# Applying Model Based Techniques for Early Safety Evaluation of an Automotive Architecture in Compliance with the ISO 26262 Standard

P. Cuenot[1], C. Ainhauser[2], N. Adler[3], S. Otten[3] , F. Meurville[4],
1: Continental Automotive France, 1 av. Paul Ourliac, 31036 Toulouse, France
2: BMW Car IT, Petuelring 116, 80809 Munich, Germany
3: FZI Research Center for Information Technology, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany
4: Valeo Group Electronics Expertise and Development Services, 2 rue André Boulle, 94046 Créteil, France

**Abstract**: In 2011, the automotive industry introduced the application of a standardized process for functional safety-related development of automotive electronic products. The related international standard, ISO 26262 functional safety for road vehicles, has high demands on process documentation and analysis. Within an engineering context this challenges the tremendous increase of complexity for modern automotive systems and high productivity demands for industrial competiveness purpose.

Model based development techniques based on an Architecture Description Language (ADL) has been identified as the best candidate to manage the system complexity and the related safety analysis with the benefit of formal description and capabilities for test automation. The proposed concept relies on the definition of a compositional error modeling approach tightly coupled with the system architecture model, capable to analyze the software and hardware architectures and implementations. This paper explains the results of the language extension based on the EAST-ADL and AUTOSAR domain model in terms of early safety evaluation of an automotive architecture, automating the qualitative and quantitative assessment of road vehicle products as claimed by the application of the ISO 26262.

**Keywords**: Automotive Architecture Description Language, ISO 26262, Functional Safety Analysis.

## 1. Introduction

The international standard ISO 26262 [1], published in November 2011, is the adaptation of the generic safety industry standard IEC 61508 [2] to comply with specific needs of the automotive industry for development of road vehicles of less than 3.5 tons. It defines the process for the overall safety lifecycle of electrical and/or electronic (E/E) systems and specifies safety-related activities to be performed during the development cycle. It emphasizes a system engineering approach with the delivery of process-related work products applicable on each engineering level such as system, hardware, software and on supporting processes.

The relevant safety analysis activities start with the item definition during vehicle development and progress with the evaluation of ability to mitigate or avoid hazards in identified operational situations. This evaluation allows the identification of the corresponding safety goals with a classified Automotive Safety Integrity Level (ASIL), in order to initiate the process-based requirements. Subsequently systematic analysis of causes and effects of the system malfunctions shall be initiated on preliminary safety analysis. During system development, safety measures have to be defined to control, reduce or avoid effects of the malfunctions of the system components. Faults inside components can be random hardware faults or systematic faults. The analysis of causes and effects of these faults has to be performed in order to evaluate if they can violate a safety goal. Technical safety mechanisms shall be specified to detect and control the components faults and mitigate their effects to prevent a system malfunction. Clause 7.4.3 of ISO 26262, Part 4 “Product development at the system level” [1], requires deductive and inductive methods to perform safety analyses. These analyses have to be completed down to the level of hardware and software components as claimed in Part 5, Clause 7.4.3 and Part 6, Clause 7.4.13, respectively for “product development” at “hardware” and “software level” [1]. Moreover, Part 5 demands evaluation of the hardware architectural metrics and of the safety goal violation due to random hardware failures.

In actual automotive engineering practice, the safety evaluation of the system architecture is performed lately in the product development. Additionally, due to the increased product complexity and the large amount of components, there is a high risk of inconsistency between the safety analysis and the designed product. This error-proneness is increased by the use of different tools for engineering design and safety/reliability analysis.

Therefore, there is a strong need for novel methodologies to facilitate safety design and safety verification activities in an iterative and intertwined process. These methodologies shall close the gap between the definition of the architecture elements and their associated malfunctions with their failure

propagations over the architecture. This eases the definition and the evaluation of the impact of safety mechanisms integration. The required quantification of the hardware metrics, Clause 8 and 9 of ISO 26262 Part 5 [1], shall be aided by methodology through organizing the failure rate of the set of hardware electronic parts to facilitate safety hardware architecture upgrade.

