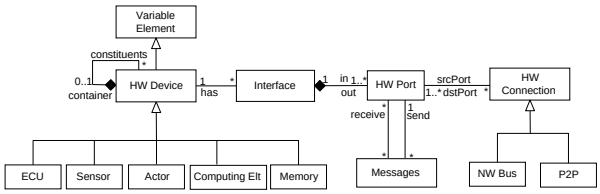

Figure 3. Basic modeling concepts for E/E systems hardware architectures

Figures 2 and 3 show each the meta model of the basic modeling concepts used in EAST ADL, SysML and AADL as well as in AUTOSAR to model the functional structures and respectively the hardware architectures. The concepts used to model the hardware architectures are quite similar with those used to model the functional architectures. A hardware device is generally composed of other smaller devices. For example, an ECU contains processors and memories while a memory is built up of registers, etc. Depending on the level of precision needed, the specification of the hardware platform can be given on the basis of the ECUs, the sensors, the actuators and the gateways or on the basis of transistors, condensators, etc. Figure 4 shows the usual abstraction levels in the modeling of hardware devices.

Except SDL that is based on communication diagrams, a common shortcoming of EAST ADL, SysML and AADL

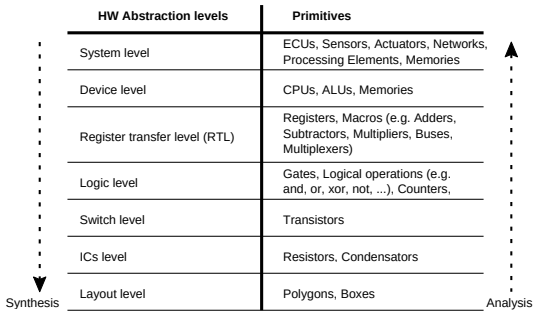

Figure 4. Abstraction levels in the specification of hardware devices

is the lack of appropriate means to specify the system's behavior. However, at the level of abstraction that is inherent to the order of complexity of system-level E/E systems, the granularity of the model elements is too coarse to enable high-resolution behavioral specifications. The most adequate behavioral modeling techniques at this level include state machines, Petri-Nets, activity and mode transitions, data and process flows, etc. These low-resolution modeling tools do not allow to specify the behavior of a system at the level of precision required for the allocation or the deployment. A useful specification must allow to extract the elementary operations of the system's behavioral components and their internal computation paths. The table in fig. 6 summarizes the analysis of these languages regarding their ability to support the partitioning. Following the results emerging from this analysis, SDL provides sufficient features to model the communications but poor capabilities to capture the structure of a system and the internal behaviors of its components, while the ADL-oriented languages provide good structuring capabilities but no feature to capture the behaviors. Thus, none of these solutions is actually the best one to support the partitioning.

4. Our partitioning approach

An imaginable solution to these shortcomings is to combine these high-level languages with low-level languages, i.e. programming languages (e.g. C, C++, Java, etc.) or HDLs (e.g. VDHL, Verilog, System C, etc.) that can be synthesized. There are two approaches to implement this solution. Following the first approach, the encapsulation mechanisms provided by the high-level languages can be used in combination with high-resolution modeling languages at the system level. One way to do this proceeds as follows: Each component of the system specification is wrapped with a capsule that defines its interface. Then, its behavior is precisely described by the means of a high-resolution language and hidden behind its wrapper. This

approach might be very helpful for the partitioning since it provides the necessary attributes at the really beginning of the implementation phase of the design. But, it suffers from the risk of rapid explosion of the size of the specification. To avoid this, low-level languages with high resolution can be introduced progressively, i.e. as long as there is no risk to lose the visibility in the specification. Practically, the highest level at which this condition is fulfilled is within the "Components View" of our design process shown in figure 5. Thus, this solution is practical only if the allocation is done after the mapping. However, as it tends to involve indefinite loops of refinement-abstraction, it can be extremely complex and time-expensive.

Our solution implements the approach shown in figure 5: As the partitioning is a multi-objective optimization problem, we can classify its objectives following their relative importance and the possibility to achieve them so that the objectives that can be achieved with high-level models first guide the partitioning, and then we progressively refine the results in order to meet the remaining objectives. Concretely, in our solution, we model the system as interconnected (SysML, EAST ADL or AUTOSAR) components, whereas the communication between the components is modeled by means of the interaction, communication and timing diagrams provided by the UML. As we can use these tools at the high-level to capture the communication rates of the components of the system, the throughputs of the connectors and ports, the resolutions and the timing schedules of the data exchanged as well as the constraints of the communication relatively accurately, we operate a mapping that optimizes the communication between the devices. In this case, the mapping is reduced to cluster the components that heavily communicate with each other in order to assign them to the same device. The clustering is done independently of the individual internal behaviors of the components of the system. Once a partition is found that minimizes the communication between the components, we refine it with detailed, high-resolution behavioral specifications of the logical devices (figures 5 and 1) so that their resource consumptions can be estimated and thus, the hardware usage within the devices can be optimized during the subsequent partitioning operations.

