A Lightweight Meta-Model to Support Automotive Systems and Software Engineering

Georg Macher, Eric Armengaud, Eugen Brenner, Christian Kreiner

A Lightweight Meta-Model to Support Automotive Systems and Software Engineering

Georg Macher[‖], Eric Armengaud[‖], Eugen Brenner[*] and Christian Kreiner[*]

[‖]AVL List GmbH, Graz, AUSTRIA
Email: {georg.macher, eric.armengaud}@avl.com
[*]Institute for Technical Informatics, Graz University of Technology, AUSTRIA
Email: {brenner, christian.kreiner}@tugraz.at

***Abstract*—Modern automotive systems exhibit an increased level of automation as well as an ever-tighter integration with other vehicles, traffic infrastructure and cloud services. Novel features, such as advanced driver assistance systems or automated driving functions, drive the need to master the increased complexity of these systems and ensure consistency of the development along the entire product life cycle. Model-based development (MBD) is still the most promising approach to tackle these issues and support development of system-wide features (such as safety and security). With MBD approaches, the model become the central role for analysis and construction of system under development and for information exchange between stakeholders. Unfortunately, many existing automotive meta-models are enormously complex and tedious to use in efficient manner. This is especially cumbersome in European R&D project cooperation, when different institutions with different field of expertise and diverse tool- and process- setups are required to work together.**

Therefore, this paper aims at improving the information interchange continuity of architectural designs from system development level to software development level with the elementary meta-model required to support systems and software engineering for embedded automotive systems. The presented UML model supports managing of development artifacts and seamless information interchange across tool boundaries to merge heterogeneous tools required for the development of automotive multi-core software.

Keywords*—*model-driven development, automotive systems, multi-core, architectural design, software engineering.

I. Introduction

Already before the introduction of wireless connectivity and automated driving functionalities, embedded automotive systems played a central role in the automotive domain. In the automotive industry, embedded systems are responsible for an added value ranging to up to 75% while making up 25% of the vehicle costs [23]. Current premium vehicles are characterized by several tens of distributed control units (more than 90 electronic control units with close to 1 Gigabyte software code [6]), with a complexity level of the E/E architecture significantly constraining vehicle performance improvements. This trend is even more pushed by the ongoing integration of external services and environmental information, as well as, the increased level of automation features.

In general, embedded system developers are confronted with more hardware and software resource constraints and need to comply with more rigorous dependability requirements and standards than desktop computer development [11]. The higher degree of integration and the criticality of the control application increases the system's complexity and raises new challenges.

To handle these issues, also in relation to rigorous automotive domain standards (such as ISO 26262), model-based development (MBD) is still the most promising approach. Model-based development supports the description of the system under development in a more structured way and enable different views for different stakeholders, different levels of abstraction, and central storage for information. With MBD approaches the model becomes the central role for analysis and construction of system under development and for information exchange between stakeholders. Due to this central role, many established automotive meta-models became enormously complex and tedious to use in an efficient manner. Especially in context of European R&D project cooperation, when different institutions with different field of expertise and diverse tool- and process- setups are required to work together, complex meta-models can hamper the cooperation.

Consequently, this work focuses on improving the continuity of information interchange from system development level to software development level for embedded automotive systems. The contribution of this paper is a lean meta-model required to (a) manage development artifacts required for systems and software engineering of embedded multi-core systems and thus (b) bridge the existing gap between model-driven system development tools and software engineering tools (both application software and basic software tools). The motivation is to provide an elementary meta-model required to support the work of systems and software engineering for embedded automotive systems in European R&D project constellation.

In the course of this document, a brief description of the

state of the art and related work is given in Section II. In Section III, a description of the proposed approach and a detailed depiction of the contribution parts is provided. An implementation prototype and a brief evaluation of the approach is presented in Section IV. Finally, this work is concluded in Section V with an overview of the work presented.

II. Related Work

Model-based development in general and the development of embedded automotive systems in particular are engineering domains and research topics aimed at moving the development process to a more automated work-flow, which improves in terms of consistency and tackles the complexity of the development process across expertise and domain boundaries.

An important topic to deal with in general terms is the gap between system architecture and software architecture. Broy et al. [5], already claimed model-based development to be the best approach to manage large complexity of modern embedded systems in 2008 and provide an overview of basic concepts and theories. The work also illustrated why seamless solutions have not been achieved so far. Frequently seamless solutions scupper due to problems arising from the use of an inadequate tool-chain (e.g. redundancy, inconsistency and lack of automation).

In European R&D context, the projects AMALTHEA [1], SAFE [2], and MAENAD [3] more recently also focus on model-based development environments for automotive multi-core systems. The MAENAD project focused on design methodologies for electric vehicles based on EAST-ADL2 language and the AMALTHEA project on an open source tool platform for engineering embedded multi- and many-core software systems. The SAFE project focused more on the efficient development of safety features in cars and the traceability of safety requirements through the whole lifecycle.

This works contribution was also affected by aspects of SPES XT [4], the idea of an open safety meta model enabling a modular, cross-tool and cross-company safety certification, and the sub-goals of EMC^2 [5] focusing on multicore technologies (SW, HW, tooling, analysis) in dynamic real-time environments. The fundamentals for this work are strongly related to CESAR project[6] and its successor CRYSTAL[7], which provide interoperability concepts for an overall development tool chain. These technologies were used as a basis in order to derive specific concepts for the exchange of dependability information.

The work of Quadri and Sadovykh [19] presents a model-driven engineering approach aiming to develop novel model-driven techniques and new tools supporting design, validation, and simulation. These authors defined profiles using a subset of UML and SysML for their approach and mentioned the usage of effective design tools and methodologies as crucial to be capable of managing complex real-time embedded systems.

The work of Holtmann et al. [8] highlights process and tooling gaps between different modeling aspects of a model-based development process. Often, different specialized models for specific aspects are used at different development stages with varying abstraction levels and traceability between these different models is commonly established via manual linking.

