A novel design method for automotive safety-critical systems based on uml/marte

Article · September 2015

CITATIONS
4

READS
62

5 authors, including:

Ralph Weissnegger
Graz University of Technology
5 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Christian Kreiner
Institute of Electrical and Electronics Engineers
155 PUBLICATIONS **295** CITATIONS
SEE PROFILE

Kay Römer
Graz University of Technology
157 PUBLICATIONS **3,771** CITATIONS
SEE PROFILE

Christian Steger
Graz University of Technology
215 PUBLICATIONS **831** CITATIONS
SEE PROFILE

A Novel Design Method for Automotive Safety-Critical Systems based on UML/MARTE

Ralph Weissnegger*†, Markus Pistauer†, Christian Kreiner*, Kay Römer* and Christian Steger*
*Institute for Technical Informatics
Graz University of Technology (TU Graz), Austria
Email: (ralph.weissnegger, christian.kreiner, roemer, steger)@tugraz.at
†CISC Semiconductor GmbH, Klagenfurt, Austria
Email: m.pistauer@cisc.at

***Abstract*—The complexity of electric/electronic systems in today's vehicles is steadily growing. New challenges arise through highly distributed systems which interact with and have an impact on the physical world, so-called cyber-physical systems. There is a need for modeling languages like UML/MARTE to support engineers and managers throughout the whole design process to reduce costs and time to market. Especially when it comes to safety-critical systems, safety aspects must be handled on various abstraction levels from high level system description to detailed modeling of hardware and software. Not only functional but also non-functional requirements need to be taken into account here. In this paper, we present a seamless model-driven architecture approach to model safety-critical systems throughout the whole design phase of the functional safety standard ISO 26262. Furthermore, SysML is used to extend MARTE with semi-formal requirements to handle the issue with traceability. In order to demonstrate its efficiency, this methodology is applied to an industrial use case of a battery management system. The results show that MARTE is very suitable for modeling systems at any level of granularity in the automotive area, in compliance with functional safety.**

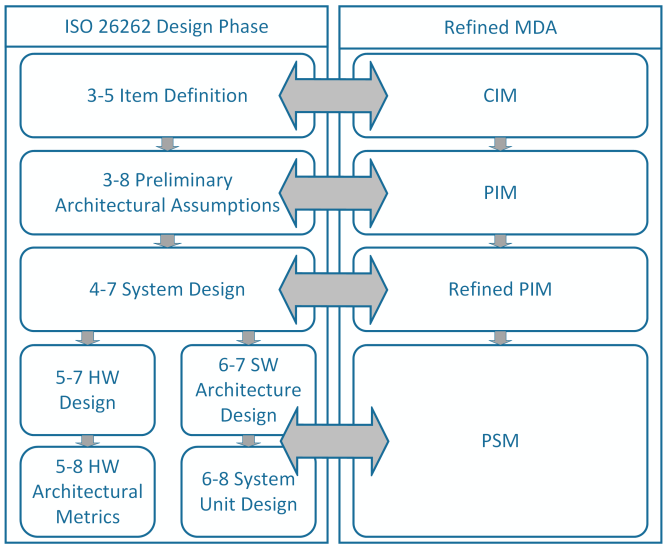

Fig. 1. ISO 26262 design phase to MDA mapping

I. Introduction

Today's cars consist of highly complex E/E systems with sensors and actuators networking with each other, in fact a car is now more or less a smartphone on wheels. It can be observed that there is a shift towards fully E/E cars, since traditional combustion engines are slowly disappearing. The sensing and controlling of these systems is the work of the highly distributed electrical control units (ECU) and it's no surprise that up to 100 of these micro-controller are currently integrated in an electric vehicle [1], [2].

Recent trends in the in-vehicle E/E architecture and new applications brought a rapid shift towards multicore, heterogeneous, networked, and reconfigurable systems. The design and development of such systems is extremely complex and imposes an enormous challenge for designers (hard- and software) from different domains in designing their applications. The next generation of such systems should be able to run in parallel on different parts (ECUs and/or processors, DSPs within a multicore architecture of a single ECU) of the system. Applications are going towards multimedia, infotainment, advanced driver assistance systems (ADAS), navigation and many more. This has an impact on design, development and management and in turn increases production costs and time to market. This has been acknowledged also on European industry level [3], [4].

With the growing complexity in the automotive area one aspect is turning out to be the key issue for future vehicle development: safety. This is especially the case whenever systems interact with and have an effect on the physical world, so-called cyber-physical systems, it is not longer sufficient to test a single behavior. The whole system must be validated as early as possible in the development cycle and at any level of granularity. This is also recommended by the ISO 26262 [5] standard for automotive E/E systems. The ISO 26262 is an adaption of the functional safety standard IEC 61508 and compliance is currently required for OEMs and suppliers of E/E systems. The ISO 26262 supports managers and engineers throughout the whole product lifecycle on different abstraction levels. As a variety of system assumptions and design solutions needs to be taken into consideration, a model-based approach is an important basis for engineers and multiple stakeholders. It helps designers to have a

quick and augmented view of the system and provides an effective way for communication, especially if systems are very complex and involve a number of teams in the design.