From software side, recently, the automotive industry adopted the AUTOSAR [3] standardized architecture supported by model based technology for describing software component implementation and integration, and the ECU resource configuration. Release 4.0 has been delivered in late 2012, containing already a well-defined set of safety mechanisms responsible for error detection (e.g. watchdog manager) and error handling (e.g. diagnostics or error reporting to the application layer). Moreover, the architecture description language EAST-ADL [4] designed to support the vehicle architecture and the control of engineering artifacts (such as requirement, variant management control, etc...) has been harmonized with the AUTOSAR architecture and initiated the direction for supporting safety analyses in the EAST-ADL version 2.1 [4].

Thanks to the use of model based techniques, we propose to extend and complement these state of the art architecture languages, to define new methodologies able to support, accelerate and pre-assess safety analyses complying with ISO 26262 qualitative and quantitative assessments from the architecture level down to the level of the hardware and software implementation.

Our paper is structured as following. Section 2 introduces the state of the art of model based safety analysis that has been considered during this work. In section 3 the concept of error modeling and fault propagation to be able to perform safety analyses on automotive products is presented. Section 4 describes the proposed continuous methodology for safety evaluation across different abstracted representations of automotive products in compliance with the ISO 26262 process. Section 5 illustrates the use of the methodology in the context of the hardware architecture supported by a concrete example. Section 6 explains the context for appliance of safety analysis for an AUTOSAR software platform. Section 7 illustrates a case study and depicts a short overview of a tool prototype implementation for this methodology. Finally, conclusions draw the consideration of the designers' as end-users' perspectives, and the description of future work in Section 8.

## 2. Related Work

State of the art methods and tools for model based safety analysis have been introduced and are current practice in aeronautics industry. The actual civil avionics recommendation standard for safety assessment ARP4754 [5] and the associated methods in ARP4761 [6] toward model techniques have been largely investigated during this decades. In particular during the ESACS project [7] the use of AltaRica [8] formal language was evaluated, lessons learnt based on Airbus aircraft safety analysis capitalized [11], and the benefit for the application of the ARP standards demonstrated. On the top of practice for E/E architecture assessment, another usage of AltaRica was proposed in [12] targeting multi-physicals systems. Nevertheless, most of these successful model based analyses only consider system at equipment level, where the equipment fulfills only a single functionality. Therefore these proposed methods are difficult to apply at software or hardware implementation level due to combinatory explosion of basic technology.

Another approach was proposed in the context of the ASSERT project [13], by automatic building of dependability-oriented analytical models from high level AADL [14] architecture models. Thanks to AADL error model, Markov analysis can be performed and fault trees can be generated and analyzed [15]. Advanced experiments were performed on Integrated Modular Avionic (IMA) architecture, having similar software architecture than AUTOSAR. The complexity of automotive ECU integrating tremendous electronics parts with large diversity (e.g. microprocessors or ASIC's) cannot be standardized as for IMA approach by high level modeling.

Related to the automotive research project ATESST2, the EAST-ADL [4] architecture language description was extended to support safety analyses thanks to the connection of failure synthesizer HiP-HOPS [17]. The proposed methodology allows performing cut set fault tree analysis (FTA) and failure mode and effect analysis (FMEA) from analysis of the automated generation of faults. Meanwhile, such proposal for analysis is mainly functional oriented to support system modeling. But it does support neither software implementation analysis nor hardware metrics calculation.

Focusing on hardware evaluation, the work of Jeon et al. [18] describes the process of performing hardware architectural metrics. However, there is neither a semi-formal or formal method nor a prototype implementation available. Additionally, the proposed methodology discussed by K. Svancara [19] for residual risk evaluation of the violation of a safety goal using Failure Rate Class method (alternative method proposed by ISO 26262 instead of using PMHF) brings strong benefit to ISO 26262. Integration in model based environments would support automation of the analyses.

In the article of [20], a model-based methodology for the implementation of hardware assessment based on detailed hardware level according to ISO 26262 is

proposed. It shall be extended to permit hardware metrics evaluation at the architecture level.