Santos et al. [22] outlines the model-based software engineering process in SysML (Systems Modeling Language) together with Simulink and compares the models for the amount of useful information, which can be transferred into software development phases. The approach addresses the V-Model through SysML and MBD in Matlab/Simulink.

The AUTOSAR consortium [1] and their AUTOSAR methodology was founded to provide standardized and clearly defined interfaces between different software components. The AUTOSAR approach features three different classes of implementation (ICC - implementation conformance class). AUTOSAR ICC1 approach clearly benefits on the time-saving in terms of no additional familiarization with usually very complex and time-consuming AUTOSAR tools, compared to full AUTOSAR approach (ICC3). The related tool framework ARTOP (AUTOSAR Tool Platform), presented in [24], is an infrastructure platform that provides features for the development of tools used for the configuration of AUTOSAR systems. These features are base functionalities that are required by different AUTOSAR tool implementations.

The new approach, AUTOSAR Adaptive Platform [3] implements a run-time environment for Adaptive Applications (ARA). The platform follows a Service-oriented Architecture (SOA) approach for future use for automated driving functionalities (ADF) and advanced driver assistance systems (ADAS). The initial version of AP R17-03 has been released as planned on March 31st 2017. It describes behavior of the software platform from application and network perspective but does not constrain the SW architecture of a platform implementation. In comparison, the AUTOSAR run-time environment (RTE) for the Adaptive Platform [1] dynamically links services and clients during run-time. Nevertheless, the focus of the AUTOSAR approach is set only on the software development level and the related meta-models are enormously complex and tedious to use efficiently in European R&D project cooperation. Time, available expertise and license-cost constraints are frequently the limiting factors in such projects.

Safety standards, such as the road vehicles functional safety norm ISO 26262 [10] and its basic norm IEC 61508 [9], present requirements and guidance for safety-critical system development and also mention MBD as a preferred development approach, but specific MBD specifications are not given. In this context, Born et al. [4] recommend a transition from a document-centric approach to a model-based approach. Their work mentions the problem that organizations already have

[1] http://www.amalthea-project.org/
[2] http://safe-project.eu/
[3] http://maenad.eu/
[4] http://spes2020.informatik.tu-muenchen.de/home.html
[5] https://www.artemis-emc2.eu/
[6] http://www.cesarproject.eu
[7] http://www.crystal-artemis.eu/

their own safety processes in place and want to keep their existing document-centric processes and tool landscape, which mostly inherits fundamental flaws in terms of traceability, a key requirement in ISO 26262.

Lovric et. al [12] sees SysML and model-based development (MBD) as the backbone for development of complex safety critical systems as a key success factor. The integration of SysML models for the development of the ECU safety concept ensures efficient design changes, and immediate awareness of functional safety needs. The paper evaluates key success factors of MBD in comparison to legacy development processes in the field of safety-critical automotive systems.

III. Automotive Systems, Safety, and Software Engineering Model

The methodical support of system architectural design and refinement of this design to software design often fell short of the mark. To handle this situation the AUTOSAR methodology [1] provides standardized and clearly defined interfaces between different software components and development tools and also provides such tools for easing this process of architectural design refinement. Nevertheless, the enormously complex AUTOSAR model requires high amount of preliminary work and projects with limited resources often struggle to achieve adequate quality in budget (such as time or manpower) with this approach.

The model presented in this work has thus emerged from full AUTOSAR based approaches, SysML, or EAST-ADL [8] approaches and focuses on a lean MBD model to gather all required information in a central MBD database as a single-source of information concept. The basis of our approach stems from the CESAR Project [20] to support continuous safety related system development according to ISO 26262 at concept phase and system development level and goes beyond these approaches to tackle also HW/SW development phases. The modeling tool in use is Enterprise Architect (EA) with special extension addin to constitute the central source of information. This database inherits all information of the involved engineering disciplines (system, software, and safety) in a structured way, and allows different engineers to do their job in their specific manner. Furthermore, it enables a reorganization from a document-centric development approach to a seamless model-based development approach. The presented model has been developed using profiles, which use a subset of the SysML language to define a system model particularly tailored to automotive engineering and safety engineering in context of ISO 26262. The system level concept of the approach was proposed by Mader [17] and is based on a specific tool-independent and language-independent methodology to support continuous safety analyses of system architecture development according to ISO 26262 at concept phase and system development level.

Within this work we describe the additional model enhancements to support also software development of basis software functions (HW driver), operating system configuration and task allocation, and modeling of complex software architectures for function software development in the context of modern multi-core systems. The proposed meta-model was intended to support systems and software engineering for embedded automotive multi-core systems within European R&D projects. Therefore, the approach has to fulfill the following requirements:

- light-weight meta-model which requires short training phase
- open source approach, not requiring additional licenses
- tool-independent applicability
- support system engineering in safety-related context
- bridge the existing gap between model-driven system development tools and software engineering tools (both application software and basic software tools)
- support of specifics of multi-core system development
- supporting HW-SW interface definitions

Table I compares the proposed solution with other model approaches and indicates different improvement factors. These factors highlight benefits/supported features (indicated with $+$), drawbacks/not supported features (indicated with $-$), and un-affected/somehow supported aspects (indicated with o).

The main benefit of this proposed approach contributes to (a) closing the gap, also mentioned by Giese et al. [7], Holtmann et al. [8], and Sandmann and Seibt [21], between system-level development at abstract UML-like representations and software-level development and (b) a meta-model reduced to the very minimal requirements to support this bridging and comprise all development artifacts of embedded automotive multi-core development. This enables information transfer between system engineering tools and software engineering tools. Furthermore, the model minimizes redundant manual information exchange between tools and contributes to simplifying seamless safety argumentation according to ISO 26262 for the developed system. The benefits of this development approach are clearly visible in terms of re-engineering cycles, tool changes, and reworking of development artifacts with alternating dependencies, as also mentioned by Broy et al. [5]. Moreover, the presented model is, due to its simplicity and tailoring of the very fundamental needs of embedded automotive multi-core development, an optimal basis for education and training of young engineers and students. The following sections describe the key contribution parts of the model in more details.