A way to model such systems is MARTE. This is an extended profile to UML2 and provides capabilities for modeling hard- and software, as well as timing and performance behavior. It is used at present by many semiconductor vendors and suppliers [6]. Today, MARTE is not very common in the automotive domain but with the newly electrification of vehicles and thus more and more components are related to E/E systems, MARTE could well help to save development costs and time in the future. Furthermore, it is the driven system-design language in the European Catrene-project OpenES [7]. OpenES is a European initiative to fill the gaps in today's system-design and to develop common solutions to stay competitive. A special focus is given to integral support for functional, but also non-functional requirements such as timing, thermal issues and power.

Another aspect of UML is that it is now supported by several commercial and open-source tools like Eclipse's Papyrus [8], that helps designers to model systems in UML and extensions like MARTE or SysML.

In this paper we present a way to model safety-critical systems on different levels of abstractions. We show that there are existing modeling languages that provides us with capabilities to represent the whole design flow of the functional safety standard ISO 26262 without compromises and helps us with additional features for safety analysis. We therefore use a refinement of the Model Driven Architecture (MDA), elaborated in the OpenES-project. We show that we can map the whole process from item definition and system design to hardware and software separation, to the model-based approach depicted in Fig.1. In our approach, each level in the ISO 26262 has an equivalent level in the MDA. This helps designers and engineers to keep a consistent view on all levels of the design phase. Furthermore we use the capabilities of SysML to model each requirement on different abstraction levels in the requirements phase to have a seamless allocation to our models and diagrams in the design phase. We also show that MARTE is very suitable for designing complex systems in the automotive area.

The paper is organized as follows: Section 2 presents the state of the art and related work. A short overview of functional safety and the safety lifecycle is given in Section 3. Section 4 describes the model driven architecture approach and the modeling languages in use. Section 5 presents our methodology applied to a case study for a battery management system. This is followed-up by the conclusion in Section 6.

II. Related Work

How to use MARTE in a co-design process is discussed in several papers [9], [10], [11]. They show how MARTE complies with the model-driven architecture and the defined abstraction levels. In these papers the issue of how to model hardware and software on different abstraction levels in the design process is also discussed. The paper authors also address the issues of modeling extra-functional properties and the mapping from software applications to platforms. However, they do not consider traceability to SysML requirements, nor do they take modeling of safety constraints into account.

The authors of [12] present a concept how to apply the ISO 26262 in the development of a safety critical system. They address the system level as contained in part 3 (concept phase) and part 4 (product development at the system level) of the functional safety standard, but do not take detailed hardware or software modeling into account. Furthermore they do not use standards like UML for system-modeling, nor are they able to maintain a seamless flow throughout the design phase. This approach also does not show how to add behavioral diagrams to the flow, nor are safe states or other behavioral functions defined. Traceability to structural and behavioral diagrams is only partially covered.

How to use SysML as representation of requirements in the automotive industry and the functional safety standard is discussed in [13], [14], [15]. The authors show how to model the requirements on different abstraction levels in a semi-formal way. Also the traceability between the different levels in the requirements phase of the safety lifecycle are handled. One approach [14] extends SysML to define requirements of safety-critical systems. In [15] also the allocation to structural and behavioral models is taken into account. This approach shows how to use SysML for requirements on different abstraction levels very well. Unfortunately this method does not cover the whole design phase of the ISO 26262 and confines traceability solely to UML/SysML diagrams. We will build upon this approach in our paper to cover all traceability aspects as recommended in the standard. As none of these approaches consider MARTE as detailed modeling language for hardware and software, we will show how to use SysML to maintain the traceability to MARTE models on multiple levels.

One language that is established in the automotive area is EAST-ADL [16]. EAST-ADL is a language for the development of vehicle embedded electronic systems and in combination with AUTOSAR, the initiative to standardize software development. The language was developed in the context of the ITEA cooperative project EAST-EEA and further projects like ATESST [17] and MEANAD [18]. Since EAST-ADL is included in Eclipse Papyrus also SysML requirements models can be used to define requirements [19].

The language is structured in five abstraction layers, each with a corresponding system behavior: vehicle level, analysis level, design level, implementation level and operational level. EAST-ADL is built on top of AUTOSAR and covers only the abstraction levels from vehicle to design level [20]. The implementation and operational levels are modeled in AUTOSAR which makes the top down traceability and also the traceability to the requirements, as specified is required by the ISO 26262 standard, very cumbersome and error prone. One purpose of the ATESST and MEANAD project was to provide capabilities to map the functional safety standard to EAST-ADL abstraction levels. These levels are not in compliance with the model driven architecture, nor does this approach address all the design-levels of ISO 26262. Furthermore this language lacks of referencing timing and performance properties or other extra-functional properties, which are addressed in detail in MARTE.

As more and more systems in the automotive domain are now related to real-time and embedded systems, bringing the safety standard to MARTE is a next step in the development of safety-critical systems.