The current work of the European project 'Safe Automotive soFtware architecture' (SAFE) [21] organized in the framework of ITEA2, aims to overcomes these gaps applying model-based approach for automotive safety-related systems.

## 3. SAFE Error Modeling Concept

The selected modeling languages for SAFE are EAST-ADL and AUTOSAR. Relationship between these two domain models have been largely discussed and documented in the ATESST project during the elaboration of the EAST-ADL language.

For the description of the automotive system architecture on different abstraction levels, AUTOSAR represents the implementation layer for software and additionally has to be improved to sketch hardware implementation. EAST-ADL has to be enhanced to represent separate but interrelated architecture views of hardware and software component.

Regarding error modeling, the proposed approach extends the structural architecture elements on each abstraction level, by a straight forward relation. The related *error model* captures the malfunctions and errors relations between the elements. In details, the error model describes the black-box view in terms of error propagation for the referenced structural elements. Thus, the error model reports the visible interfaces *external faults* for incoming faults and *external failures* for output failure. The error model is also associated with white-box information as the error behavior of an element. In this case, the internal details of the structural element are known and the respective *internal faults* (e.g. random internal hardware faults) as well as the *process faults* (systematic faults such as design, implementation, installation, operation and overstress faults) can be described. It is possible to describe with semi-formal notation how malfunctions as external faults, internal faults and process faults are related with external failures. This behavior is defined as *error behavior*, as depicted in Figure 1.

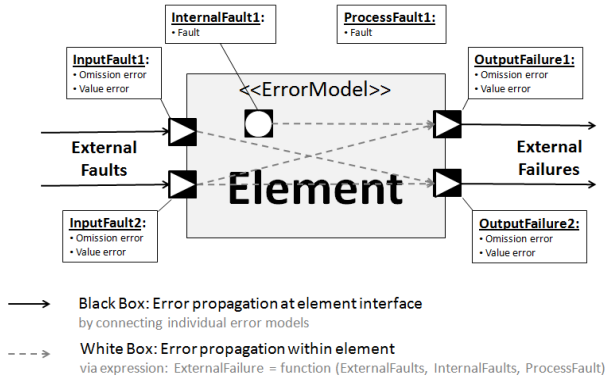



**Figure 1: Error model overview**

Moreover, the integrated view of the error models allows connecting external failures and external faults using the concept of a cause-effect relation named *fault failure propagation link*. This way it is possible to describe how external faults can be caused from preceding architectural elements and describe a complete error propagation chain from the root fault(s) towards the failure of interest. In addition a vertical propagation *error model mapping* is introduced to handle assumption made during safety analysis across different abstraction levels.

In the scope of the error behavior, the SAFE meta-model allows to either use a selected language (AADL, HiP-HOPS, AltaRica or free format) or the simplified SAFE language for the same purpose. The objective was not to reinvent a complete language, but to build the requirements for the grammar and the semantics of a simplified SAFE language that could be transformed transparently in HiP-HOPS or AltaRica for the user.

## 4. Safety Concept Continuous Evaluation

The safety analyses for each safety goal are used to support the safety concept and the development design phase activities. Requirements are derived from safety goals and refined up to HW/SW requirements as reflected in the ISO 26262 Part organization.

The top level *malfunctions* can be formally captured for the identified model element of the item decomposition as recommended in Clause 7.4.2 of Part 3 "Concept Phase" during the hazard and risk analysis phase.

The allocation of functional safety requirements onto the functional architecture represented in the analysis level of EAST-ADL allows defining the scope of the initial safety analyses. The analyses are built on local element malfunctions, e.g. similar to FMEA, used to perform systematic analysis of causes and effects of the malfunctions of the system elements. The malfunctions and error behavior are captured in the error model and propagate through the complete system model, finally linked to the top level malfunctions. The transformation of the complete system error model to a failure synthesis tool environment e.g. AltaRica [9] [10] or HiP-HOPS [16], allows qualitative evaluation of the violation of the safety goal. The tool output shall at least exhibit failure cut sets and sequence failures, extracted from a generated fault tree. To satisfy safety requirements, safety measure requirements are introduced to mitigate or control effects of the malfunction of the system elements. They can then be traced to the structural elements, and may induce modification on architecture elements to mitigate the respective failure. According to the update of the error model, the safety analyses can be re-executed to verify the functional safety concept, as granted by the Clause 8.4.5 of the Part 3 "Concept Phase". The

overview of the safety analysis process is summarized in the Figure 2.