A. System Engineering Model Part

To support the modeling approach on system level SysML 1.1 was modified using a meta model enhancement. This meta model is implemented as an UML profile for Enterprise Architect (EA) and requires only very little adaptations.

[8] http://www.east-adl.info/

TABLE I. COMPARISON OF MODELING APPROACH FACTORS

Indicators	Proposed Approach	CESAR Approach	AUTOSAR meta-model	SysML	EAST-ADL	AMALTHEA Approach	SAFE Approach	SPES XT Approach
System Engineering Support	+	+	o	+	+	o	+	+
SW Engineering Support	+	-	+	o	o	+	o	+
HW - SW Interface Support	+	-	+	-	-	o	o	o
Timing Support	+	-	+	-	-	+	-	o
Additional (safety) constraints (such as ASIL indicator, requirements)	+	+	-	o	o	o	+	+
Multi-core system support	+	o	+	o	o	+	o	o
Model Complexity	+	o	-	-	-	o	o	o
Licenses	+	o	-	+	+	+	o	+
Tool-independent solution	+	o	o	+	+	+	o	+
Training / Learning phase	+	o	-	o	o	o	o	o

For the item definition, the preliminary architecture of the item is annotated with non modified SysML 1.1 elements and specific connectors for information, material and energy flows (depicted in Figure 1). For this purpose a block definition diagram (BDD) and an internal block diagram (IBD) shall be created using SysML 1.1 and the proprietary elements defined.

The BDD represents the top or container element of the individual components as $<< block >>$ element. These element serves as container for all the other elements created in the BDD before and also represents the system boundary. The IBD is used to model the dependencies between the components. To model the dependencies between the components, proprietary flowPorts are used, which represent physical connections between the components. To specify the type of the connection, different types of flowPorts are introduced as depicted in Figure 1 and should be chosen in respect of its principal purpose. Further, also the direction of the flow of every flowPort (can be either in, out or inout) and its SI unit must be specified. Defining the BDD and IBD is the basic procedure to model the structure of the system under development.

Describing the main functionality of the item is done by using SysML standard use case diagrams (UCD) and activity diagrams (ACT) and is subject to publications [17], [18].

B. Application Software Model Part

The first SW part is a specific UML model enabling software architecture design in AUTOSAR like representation. A specific SysML profile is used to limit the SysML possibilities, to the needs of software architecture development of safety-critical systems and enable software architecture design in AUTOSAR like representation within the system development tool (Enterprise Architect).

This profile makes the SysML representation more manageable for the needs of the design of an automotive software architecture by taking advantage of an AUTOSAR aligned VFB abstraction layer. Additionally to the AUTOSAR VFB abstraction layer [2], the profile enables an explicit definition of components, component interfaces, and connections between interfaces. This provides the possibility to define software architecture and ensures proper definition of the communication between the architecture artifacts, including interface specifications (e.g. upper limits, initial values, formulas). The SW architecture representation within EA can be linked to system development artifacts in the same tool and traces to requirements can be easily established. This further benefits in terms of constraints checking, traceability of development decisions (e.g. for safety case generation), reuse, and ensures the versatility to also enable AUTOSAR aligned development as proposed in [13]. Note that the proposed ASW model matches the classic AUTOSAR approach, but does not feature the more recent AUTOSAR adaptive approach with its adaptive service interfaces.

Figure 2 shows the representation profile of application software architecture artifacts. Note attributes starting with an underscore character (such as _image, _sizeX, and _metatype) are solely for representative purposes within EA and do not directly belong to the proposed meta-model. The AUTOSARComponent is used to design the SW architecture and represent the SW components. These components can be of 9 different kind (highlighted in lower left corner of Figure 2), representing AUTOSAR standard aligned component types, triggers, and BSW interfaces. Additionally, the AUTOSARComponent artifacts inherit an ASIL classification, as required for safety-related development according to ISO 26262 (also highlighted in the lower left corner of Figure 2).

The AUTOSARPort class is used to represent the SW interfaces (ports). These interface artifacts include the port direction (in, out, or inout), the respective data type (e.g. uint8) and AUTOSAR port type (highlighted in lower right corner of Figure 2). A summary of all information represented by the SW interface artifact can be found in Table II. The connector type is only used for visualization and automated constraint checking functionalities. As can be seen in the depiction, all artifacts required to model the SW architecture are represented and inherit the required information as tagged values. Note that some of the value and tag combinations might not be applicable (e.g. for AUTOSAR Client Server port configuration some tags will be ignored). However, this