III. Functional Safety

ISO 26262 is an adaption of the function safety standard IEC 61508 for automotive E/E systems. Since ISO 26262 is treated as state of the art in court, OEMs and their suppliers are required to comply with this standard today. It addresses hazards caused by safety related E/E systems due to malfunction and covers functional safety aspects through the whole lifecycle. It governs the identification, design, implementation and testing in form of an industry-standard V-model, called automotive safety lifecycle. The standards also provides an Automotive Safety Integrity Level (ASIL) analysis to specify the items necessary safety requirements for the development to hardware and software components. A safety goal is derived for each hazardous event with an ASIL-classification. The safety goals are the source for the whole chain of the safety lifecycle. This in turn results in a safety case, to show that the system is acceptably safe. The safety case is used to collect and present evidence, to support safety claims and arguments. In this work we address the left side of the V-model, from concept phase to product development. Since the right side addresses verification, testing and production it is not handled by our approach and is beyond the scope of this work. We use an MDA approach to demonstrate how we can model safety aspects in the design phase of the automotive safety standard.

IV. Model-Driven Architecture

A. Modeling languages

UML is a modeling standard of the OMG (Object Management Group) [21]. It is a graphical representation for specification and documentation of software and other systems. It delivers a complete view of the system, their individual components and the interaction between them. UML has different types of structural (e.g. Class, Component, Composite Structure) and behavioral (e.g. Activity, UseCase, State Machine) diagrams. Although this language is well suited for developing software, it provides no capabilities to design hardware and non-functional properties.

Another approach is SysML [22] as domain-specific modeling language. It uses a subset of UML2 and provides additional extensions to describe complex systems in system engineering. SysML supports UML2 by two additional diagrams (requirements, parametric) for requirements-engineering and performance-analysis. It provides a good mechanism for allocating requirements to components or behavioral diagrams, but is inaccurate in modeling hardware and resources.

MARTE was defined as an adaption from the OMG to address the shortcomings of modeling platforms in UML. MARTE [23], [24] is a domain-specific modeling language intended for model-based design and analysis of real-time and embedded software of cyber-physical systems. MARTE is defined as a profile in UML2 and provides additional mechanisms for modeling real-time systems, which are missing in UML. MARTE has the advantage of precise hardware and software resources in the form of *HRM* and *SRM* stereotypes. In addition it is possible to allocate software applications to hardware resources with the help of the MARTE allocation mechanism. MARTE follows the philosophy of cyber-physical systems to deal with whole systems rather than a set of specialized parts. This is also recommended by the ISO 26262 for the design of safety-critical systems.

The MDA approach has gained more importance as a result of a trend to pursue more formal modeling languages and greater exploitation. We can see from the definition of the different sub-profiles that MDA is also anchored in the MARTE language. The importance of this development has also been acknowledged on European level, where MARTE and the MDA approach have a significant part in the Catrene project OpenES. For our approach we only use standardized MARTE elements.

B. Refined model-driven architecture

Since the levels of the standard MDA by the OMG were not adequately specified and lacked formal definition in the OpenES-project, partners elaborated a refinement of the MDA-approach (Fig. 1). This figure illustrates our mapping between the different levels in the design phase of ISO 26262 and the levels in the MDA. In our methodology, each level of the functional safety standard has an equivalent level in the MDA approach. The detailed definition of each level is given below:

Computation Independent Model (CIM) - aims at providing a system level view, mainly focusing on its functional structure. It does not specify any information on how the functionality will be implemented. In particular there

is no hardware software identification. Moreover, this kind of model is not precise enough to execute models. Despite the lack of an explicit or implicit model of computation, CIM can include system level use cases showing synchronous and asynchronous communications between the different functional blocks. CIM is too abstract to specify any non-functional properties.

Platform Independent Model (PIM) - includes CIM capabilities with additional behavioral models like UML state machines or activity diagrams. Moreover, non-functional properties like timing, power, thermal issues or safety can be expressed at this level of refinement. PIM are not fully executable models, only a part of the whole system can be detailed more precisely. If the behavior is not directly expressed in diagrams, the models can be referenced to existing implementation code. These models can be used for an early and high-level simulation of the system-behavior. A PIM shows that part of the specification, which does not change from one platform to another.

Refined PIM - has been explicitly identified to fit with the OpenES sub-system definition concept. It consists in a functional decomposition of the PIM with a granularity detailed enough to allocate each of its blocks to a single hardware or software execution resource. In other words, a PIM functional block cannot be allocated on several execution resources. The functional blocks should be split beforehand into different sub-functionalities. Furthermore, their communication interfaces should be identified before mapping them onto different execution resources.

Platform Specific Model (PSM) - encompasses several aspects. It should first contain elements that will describe the execution platform, including hardware and software execution resources. On the hardware side, the model can contain low level details, such as registers and memory map information. On the software side, the execution platform description can specify OS-specific information, such as tasks, scheduling algorithms or middleware services. Complementary to those platform description aspects, the PSM can contain a new refinement of the PIM model where platform independent functional components are transformed into platform specific components, with explicit references to the execution resources services. The PSM part can be partially generated by the Allocation Model described below.