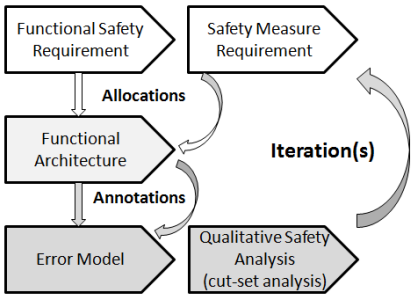



**Figure 2: Safety analysis process overview for functional safety concept**

Afterwards, the first version of the technical safety architecture and associated error model can be defined. It is built from requirement refinement (respectively functional safety requirements (FSR) into technical safety requirements (TSR) and from safety measure requirements into concrete software and hardware safety requirements) as well as from the allocation of technical requirements onto hardware and software architecture elements. Similar to previous methodology, the qualitative evaluation is performed compliant to Clause 7 of the ISO 26262, Part 4 "System Level" and common cause analysis can be preliminary assessed on component level.

Before starting the hardware design, the quantitative evaluation of hardware architectural metrics and residual risk targets for each safety goal conforming to Clause 8 and 9 of the ISO 26262, Part 5 for the hardware level can be performed to help refining the hardware safety requirements .More detailed information about additional modeling constructs and methodology is detailed in section 5.

Finally, corresponding hardware and software disciplines can start implementation of hardware and software products. The error model and safety analyses can still be applied on the modeling elements using aggregation of error model to AUTOSAR modeling constructs. Failure mitigation and assumption can still be traced and verified according to the architecture thanks to the use of the error model mapping construct.

## 5. Hardware architecture evaluation

As introduced above, the hardware architecture captures the technical capability of the intended hardware to the achievement of functional safety. It is based on decision of allocation of technical safety requirements and safety measures requirements to hardware elements. To do so the, artifacts of the EAST-ADL hardware design architecture are used. The hardware architecture is represented by a net of logical hardware components connected by their electrical connections. These components can be simple sensors or actuators, or complex components composed by parts representing the functionalities they are embedding. The component net is the hardware view of the technical safety concept iterated during safety analysis. It shall be emphased that this representation does not describe the electronic schematic including all the parts of the hardware design (e.g. resistance, capacitor, etc.), but it depicts blocks for hardware function scope definition.

Then, failure propagation analysis is performed. Hardware complex component (e.g. microcontroller) has a strong interaction between hardware and software. Accurate system failure propagation can only be made by a mixed analysis of software and hardware architecture. Therefore analysis of overall architecture change, in particular for introduction of safety mechanisms, shall always consider both failure perspectives and theirs interactions.

The relation between hardware and software architecture is defined according to the Hardware-Software Interface (HSI) as required by Clause 7.4.6 of ISO 26262 Part 4 "System Design". This element is an abstraction of a hardware detailed element, e.g. memory mapping, operating mode, etc... It has a dual nature to allow referencing elements of the hardware domain to elements of the software domain. It describes the fault propagation between the two domains.

As presented in the previous section, the ISO 26262 claims quantitative evaluation of the hardware metrics according to the ASIL level. These evaluations require dedicated hardware constructs to be able to capture reliability data of the hardware element: *component failure mode* as a specialization of a malfunction and *component failure rate.* As a hardware component is an abstraction of a section of a detailed electronic schematic (i.e. a super set of electronics parts), reliability values are defined as component allocation values or reused from pre-existing design. The calculation of the hardware architecture metrics, Clause 8 of ISO 26262 Part 5 "Hardware development", needs to identify the type of the fault. For the purpose of Single-Point Fault Metric (SPFM) and Latent-Fault Metric (LFM) calculation, the fault shall be tagged as single point fault, residual fault or latent fault. This tag "component fault" of an element depends on the context. It is defined from combination of the result of the qualitative analysis as the cut-set order, a malfunction as a leaf of a fault tree, and from the traceability information identifying its coverage by a safety mechanism. This is identified by a *satisfy* relationship from hardware component (or its sub-part) playing the role of a safety mechanism, to a *safety mechanism requirement*.