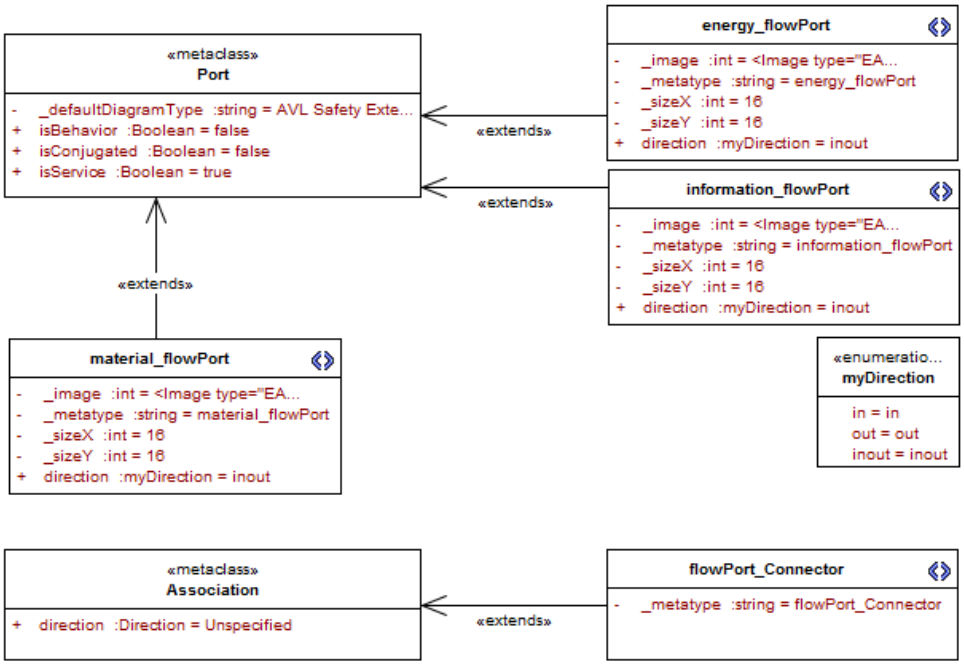

Fig. 1. Meta-model Definition of flowPorts

validity and constraint checks, as well as, configuration option may be subject to other publications and have been omitted in this work. The proposed approach thus supports the ISO 26262 requirements of traceability along the development process and also ICC1 AUTOSAR aligned development. Figure 3 shows an example of software architecture artifacts and interface information represented in Enterprise Architect.

TABLE II. ESSENTIAL HSI ATTRIBUTES REPRESENTED IN THE APPLICATION SW MODEL

Artifact	**Configurable Attribute**
SW signal name	signal identifier for ASW
port type	AUTOSAR aligned port types
signal direction	input or output
ASIL	Automotive Safety Integrity Level
SW data type	-
scaling LSB scaling offset	fixed-point arithmetic scaling
SW min range	-
SW max range	-
SW unit	physical unit representation
default value	default value in case of invalid signal

C. Basic Software Model Part

The basic software (BSW) model extension allows the graphical visualization of OSEK OIL objects and representation in the MBD environment. This profile extension ensures the accumulation of additional information which enable the mapping of tasks to a specific core and clear arrangement of dependencies and shared resources in context of multi-core development. Figure 4 shows the additional profile elements and their accumulated element information. This part of the model offers an intuitive way for highlighting functionality of safety-related software tasks and resources. This also enables the possibility of a traceable automatic OIL file configuration generation instead of the typical manual definition, which inherits increasing significance in terms of safety-critical system development according ISO 26262. Consequently, the system description can be refined down to the operating system. Note that this model part is straightened for the OSEK-based specifications for embedded operating system, communications stack, and network management protocol for automotive embedded systems. For more information related to the OSEK support see [15]

As mentioned previously, the BSW model part (depicted in Figure 4) includes timing information representation possibilities for tasks and OSEK OS related objects and attribute properties, such as:

- TASK - software functionality handled by the RTOS
- OS - specification of the RTOS of the specific core
- EVENT - synchronization mechanism for tasks
- APPMODE - definition of different application modes to control task features or support reduced functionalities
- ISR - specification of interrupt service routines
- ALARM - notification mechanism
- RESOURCE - specification of different resources of the ECU