Allocation Model - is an intermediate step between the refined PIM and the full PSM. It expresses how refined platform-independent functional components can be allocated onto hardware or software execution resources. It implies that part of the PSM already exists, to identify and reference those execution resources. It can be the input of extra-functional properties analysis tools to verify if a given mapping will allow meeting expected extra functional property constraints. It can also be the source of code generation or model transformation to obtain detailed platform-specific application model. Since the allocation model is more like a link to a higher detailed model, it is not a separate level in our approach.

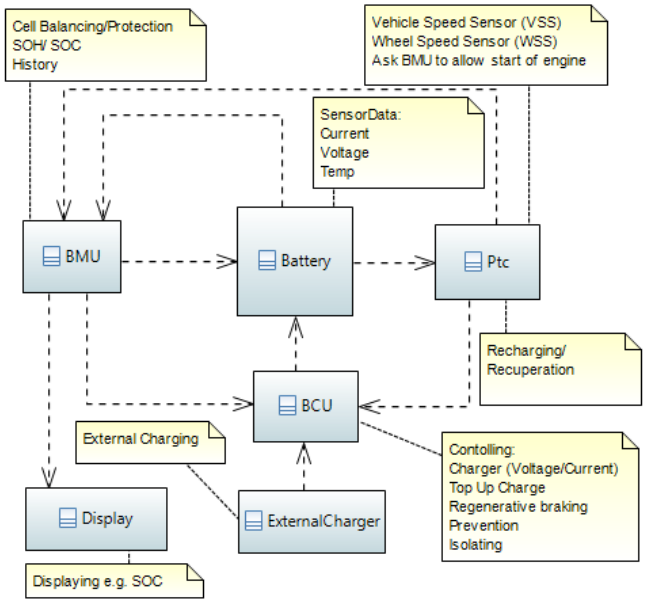

Fig. 2. **Item definition :** high-level functional view

V. Example Case Study: Battery Management System

To show how we can use MARTE and the MDA approach through the whole design phase of the functional safety standard we demonstrate this by an industrial example of a battery management system with recuperation features provided by CISC Semiconductor. As more and more vehicles are now powered by Li-Ion-batteries, the challenge for engineers to ensure reliability and fault-tolerance of batteries is also greatly increasing. Problems with overheating or even explosions have been frequent in the past. The mainly cause of these problems was excessively high energy intake from regenerative braking or harsh environmental conditions. Management systems and mechanisms are thus essential to assure that persons are not put at risk and that no damage is caused. Safety mechanisms such as redundant and diverse measurements of the temperature and voltage of battery-cells decrease the occurrence of single-point, residual and multiple-point faults. Also the multiple and diverse calculation of sensor-data is an important measurement at a high integrity level. Since it is beyond the scope of this paper to examine all safety aspects of the item, we focus here on monitoring the state of the battery. The parts of the Battery Management System are explained in detail below:

Battery Monitoring Unit (BMU) - is the main controller of the battery. It measures different values coming from sensors of the battery-cells. The BMU computes the State-Of-Charge (SOC), State-Of-Health (SOH) and is responsible for cell balancing, cell protection and demand management of the battery. It also controls a hardware switch, which connects the battery to the electric motor.

Battery Control Unit (BCU) - controls the voltage and current profile of the charger output during the charging process. Besides controlling the charge of the external charger it monitors the regenerative braking charges and dumps it when the battery is fully loaded. If a fault occurs the battery can be isolated or the BCU sends a signal to the PTC to restrict the speed limit.
Power Train Controller (PTC) - is the main contactor between the battery and the electric motor. It controls vehicle and wheel speed and instructs the BMU, which monitors the state of the battery, to start the motor.
Battery - consists of 12 cells, connected in series. Each cell has a temperature sensor and connections to measure the voltage.
Display - shows the current SOC and SOH and warns the driver if a critical threshold is reached.

The first and most essential step in the development process in the context of the functional safety standard is to define the item. The definition of an item is a system or array of systems to implement a function at vehicle level to that the safety standard is applied. A system is a set of elements containing at least a sensor, controller and actuator, whereby an element can be a hardware or a software part. Figure 2 shows the item definition modeled as functional blocks by means of a UML composite structure diagram and informational flows. This provides a good view of the whole item at CIM-level with boundaries and interfaces to the environment. On this level we show how the functional blocks communicate with each other, but do not specify how the functionality will be implemented. A safety goal is derived with the help of the item definition and the hazard analysis and risk assessment, for each hazard. Each safety goal has its own ASIL-level that classifies the severity, exposure and controllability of the operating scenario. For this example an ASIL C safety goal was stated: *"excessive battery temperature must be avoided"*. The safety goals provide the basis for the functional safety requirements (FSR) specified in the functional safety concept (FSC), (ISO 26262, Part 3-8). A derived FSR for this example would be *"A BMS shall monitor and control the battery. The battery must operate in working range"*.