The diagnostic coverage (DC) for latent and residual faults is a property of a "safety mechanism". Their values are used for final SPFM and LFM calculation.

They are defined as targets for the architecture for implementation of a *safety mechanism* solution element. The target value shall be compatible with the impact in hardware metrics for each safety goal, and be bounded by the worst case value matching the maximum ASIL level. They have to be verified from the detailed design architecture using electronics parts reliability data. The DC target value and fault qualification can then be used to derive hardware safety requirements assigned on selected parts of the electronic design. We propose a quantification of the failure rate of this hardware component by a formula built from the reliability data of the super set of electronic parts. The result of the calculation can then be re-introduced as the value for the failure rate of the respective hardware component, to verify the estimated value from the preliminary SPFM and LFM calculation.

The concept of component failure rate allocation is also applied for the evaluation of residual risks of safety goal violations from Clause 9 of ISO 26262 Part 5. Two alternatives are proposed: Failure Rate Class (FRC) method and “Probabilistic Metric for Random Hardware Failure” (PMHF). FRC method evaluates each hardware component individually, while PMHF presents a global approach using as an example a quantified FTA. As for architectural metrics, the relation to electronic parts super set and associated formula can be used.

## 6. AUTOSAR software evaluation

In the AUTOSAR architecture, the software is composed of many interconnected AUTOSAR software components, which are deployed on the microcontrollers within the ECUs. In addition, the microcontrollers also contain an AUTOSAR basic software stack which controls the Microcontroller Unit (MCU) hardware and provides generic services to the software components, like access to input/output channels, persistent memory or partitioning. The use of the error model concept allows creating a clear separation between the application layer and the application environment given from the AUTOSAR infrastructure (ECU-hardware, basic software and RTE). This split permits to analyze the software considering the impact of the different malfunctions from the application environment. The set of malfunctions originated from the environment are clustered into computing and communication anomalies. Figure 3 below depicts the separation of the application layer and the application environment for safety analysis purposes.

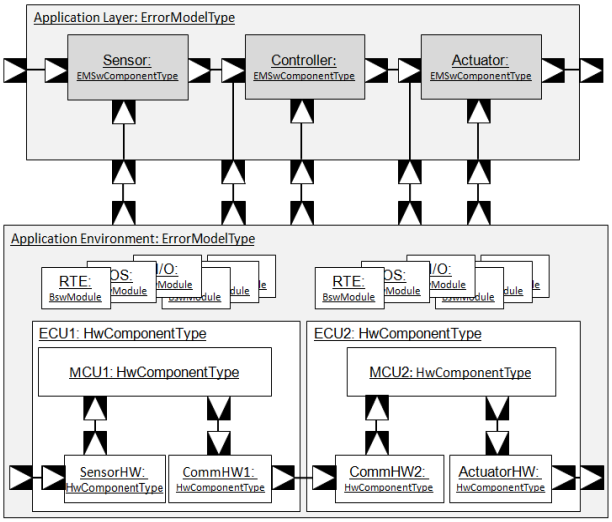



**Figure 3: AUTOSAR error model separation**

The failure ports of the application layer match exactly those of the application environment. Moreover, the abstraction of the infrastructure can be improved to reach the level of ECU partition as defined by the AUTOSAR specification. The ECU partition, (e.g. each ECU in Figure 3 could be seen as independent ECU partition) guarantees freedom from interference of software applications and allows to deploy application software with different ASIL allocation to the same ECU.