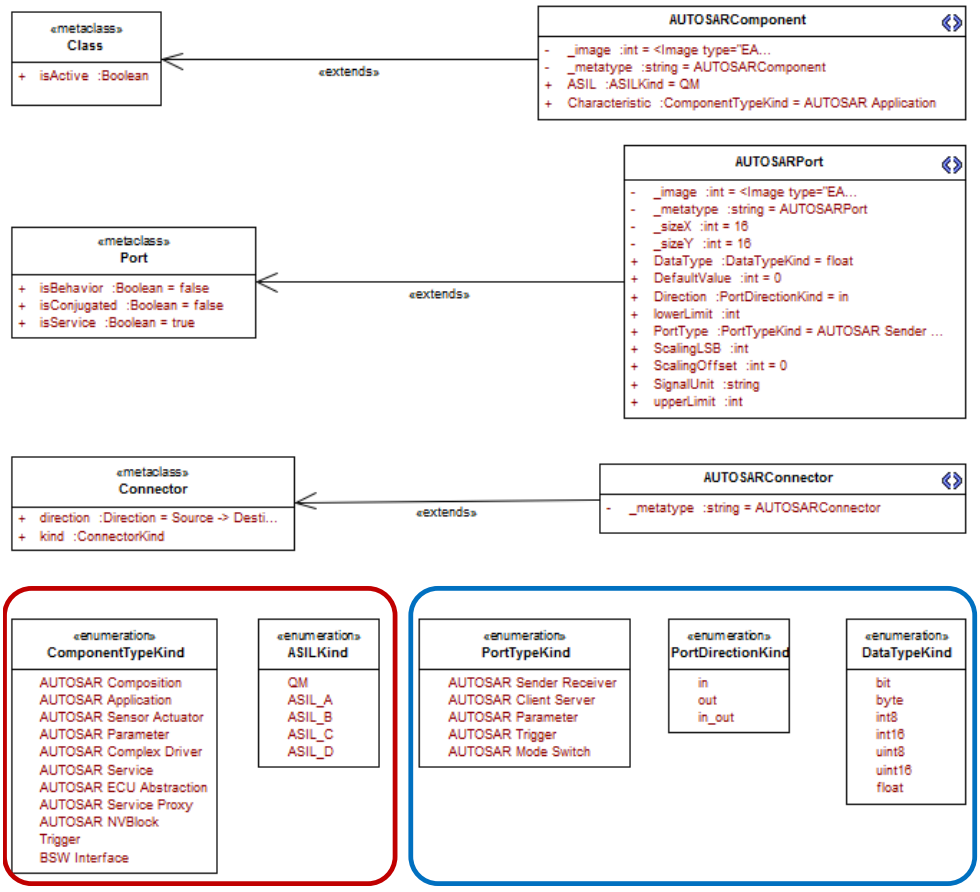

Fig. 2. Representation of Application SW Artifacts

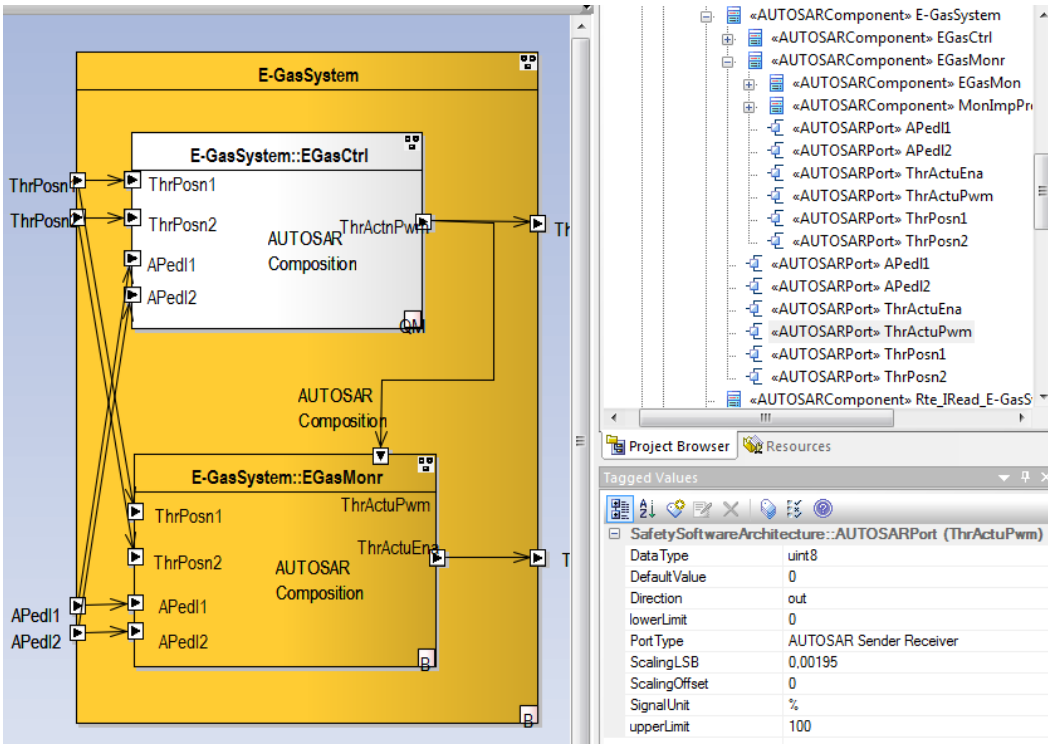

Fig. 3. Screenshot of the SW Architecture Representation within the System Development Tool and Representation of the Interface Information

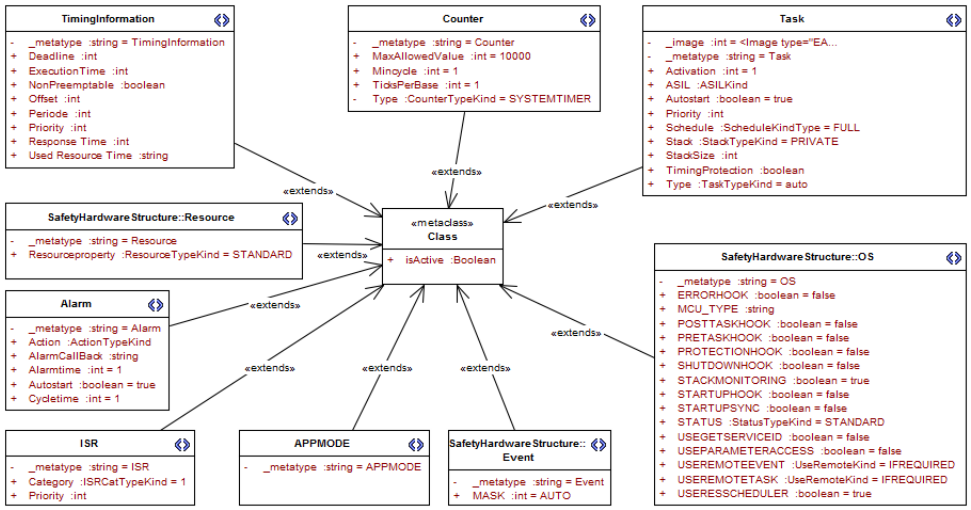

Fig. 4. Meta-model Representation of Basic SW Artifacts and OSEK Artifacts

– COUNTER - a SW/HW resource for alarms

As depicted in Figure 4 the BSW profile is extended to additional include OSEK and timing information. The ability to specify timing constraints of software modules enables the ability to use the model to interlink scheduling and task allocation analysis tools. This enables the analysis and optimization of resource utilization, especially important for multi-core system development. This enrichment may only be highly required for development of multi-core systems, but, due to the increased amount of multi-core development in the automotive domain, is still not counted contradicting to the requirement of a minimal model.

D. Hardware Model Part

The AUTOSAR architectural approach ensures hardware-independent development of application software modules until a very late development phase. The previously mentioned basic software (BSW) and the hardware representation are assigned to establish links to the underlying basic software and hardware layers. Therefore, the hardware profile of the approach, depicted in Figure 5, allows a graphical representation of hardware resources and connected peripherals which interact with the software. This enables an intuitive graphical means of establishing software and hardware dependencies and a hardware-software interface (HSI), as required by ISO 26262. Software signals of BSW modules can be linked to HW port pins via dedicated mappings (depicted as HwSwMapping in the center of Figure 5). Additionally, SW units can also be allocated to dedicated cores (also multi-core systems) and HW resources. This enables the modeling and mapping of HW specifics and SW signals, which establishes traceable links to port pin configurations (depicted as ConnectorPinSetting on the bottom of Figure 5). More details regarding the HSI definition can be found in [16].