The ISO 26262 standard recommends different methods and design techniques to achieve safety on certain ASIL-levels. In this example we choose the heterogeneous duplex pattern to increase the reliability and availability of the system. The heterogeneous duplex pattern uses extra and diverse hardware components and has the advantage of being able to handle not only random but also systematic faults. Different hardware components will lead to the hardware reacting in different ways and also increasing the coverage of common cause failures. As our item has an ASIL C classification, the standard recommends using two independent and diverse signals to control the battery-cells, to achieve redundancy and diversity. We combine this with the decomposition-mechanism of the ISO 26262, which allows us to decompose

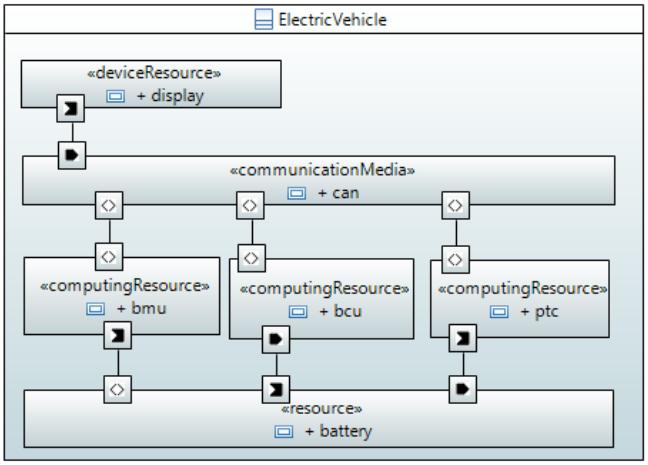

Fig. 3. **Architectural assumption:** a pre-version of the system design, no information about the used platform is given

our ASIL C sensor into two ASIL B(C) sensors in order to have the same classification but also to increase reliability by using independent redundant and diverse measurements. In the next step we make a refinement of our previously defined FSR to *"A safe state will be switched to, if the two measurements deliver different values of the battery-cells (plausibility-check)"*. The safe state is the desired behavior of the system in the event of a fault. It ensures the safe operation of a system. This corresponds in our case to a safe state such as *"Reduce the power (degradation function) or even shutdown the connection from the battery to the motor"*.

The result of the functional safety concept is the preliminary architectural assumption illustrated in Fig.3, a pre-version of the system design, which is the actual solution to the functional requirements. In this diagram the relationship between the different components is more explicitly expressed and gives a more detailed view of the system. Moreover MARTE flow-ports (in,out,in_out) of component instances are also present, as well as connectors between them. On this abstraction level (PIM) we are independent of the actual implementation and therefore do not specify any technical details nor platform on which the function may be implemented. With the use of the MARTE general resource model (GRM) we are able to make our first assumptions regarding the system design. With the stereotype *"communicationMedia"*, properties such as capacity or transmission-mode for the CAN bus can be defined. The *"computingResource"* stereotypes is used in order to also model processing resources at a very high level of abstraction with no concern about the details of CPU speed or memory capacity.

At this stage additional behavioral models like activity diagrams or state machines are also added to describe the behavior of the system. These behavioral diagrams are allocated to the blocks of the architectural assumption. In Fig.4 the safe state is modeled by means of nodes and

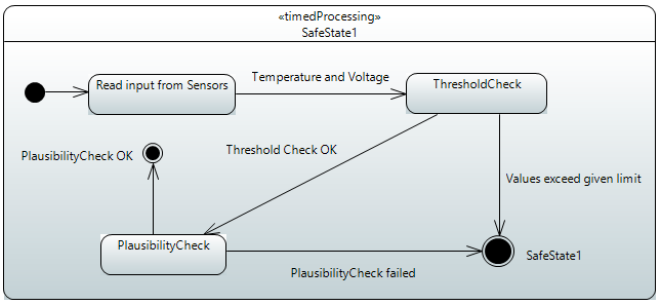

Fig. 4. **Safe state:** activity diagram with extra-functional timing properties

edges. First assumptions regarding timing constraints can also be considered in the activity diagram with the help of *"timedProcessing"* stereotypes. This is not shown in the figure but it is specified by a duration of *"value=30;unit=ms"* in the value specification modeling format (VSL). A more precise timing behavior of each node in the diagram can be made in a subsequent refined step, where more detail is given from the hardware software interface (HSI). An example would be the detailed description about system reaction or fault reaction time. It is the advantage of MARTE to capture timing information by means of qualitative and quantitative annotations in the description of the behavior. These expected behavior specifications can provide important inputs to perform model validation in later phases of the process. Another aspect of behavioral diagrams and timing constraints is to use them later for the generation of testbenches for simulation-based verification purposes. Furthermore they can be used for comparison with simulation results.