The application environment describes the malfunctions induced by the execution platform incl. network technology. Horizontal propagation links express how errors propagate from one software component to another on the same level. Having this information as input, a complete AUTOSAR software safety qualitative evaluation can be performed using the same tool environment as proposed in section 4.

## 7. Case study and tool prototype support

One of the selected public case studies to exhibit methods and tools developed in the SAFE project is extracted from the ISO 26262 Part 5 Annex E. This initial example is re-engineered to demonstrate safety analysis during the engineering development down to the level of implementation level. This section explains the use of the case study to reach SAFE project objectives. In addition, relation to relevant parts of the SAFE tool environment is given.

The SAFE tool environment is built upon the open source SAFE reference platform connected with specialized safety analysis tools like PREEvision [24] from Vector or SafetyDesigner from Dassault Systèmes [10]. The SAFE reference platform provides an EMF-based Java implementation of the SAFE meta-model and integrates the AUTOSAR meta-model implementation from ARTOP [22] as well as the EAST-ADL meta-model implementation from EATOP [23]. The SAFE meta-model platform offers a basic authoring experience, i.e., an Eclipse

perspective with a tree-based model explorer view for navigating through SAFE model files. It allows editing safety-related extensions for EAST-ADL and AUTOSAR models. This content is defined as the minimum content and shall facilitate platform extensions, in particular for development of graphical user interface and specialized safety analysis tool development.

**Safety Goals**

The case study is an automotive system that realizes two functions inside the same ECU. Function one is controlling a sensor temperature in order to open an external valve in case of overheating. The potential hazard is a vehicle catching fire. First hazard and risk analysis leads to the following safety goal classified with ASIL B criticality: "valve 2 shall not be closed for longer than x ms when the temperature is higher than 100 ℃". Function two, with the constraints to be integrated into the ECU with only one processor is controlling the vehicle speed to open another external valve in case of upper limit speed, similar to cruise control functionalities. The second hazard can be identified as vehicle accident classifying the safety goal "valve 1 shall not be closed for longer than y ms when the speed is higher than 100 km/h" with ASIL C criticality.

**Functional Safety Concept**

The initial representation of the functional architecture of the system uses the analysis level of EAST-ADL. It is used to represent the Functional Safety Concept thanks to EAST-ADL *analysis blocks* complemented by the SAFE error model. For function one (seeFigure 4), the analysis level is represented by a *FunctionalDevice* for the temperature sensor, a second one for the output valve and an *AnalysisFunction* for the calculation of the temperature controlling the valve 2 activation.

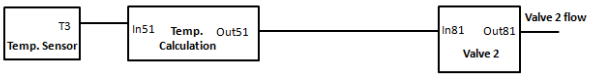



**Figure 4: Function one representation**

The SAFE error model references this top level architecture and is composed by three interconnected *ErrorModelPrototypes* (see Figure 5), one for each function block. Each *ErrorModelPrototypes* is typed by a respective *ErrorModelType*, which defines its *MFPFunctionPorts* in the role as external faults or failures. They are typed by a general *MalfunctionType* called "PError" representing the general error as failure propagated through the port. Each *MFPFunctionPort* is referencing the respective *FunctionFlowPort* of the selected EAST-ADL block to link the system model and the error model. The *ErrorBehavior* of each *ErrorModelType* is enriched by one *MalfunctionPrototype* in the role as internal fault and typed by one of the general *MalfunctionTypes Sensor_Error*, *Actuator_Error* or *Function_Error*.

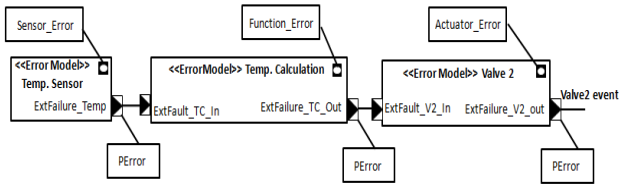



**Figure 5: Corresponding Error Model**

In addition, the internal failure propagation shall be defined according to the SAFE expression language. For Temp_Calculation, the failure propagation is expressed as follows: *"ExtFailure_TC_Out.ERROR = ExtFault_TC_In.ERROR OR Internal_TC-F_Error".* The implementation of the safety goal for function one corresponds to the deviation of one of the external failures of the top-level error model. This is shown as "valve2 event" in Figure 5. The deviation is represented by the occurrence of the malfunction *PError* on the *MFPFunctionPort* output. This first error model representation allows performing qualitative safety analysis on the functional safety concept as recommended in Figure 2 of section 4.