IV. Application and Evaluation of the Proposed Model

This section briefly demonstrates the usability of the model for development of automotive embedded systems. The presented model is the fundamental part of a bridging approach [14] to seamlessly transfer system development and software development tools. For this evaluation a prototypical implementation of the approach has been made for Enterprise Architect [9]. An automotive use-case of a central control unit (CCU) of a battery management system (BMS) prototype for (hybrid) electric vehicle has been chosen for evaluation of the approach. This use-case is an illustrative material, reduced for internal training purpose and is not intended to be exhaustive or representing leading-edge technology.

The system model of the system under development consists of 1132 model artifacts in total and 2206 connections between these elements. 163 information flow ports and 103 energy flow ports (representing mechanical and electrical energy) are used to represent the dependencies between the different model artifacts on the high abstraction level. Details of these higher levels of abstraction are omitted due space constraints but are related to 144 system artifacts distributed over 170 packages, including 76 diagram representations. The depiction in Figure 6 constitutes a more concrete layer of system development. This figure shows the connections between the main module (CCU) and the various satellite modules of the INCOBAT battery. Information and energy exchange ports as well as an optional fan controller are depicted and the information ports of the HV interlock loop (HVIL) can be seen.

The definition of the software architecture is usually done by a software system architect within the software development tool (Matlab/Simulink). Using the presented approach this work package is also included in the system development tool.