Now that the pre-version of our system design has been completed, the MARTE-models are coupled with preexisting implementation-models written in SystemC-TLM. This allows us to use a high-level simulation to have a closer look at the dependencies between the different components. For this purpose we use SHARC [25], an Eclipse-based tool under development for modeling and simulation of cyber-physical systems at different levels of abstraction. SHARC is an enhancement of SyAD/SIMBA [26] and uses co-simulation of various distributed components written in SystemC, Matlab or VHDL. It also allows us to switch to lower implementation levels of single components in the system, such as RTL-level simulation. This is particularly important for the verification of hardware safety mechanisms with methods like fault-injection and also recommended in the hardware design verification methods by the ISO 26262. The outcome of the preliminary architectural assumption, HSI, FMEDA/FTA and the hardware architectural metrics results in achieving the technical safety requirements (TSR). A FMEA/FTA on UML-models can be performed by approaches described in [27] or [28]. In this case-study example the TSR1 is defined as, *"Plausibility-checks every 30 ms of two analog sensors (temperature and voltage). If the difference of* $10\,^{\circ}\mathrm{C}$ *is above the tolerance threshold for more than a certain time, go to*

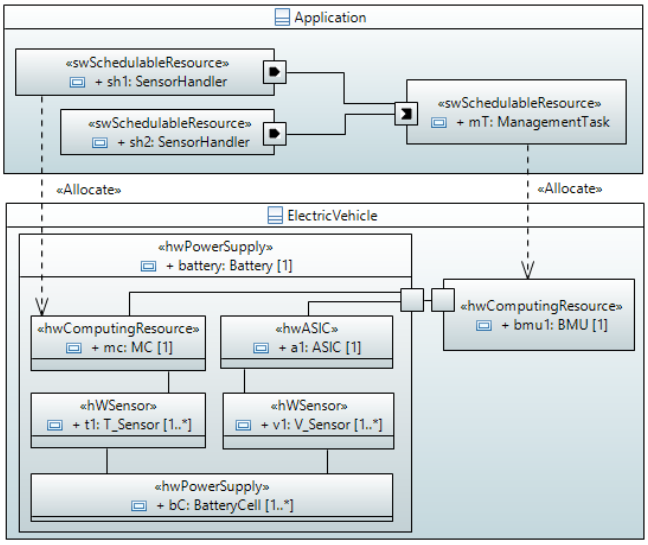

Fig. 5. **System design:** applications are allocated to the used hardware platforms

safe state". This in turn results in a first version of our system design. At this stage we make our first assumptions about what we are going to realize in hardware and software. As defined in the OpenES refined PIM, the systems must be split to subsystems if no allocation from each block to hard- or software is possible (atomic). We now refine the architecture by adding two ASIL B(C) hardware sensors, as a result of our foregoing step where the ASIL C sensor was split into two ASIL B(C) sensors with the decomposition-mechanism. We achieve a vertical traceability throughout the ISO26262 levels by aggregating the components in the UML class diagram. For the modeling of our hardware-platform we use the MARTE hardware resource model (HRM). In Fig.5 the battery is split into a multiplicity of battery-cells, each linked to a voltage and temperature sensor tagged with MARTE *"hwSensor"*. Included are also two processing units, a micro-controller and an ASIC tagged with *"hwComputingResource"* and *"hwASIC"*, which are specializations of *"hwResource"* stereotype. These two processing units are used to calculate the data coming from the sensors, independent and redundant. Furthermore, they increase the fault tolerance of the system. To show that we have now a clear separation between software and hardware, we allocate the sensorhandler- and the management-task to our computing resources, micro-controller, respectively battery management unit. At this point where applications are allocated to platforms, a schedulability analysis in MARTE like described in [29] can be performed. This enables an early analysis of design alternatives before committing to a particular design for implementation. This is especially important in the design of safety-critical systems where resource-handling must be carried out very carefully. It must be ensured that common tasks like multimedia applications are not influencing or locking resources from safety-critical tasks that must perform in a given time. A

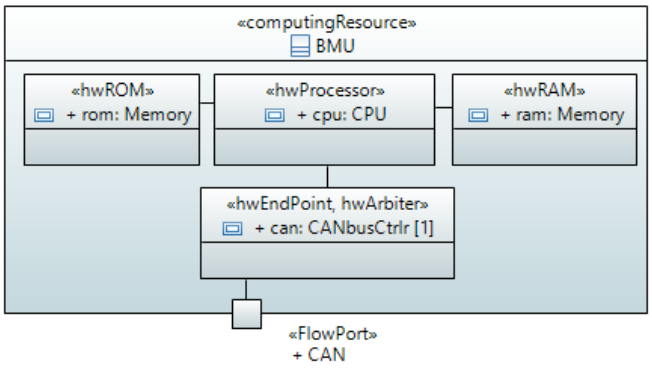

Fig. 6. **Hardware-description:** detailed structure of the battery management unit

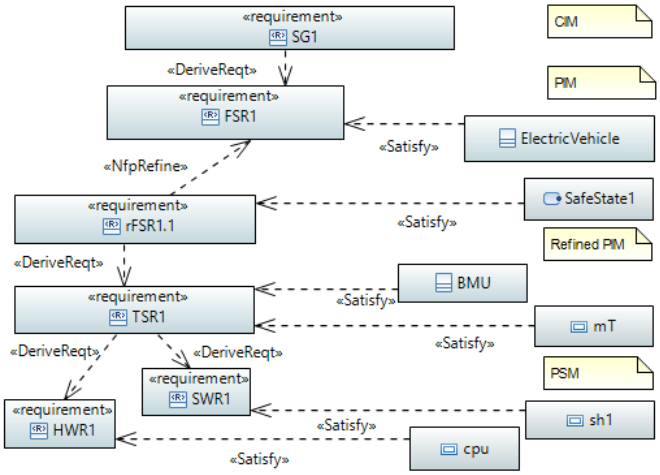

Fig. 7. **Requirements allocation to components and diagrams:** vertical and horizontal traceability

task may be switched to another processing unit that needs his own protected memory. This approach allows to analyze worst-case scenarios of different tasks allocated to processor cores.