The connection of fault synthesis tools to the SAFE reference platform is still under development. So, the error model was re-captured in the syntax of the fault synthesizer using the HiP-HOPS [16] analyzer. The functional architecture and SAFE error model element were mapped to the syntax of HiP-HOPS to evaluate function one. Not surprising, the results lead to an order one of the cut set caused by basic event *Sensor_Error* on the temperature sensor block or by a *Function_Error* on the temperature calculation and control block. A functional safety requirement is created, where the attribute *Tactic* field indicates the expected detection and mitigation strategy. For function one, it indicates that the external failures associated with the error models for Temperature_Sensor and Temperature_Calculation shall be mitigated. Thus, a new EAST-ADL analysis block *LifeChecker* is introduced, as shown in the complete architecture in Figure 6. The hardware watchdog of the example is seen as a possible implementation of the *LifeChecker*.

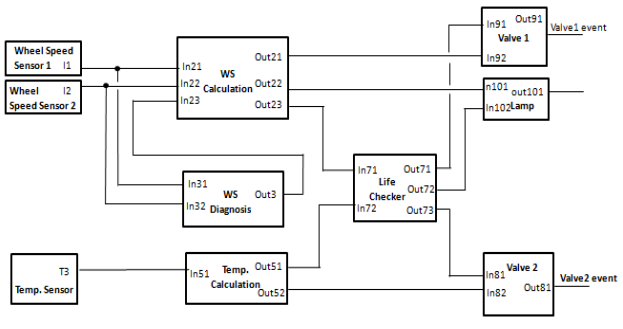



**Figure 6: Functional architecture example**

The corresponding error model is updated with the objective for LifeChecker error model to mitigate the failures of *Temperature_Sensor* and *Temperature_Calculation*. The *LifeChecker* detects the fault on its port *In72* and mitigates the failure propagation by controlling the deactivation of the valve 2, as safe state, on its port *Out73*. The *ErrorBehavior* of the *LifeChecker* error model is created with no dependence on the incoming input fault (logical expression defined as *"ExtFailure_LC-F_InhValve.ERROR = Internal_LC-F_Error"*. As we can see on the HiP-HOPS result shown in Figure 7, the cut set is now of order two and the mitigation of the *LifeChecker* is visible.

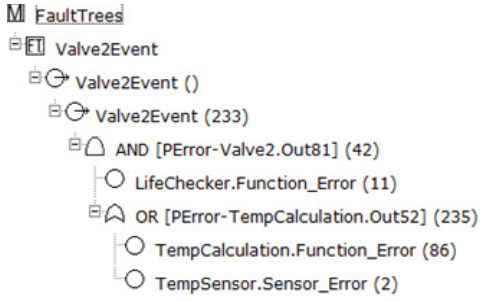



**Figure 7: Temperature Control fault tree analysis**

The same process was applied on the wheel speed control (as function two) leading to the introduction of the second sensor and the diagnosis block and for the integration with the *LifeChecker* block.

### Technical Safety Concept

Now the technical safety concept is defined, based on technical safety requirements and the allocation of safety requirements to hardware and software.