[9] http://www.sparxsystems.com/

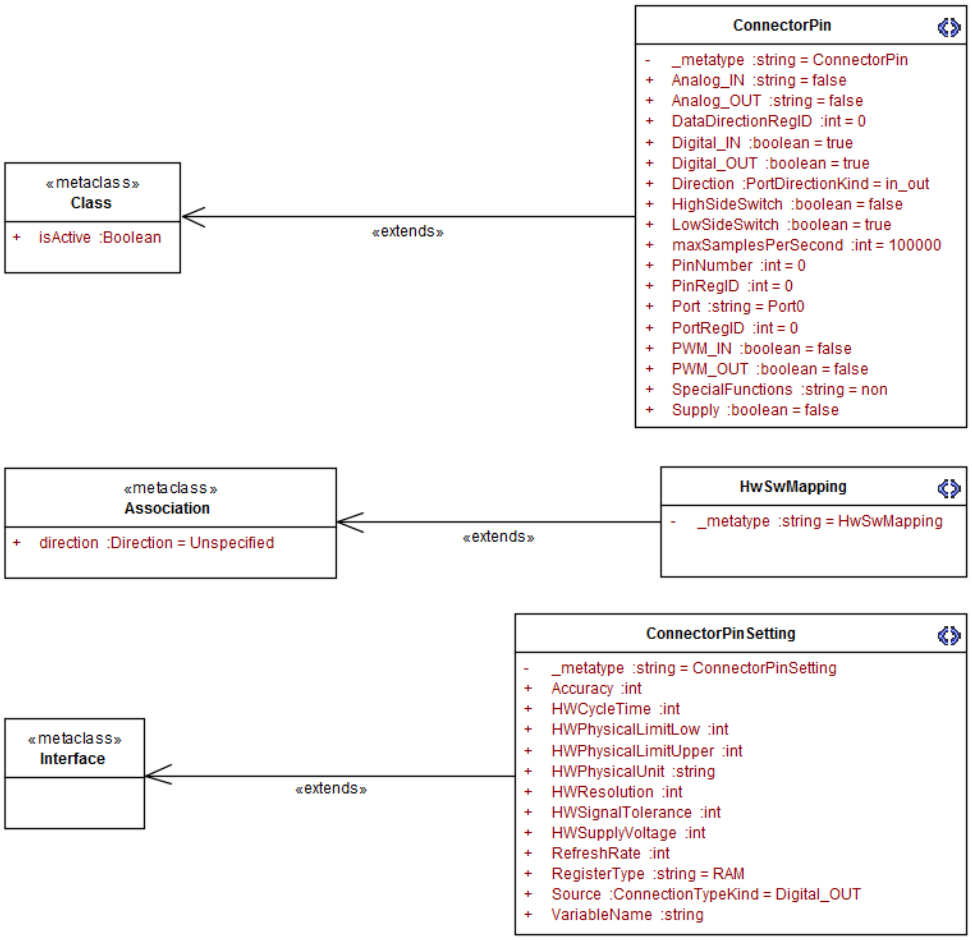

Fig. 5. Meta-Model Representation of HW Artifacts

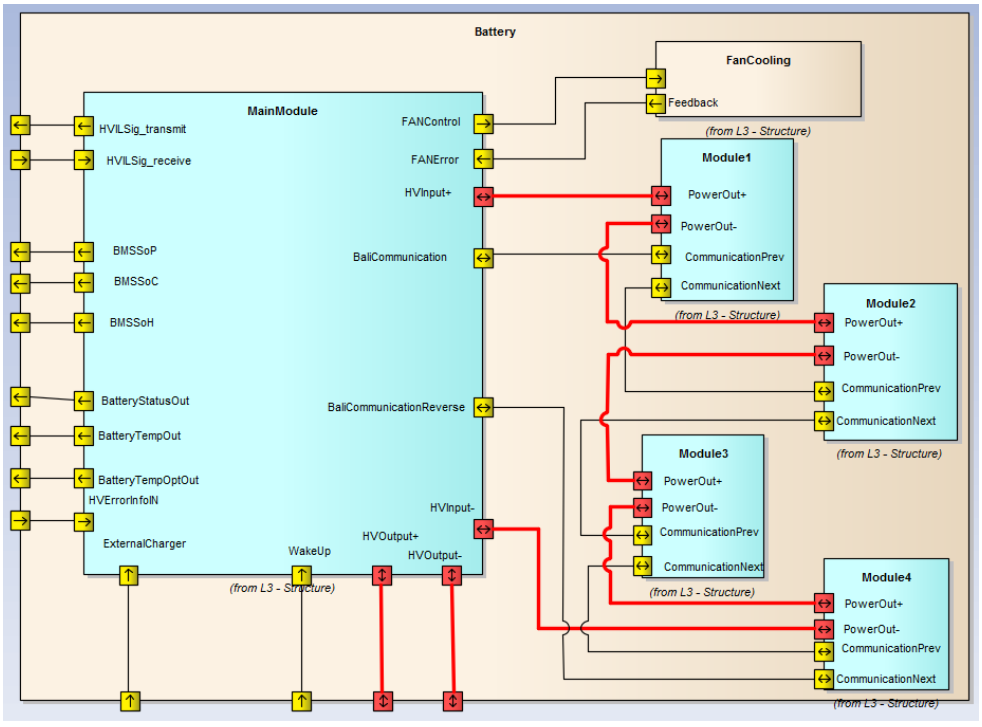

Fig. 6. Depiction of the Battery Control System

Object type	Element-count	Configurable Attributes per element
ASW Modules	15	3
BSW Modules	11	3
ASW Module Inputs	54	10
ASW Module Outputs	32	10
ASW/ASW Interfaces	48	-
ASW/BSW Interfaces	19	-
HW/SW Interfaces	19	13
CPU	4	2
OS	4	15
APPMODE	2	1
TASKS	47	9
COUNTER	12	5
ALARMS	16	6

TABLE III. OVERVIEW OF THE EVALUATION USE-CASE, ELEMENT COUNTS, AND NUMBER OF CONFIGURABLE ATTRIBUTES PER ELEMENT

This does not hamper the work of the software system architect but enables the possibility to also link existing HSI mapping information to the SW architecture.

This abstraction level, depicted in Figure 7, is not included in classical system development approaches but defines the start of parallel software and hardware architecture design. This abstraction level is therefore mostly only present in special purpose SW development tools or HW design tools rather than SysML based system development tools. With the presented modeling approach this tool breach and semantic gap can be bridged and SW architecture definition can be represented by the MBD tool.

The SW model consists of 15 ASW modules and 11 BSW modules with 19 interface definitions between ASW and BSW, 1 operating system instance per core which schedule 47 tasks and manage 23 HW resources (including among others,system counters, CPUs, and peripherals). Table IV provides an overview of the different modeling artifacts and the number of configurable attributes per element. Figure 7 depicts an excerpt of the SW architecture modeled within the system modeling tool. In the figure the HVIL management SW implementation is show with its interfaces and SW subcomponents. Also a highlighting of the assigned ASIL and a decomposition can be seen.

As can be seen in Table IV, 7 ASW/BSW input interfaces and 12 ASW/BSW output interfaces need to be defined. This definitions can be used for automatic generation of HW/SW interfaces and automatic transferring of SW architecture to the SW development tool. Furthermore, the managed amount of 85 operating system objects (OIL container) and relations between them can be used to automatically configure the operating system. This small example already indicates that relations between the elements increase quickly and become confusing. In terms of safety-critical development and reuse the presented model can serve as basis to support round-trip engineering for information transfer between separated tools and links to supporting safety-relevant information.

V. CONCLUSION

The challenge with modern embedded automotive multi-core systems is to master the increased complexity of these systems and ensure consistency of the development along the entire product life cycle. Automotive standards, such as ISO 26262 safety standard provide a process framework which requires efficient and consistent product development and tool support. Model-based development (MBD) seems to be the most promising approach to tackle these issues and support development of system-wide features (such as safety and security). Nevertheless, various heterogeneous development tools in use hamper the efficiency and consistency of information flows and existing automotive meta-models are enormously complex and tedious to use in efficient manner.

This work thus provided the minimal meta-model required to manage these development artifacts required for systems, safety, and software engineering of embedded multi-core systems. On basis of this model several heterogeneous tools required for development of automotive systems can be linked to support seamless interchange of information across tool boundaries [14]. Additionally, the minimalistic meta-model can optimally be used for development trainings of embedded automotive multi-core systems for young engineers and students, due to its limited overheads and the resulting steep learning curve.

The applicability of the model has been demonstrated utilizing an automotive BMS use-case, which is also intended for training purposes for students and engineers and does not represent either an exhaustive or a commercially sensitive project. The main benefits of the presented model are: improved consistency and traceability from the initial design at the system level down to the software implementation. Further improvements of the approach include the progress in terms of reproducibility and traceability of design decisions.

ACKNOWLEDGMENTS

This work is supported by the *DEIS* project - *Dependability Engineering Innovation for automotive CPS*. This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 732242.

REFERENCES

[1] AUTOSAR development cooperation. AUTOSAR AUTomotive Open System ARchitecture, 2009.

[2] AUTOSAR Development Cooperation. Virtual Functional Bus. online, 2013.

[3] AUTOSAR Development Cooperation. Adaptive Platform Release Overview. online, 2017.

[4] Marc Born, John Favaro, and Olaf Kath. Application of ISO DIS 26262 in Practice. *CARS 2010*, 2010.