Now that the functional blocks of the system are split into hardware and software components the definition of the requirements for hardware and software can be made. The final level (PSM) of the design approach, the hardware and software architectural design, is derived from these requirements.

The MARTE profile provides a set of concepts for hardware modeling that can be used to define very detailed models of computing hardware. In this example the hardware is designed by means of composite structure diagrams to graphically show the inputs/outputs and interfaces, depicted in Fig.6. The BMU designed on system level is now split into four detailed components: CPU, ROM, RAM and CAN. With the help of MARTE *HRM*, the blocks are tagged with specialized stereotypes to specify the properties. Also additional safety relevant properties required for the hardware design such as failure rate, safety-related or not, hardware safety mechanism and associated diagnostic coverage can be annotated to the hardware description. In this approach we also use a self-defined MARTE profile, for describing the hardware in the IP-XACT [30] standard. The software tasks are described as usual in the software design by traditional class diagrams. The MARTE software resource model (*SRM*) and also the high level application model (HLAM) are sufficient for defining constraints to our system including message size or memory size. As the next step operations are added to the management task for the BMU such as ThresholdCheck(), Display() or PlausibilityCheck(). Also timing-attributes tagged with MARTE timing notations are added to the task. As more and more details are attached, this models are latter used for automatic code generation for hard- and software in SystemC [31]. The Gaspard2 framework uses MARTE models to generate RTL code for synthesis or TLM code for simulation on a higher level of abstraction. This closes the gap from early safety-critical requirement analysis and system specification to simulation and synthesis.

A decisive issue which is often mentioned in the ISO 26262 standard is assuring traceability. Traceability starts with the safety goal and runs through the entire requirement and design phase, from derived requirements like FSR and TSR to behavioral and structural models. Traceability must be assured at each level of the lifecycle to support not only engineers and managers from different domains but also the argumentation in the safety case. In our approach the requirements tree is modeled in SysML and the relationship between each requirement and level is linked with SysML *"derived"* stereotypes. If the description of the requirement is not detailed enough, another requirement can be linked with *"refine"* stereotype. In Fig.7 we show that we not only achieve vertical, but also horizontal traceability through the whole lifecycle by allocating each requirement to the associated component or diagram. Another approach is to represent SysML requirements in a specified spreadsheet-like table-view in Papyrus. This provides a good overview of all requirements and their *"satisfiedBy"* relationship. This table is fully dynamic and immediately updated if relationships between requirements and models are added or modified. It also scales up for larger systems.

VI. Conclusion

In this paper we demonstrated a methodology to model safety-critical systems at any level of granularity in the design phase of the functional safety standard ISO 26262, with the model-driven architecture approach (MDA). The specification and refinement of the MDA-levels were elaborated in the European Catrene-project, OpenES. A UML-profile MARTE for real-time and embedded systems was used to model safety aspects on all abstraction levels of the design-phase of the ISO 26262. We show that MARTE is very suitable for modeling E/E systems in the automotive area without using any extended

self-defined UML-profiles. Through the use of standardized modeling languages we achieve a high reusability with other tools in this domain. Furthermore, SysML was used for horizontal and vertical traceability of requirements to components and behavioral models. With the link to MARTE diagrams we achieve a very high traceability level as demanded by ISO 26262. We showed the efficiency of this approach by applying the methodology to a battery management system. Future work will deal with the simulation of design models for verification of safety-critical systems. With the tool SHARC, MARTE-models will be linked to implementation models in SystemC or Matlab on various abstraction levels. In a next step, behavioral models will be used to automatically generate testbenches for simulation based verification.

ACKNOWLEDGMENT

The approach presented above was experimented in the OpenES Catrene- project: CA703 - 2013 research program supported by CISC Semiconductor.

REFERENCES

[1] R. N. Charette, "This Car Runs on Code - IEEE Spectrum," 2009. [Online]. Available: http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code

[2] Etas, "Electronic Control Unit (ECU) - Webinar Basics of Automotive ECU ETAS Embedded Systems Consulting Embedded Software , AUTOSAR and Safety Consulting Chandrashekara N (ETAS / ESC) Lead Consultant Embedded Systems," pp. 1–30, 2014.

[3] Gartner, "Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020," 2013. [Online]. Available: http://www.gartner.com/newsroom/id/2636073

[4] ARTEMIS Industry Association, "2014 MultiAnnual Strategic Research and Innovation Agenda for the ECSEL Joint Undertaking," Tech. Rep., 2014. [Online]. Available: http://www.smart-systems-integration.org/public/documents/publications/2014ecselmasriapartc.pdf

[5] ISO, "Road vehicles Functional safety Part 1: Vocabulary," 2011. [Online]. Available: https://www.iso.org/obp/ui/\#iso:std:iso:26262:-1:ed-1:v1:en

[6] J. Medina, "The UML Profile for MARTE: modelling predictable real-time systems with UML."