5. Conclusion

Most of the E/E modeling languages provide a powerful support for the specification of the structures of E/E systems at the high level. But, they do not accurately address the modeling of the behaviors of these systems and thus, they provide very poor support for the partitioning. Our solution to this shortcoming is based on the divide and conquer principles and the flexible design process shown in figure 5. Within this process, we can specify the communications of

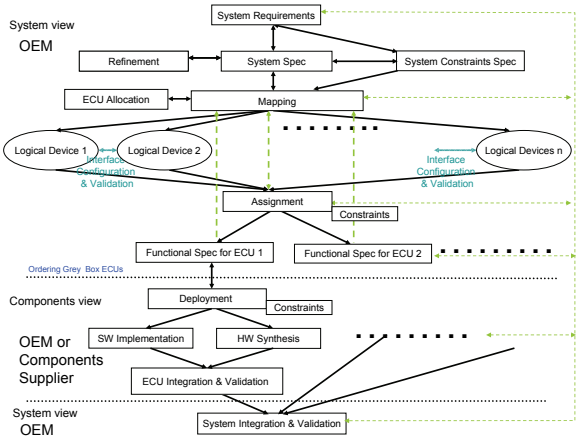

Figure 5. Our system-oriented design flow

the components of the system under design at the highest level of the design by means of interactions, communication and timing diagrams in a way that allows to measure the closeness between them. These closeness values are sufficient to operate the mapping. After the mapping, we introduce high-resolution languages to model the behaviors of the resulting logical devices. This solution is effective since it uses the divide and conquer principle to handle the complexity of system-level specifications, but it is not optimal. A straightforward execution of the partitioning needs precise modeling techniques at the right beginning of the implementation phase of the system development.

References

[1] *ITU-T Serie Z; Languages and General Aspects for Telecommunication Systems; Formal Description Technics- Specification and Description Language (11/99)*.

[2] *Systems Modelling Language (SysML) Specification, OMG document: ad/2006-03-01; version 1.0 Draft*.

[3] *UML Profile for AUTOSAR; V1.0.0; AUTOSAR Adminstration web content, 28.04.2006*.

[4] AADL. http://www.aadl.info/.

[5] B. P. Douglass. *Real-Time UML. Developing Efficient Objects for Embedded Systems*. Addison-Wesley, 1998.

[6] EAST-EEA. Embedded Electronic Architecture. Definition of Language for Automotive Embedded Electronic Architecture v. 1.02. Technical report, ITEA, 30.06.2006.

[7] A. Kebemou. Partitioning Metrics for improved Performance and Economy of Distributed Embedded Systems. *IESS proceedings on IFIP TC10 Working Conference, pp 289-300*, Aug. 15-17 2005.

	UML2	SDL	SysML	EAST ADL	AUTOSAR
Modeling style	Components-based	OO	Components-based	Components-based	Components-based
Substance	Atomic components	Elementary processes	Internal blocks	Elementary functions, Clusters	Runnable entities, Atomic software components
Modularization tools	Components and objects, classes, compositions, subsystems and systems	System, blocks, processes and procedure hierarchies	Blocks and parts hierarchies; internal blocks and packages	Components, analysis, environment, functions, functional devices, elementary and composite functions, etc.	AUTOSAR software components, compositions
Ports	In- and Output ports; Ports are optional	Not explicit	Standard and flow ports	Functions vs. Systems ports; Provided vs. Required, In vs. Out; Signal vs. Operation ports	Provided and Receive ports
Interfaces	Provided and required interfaces	OO interfaces	Provided and user interfaces	Provided and required interfaces, function port and signal port interfaces	AUTOSAR interfaces, standardized AUTOSAR interfaces, Standardized and Private interfaces
Connections	Explicit connectors between components as communication channels	Channels of communication	Explicit connectors for service invocation and signal transfers between ports	Explicit functions and signal connectors between ports	Sender-Receiver and Client-Server
Semantics of ports	Used to structure the interface; Can be stereotyped to contain behavior	Methods	Used to structure the interface; Can be stereotyped to contain behavior	1 port is associated with n message passing interfaces	Port is interface; Message passing interfaces vs Service invocation interfaces
Data handling	Flexible data type definition	Abstract data type (ADT) and ASN1 data models	Flexible data type definition	Flexible data type definition	Flexible data type definition
Time conception	Order and low resolution duration	Order and low resolution duration	Order and low resolution duration	Order and low resolution duration	Order and low resolution duration
Computation modeling tools	Use case, activity, state, transition diagrams	Processes, hierarchical extended FSMs	Use case, activity, state and transition diagrams	UML2 state and interaction diagrams	Activity, state charts and interaction diagrams
Communication modeling tools	Sequence, communication, interaction overview, timing diagrams	Parameterized signal passing through routes and channels	Sequence, communication, interaction overview, timing diagrams	Interaction diagrams	Service invocations, signals passing over connectors
Execution/ Synthesis	Executable behavior models, but no synthesis	Executable specifications and wide extensions for CAD tool-supported synthesis	Same like UML	Executable behavior models; Possible mappings to synthesizable languages are in focus	Same like UML, but contains methodologies for mappings to synthesizable languages
Abstraction levels	Not in focus but easy to conceive	Not in focus but easy to conceive	Not in focus but easy to conceive	5 conceptual levels	Not in focus
Support for non-SW components	Stereotype mechanisms; Extension and adaptation of the notion of component	Not in focus	Stereotype mechanisms plus explicit semantics for hardware device and hardware ports	Explicit semantics for hardware device, hardware ports and connections, cf. environment function	Explicit semantics for system model, ECU model, hardware elements, etc., cf. sensor/actuator SW-C
Transitions modeling	Object type definition and unique identifiers	OO concepts of inheritance; Types and subtypes definitions	Object, class and component types definition; Instantiation mechanisms	Explicit binding charts	Unique identification for SW-Cs
Variance handling and configuration	Not in focus	Not in focus	Not in focus	Explicit concepts of varying/configurable elements	Explicit concepts; Central motivation
Relations with standards	OMG standard	ITU standard	Based on UML2.0; Proposed UML profile	Based on UML2.0; Proposed UML profile	Based on UML2.0; Proposed UML profile

Figure 6. Capabilities of the high-level modeling solutions regarding the partitioning