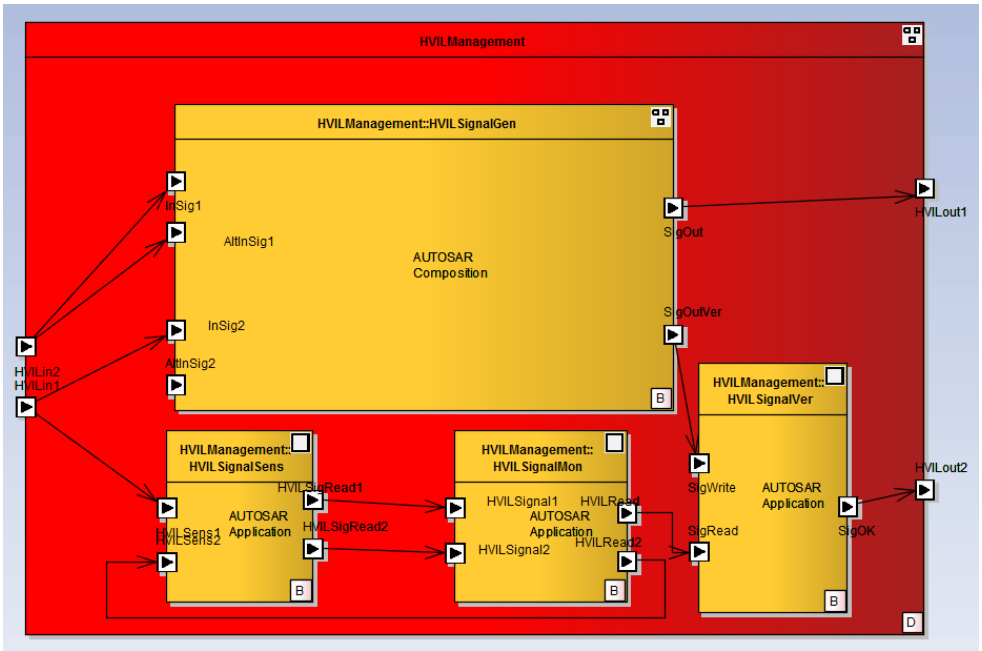

Fig. 7. Excerpt of SW architecture of HVIL Management Software Implementation within the System Development Tool

[5] Manfred Broy, Martin Feilkas, Markus Herrmannsdoerfer, Stefano Merenda, and Daniel Ratiu. Seamless Model-based Development: from Isolated Tool to Integrated Model Engineering Environments. *IEEE Magazin*, 2008.

[6] Christof Ebert and Capers Jones. Embedded Software: Facts, Figures, and Future. *IEEE Computer Society*, 0018-9162/09:42–52, 2009.

[7] Holger Giese, Stephan Hildebrandt, and Stefan Neumann. Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent. *LNCS 5765*, pages pp. 555 –579, 2010.

[8] Joerg Holtmann, Jan Meyer, and Matthias Meyer. A Seamless Model-Based Development Process for Automotive Systems, 2011.

[9] ISO - International Organization for Standardization. IEC 61508 Functional safety of electrical/ electronic / programmable electronic safety-related systems.

[10] ISO - International Organization for Standardization. ISO 26262 Road vehicles Functional Safety Part 1-10, 2011.

[11] Chorng-Shiuh Koong, Chihhsiong Shih, Pao-Ann Hsiung, Hung-Jui Lai, Chih-Hung Chang, William C Chu, Nien-Lin Hsueh, and Chao-Tung Yang. Automatic testing environment for multi-core embedded softwareATEMES. *Journal of Systems and Software*, 85(1):43–60, 2012.

[12] Tomislav Lovric, Manuel Schneider-Scheyer, and Samir Sarkic. SysML as Backbone for Engineering and Safety - Practical Experience with TRW Braking ECU. In *SAE Technical Paper*. SAE International, 04 2014.

[13] Georg Macher, Eric Armengaud, and Christian Kreiner. Automated Generation of AUTOSAR Description File for Safety-Critical Software Architectures. In *12. Workshop Automotive Software Engineering (ASE)*, Lecture Notes in Informatics, pages 2145–2156, 2014.

[14] Georg Macher, Eric Armengaud, and Christian Kreiner. Bridging Automotive Systems, Safety and Software Engineering by a Seamless Tool Chain. In *7th European Congress Embedded Real Time Software and Systems Proceedings*, pages 256 –263, 2014.

[15] Georg Macher, Muesluem Atas, Eric Armengaud, and Christian Kreiner. A Model-Based Configuration Approach for Automotive Real-Time Operating Systems. *SAE International Journal on Passenger Cars - Electronics and Electronical Systems*, 8(2):270–277, 2015.

[16] Georg Macher, Harald Sporer, Eric Armengaud, Eugen Brenner, and Christian Kreiner. Using Model-based Development for ISO26262 aligned HSI Definition. In *EDCC Conference Proceedings*, 2015.

[17] Roland Mader. *Computer-Aided Model-Based Safety Engineering of Automotive Systems*. PhD thesis, Graz University of Technology, 2012.

[18] Roland Mader, Gerhard Griessnig, Andrea Leitner, Christian Kreiner, Quentin Bourrouilh, Eric Armengaud, Christian Steger, and Reinhold Weiss. A Computer-Aided Approach to Preliminary Hazard Analysis for Automotive Embedded Systems. In *18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, pages 169 –178, 2011.

[19] Imran Rafiq Quadri and Andrey Sadovykh. MADES: A SysML/MARTE high level methodology for real-time and embedded systems, 2011.

[20] Ajitha Rajan and Thomas Wahl. *CESAR - Cost-efficient Methods and Processes for Safety-relevant Embedded Systems*. SpringerLink : Bücher. Springer Verlag, 2012.

[21] Guido Sandmann and Michael Seibt. AUTOSAR-Compliant Development Workflows: From Architecture to Implementation - Tool Interoperability for Round-Trip Engineering and Verification & Validation. *SAE World Congress & Exhibition 2012*, (SAE 2012-01-0962), 2012.

[22] Max Mauro Santos, Celso Mendes, Taysa Banik, Felipe Franco, Joo Neme, Wanderley Prado, Fernando Cerri, and Lauro Nunes. New approach of tools application for systems engineering in automotive software development. In *SAE Technical Paper*. SAE International, 03 2017.

[23] Giorgio Scuro. Automotive industry: Innovation driven by electronics. http://embedded-computing.com/articles/automotive-industry-innovation-driven-electronics/, 2012.

[24] Stefan Voget. AUTOSAR and the Automotive Tool Chain. In *DATE10*, 2010.