[7] Catrene, "OpenES CATRENE Project: CA703 - 2013," 2013. [Online]. Available: http://www.ecsi.org/openes

[8] Eclipse, "Papyrus," 2015. [Online]. Available: https://www.eclipse.org/papyrus/

[9] L. G. Murillo, M. Mura, and M. Prevostini, "MDE Support for HW/SW Codesign: A UML-based Design Flow," *Advances in Design Methods from Modeling Languages for Embedded Systems and SoCs*, vol. 63, pp. 197–212, 2010.

[10] J. Vidal, F. D. Lamotte, G. Gogniat, P. Soulard, and J.-P. Diguet, "A co-design approach for embedded system modeling and code generation with UML and MARTE," *2009 Design, Automation & Test in Europe Conference & Exhibition*, 2009.

[11] A. Koudri and D. Aulagnier, "Using marte in a co-design methodology," *UML Workshop at Date'08*, 2008.

[12] W. Taylor, G. Krithivasan, and J. J. Nelson, "System safety and ISO 26262 compliance for automotive lithium-ion batteries," *2012 IEEE Symposium on Product Compliance Engineering, ISPCE 2012 - Proceedings*, pp. 6–11, 2012.

[13] D. D. Ward and I. Ibarra, "Development Phase in Accordance with ISO 26262," *System Safety Conference incorporating the Cyber Security Conference 2013, 8th IET International*, pp. 1–6, 2013.

[14] M. Adedjouma, H. Dubois, K. Maaziz, and F. Terrier, "A Model-Driven Requirement Engineering Process Compliant with Automotive Domain Standards," *Third Workshop on Model Driven Tool and Process Integration (MDTPI), Paris, France, June 16, 2010 Proceedings*, 2010.

[15] L. Muat, M. Hübl, A. Buzo, G. Pelz, L. Musat, M. Huebl, A. Buzoee, G. Pelz, S. Kandl, and P. Puschner, "Semi-formal Representation of Requirements for Automotive Solutions using SysML," in *2014 Forum on Specification & Design Languages (FDL)*, 2014.

[16] "EAST-ADL Association," 2013. [Online]. Available: http://www.east-adl.info/

[17] "ATESST," 2010. [Online]. Available: http://www.atesst.org

[18] "maenad.eu," 2014. [Online]. Available: http://www.maenad.eu/

[19] J. L. Boulanger and Q. Van Dao, "Experiences from a model-based methodology for embedded electronic software in automobile," *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications, ICTTA*, pp. 1–6, 2008.

[20] P. Cuenot, D. Chen, S. Gérard, H. Lönn, M. O. Reiser, D. Servat, C. J. Sjöstedt, R. T. Kolagari, M. Törngren, and M. Weber, "Managing complexity of automotive electronics using the EAST-ADL," *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, no. Iceccs, pp. 353–358, 2007.

[21] "Object Management Group (OMG)," 2015. [Online]. Available: http://www.omg.org/

[22] "SysML.org: SysML Open Source Specification Project," 2014. [Online]. Available: http://sysml.org/

[23] "The UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems — www.omgwiki.org/marte," 2013. [Online]. Available: http://www.omgmarte.org/

[24] B. Selić and S. Gérard, *Modeling and analysis of real-time and embedded systems with UML and MARTE*, 2014.

[25] R. Weissnegger, M. Pistauer, and C. Steger, "Program & Book of Abstracts," in *The 10th International Conference on Scientific Computing in Electrical Engineering SCEE 2014,Program & Book of Abstracts*, Wuppertal, 2014, pp. 57–58.

[26] C. Trummer, C. M. Kirchsteiger, C. Steger, R. Weiß, M. Pistauer, and D. Dalton, "Automated simulation-based verification of power requirements for systems-on-chips," *Proceedings of the 13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2010*, pp. 8–11, 2010.

[27] F. Mhenni and N. Nguyen, "Automatic Fault Tree Generation From SysML System Models," 2014.

[28] H. Kim, W. E. Wong, V. Debroy, and D. Bae, "Bridging the Gap between Fault Trees and UML State Machine Diagrams for Safety Analysis," *2010 Asia Pacific Software Engineering Conference*, pp. 196–205, 2010.

[29] L. S. Indrusiak, I. Quadri, I. Gray, N. Audsley, and A. Sadovykh, "A MARTE Subset to Enable Application-Platform Co-simulation and Schedulability Analysis of NoC- based Embedded Systems."

[30] SPIRIT, "IEEE SA - 1685-2009 - IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows." [Online]. Available: http://standards.ieee.org/findstds/standard/1685-2009.html

[31] E. Piel, R. B. Atitallah, P. Marquet, S. Meftali, S. Niar, A. Etien, J. J.-L. Dekeyser, P. Boulet, and I. Europe, "Gaspard2: from marte to systemc simulation," *DATE'08 Workshop on Modeling and Analysis of Real-Time and Embedded Systems with the MARTE UML profile*, vol. 8, pp. 1–6, 2008.