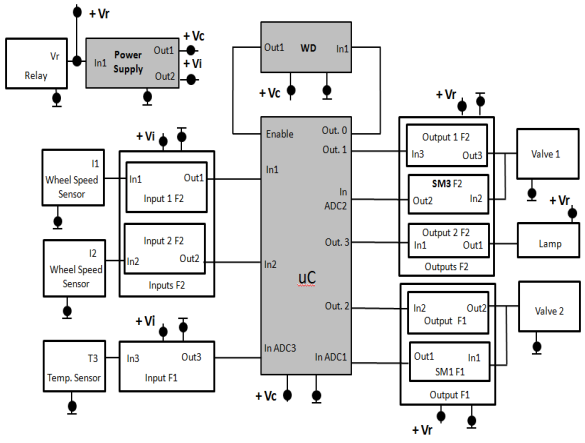



**Figure 8: Hardware architecture example**

The hardware architecture is based on EAST-ADL *HWComponentPrototypes representing* hardware functional blocks, and integrates the electronic driver abstraction (as represented in Figure 8). The software architecture is defined using EAST-ADL *DesignFunctionPrototypes*. Figure 9 depicts the software architecture for function with the relation to hardware elements, including an exemplary declaration of software driver abstractions later integrated in the AUTOSAR Software infrastructure (BSW blocks Temp ADC Driver and Current ADC Driver).

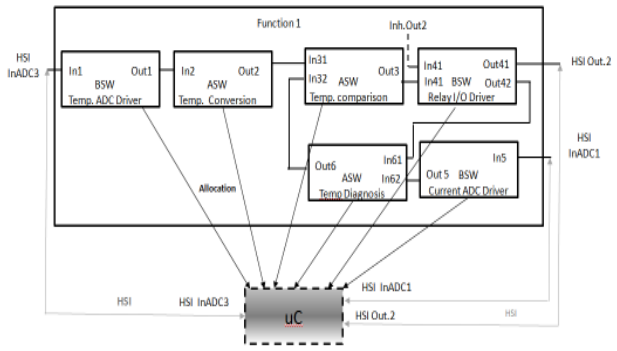



**Figure 9: Design level software architecture example**

The error propagation between hardware and software needs to consider first via their interface (HSI). For this purpose, SAFE declares the *HardwareSoftwareInterfaceElement*, allowing to reference port interfaces between the EAST-ADL elements *FlowPort* and *HWPin*. Secondly, the software function allocation to hardware processing units was declared using the *Allocation* element of EAST-ADL. Then, system error model is built and split into two perspectives, one for the hardware and one for the software. For interfacing the error model, each perspective declares its external faults and/or external failures to describe the incoming faults and propagating failures of the individual perspective. They are modeled as *MFPFunctionPort* for software and *MPFHWPin* for hardware and linked by a *FaultFailurePropagationLink*. This link is referencing the port malfunctions and the source element of this propagation, respectively an *HSI* element or an *Allocation* net. HiP-HOPS support the perspective concept and dedicated syntax for fault propagation between perspectives and the previously created SAFE artifacts like *MFPFunctionPort* or *ErrorModel* are mapped to the HiP-HOPS syntax. This modeling style allows synthesizing a fault tree including elements from software and hardware perspective. As for functional safety concept analysis, the introduction of malfunction mitigation through technical safety requirements and concrete realizations allows defining the complete technical safety concept.

The SAFE reference platform was used to capture the architecture and error model. The HiP-HOPS file has been generated manually. As we can see in the Figure 10 for temperature control (function one), the introduction of the output diagnosis (port *OutFbV2*) mitigates a bad software control (port HIS_*Drv2*_out_v2). It can also be noticed that failure of the output power stage hardware driver directly affects the control of the valve 2.

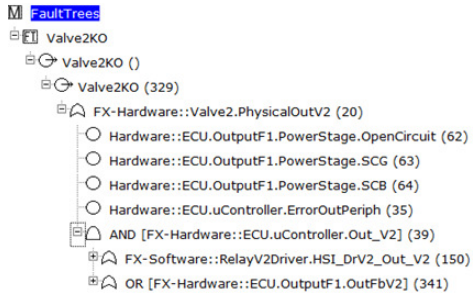



**Figure 10: Temperature Control fault tree analysis**

## Software Safety Analysis

At implementation level, we further refine the system design and define the AUTOSAR software architecture of the mentioned functions. For function one (controlling the temperature), the software architecture consists of the software components shown in Figure 11. *TempDiagnosis* checks if the control of valve 2 has been operated correctly by comparing the expected actuator state and the input sensing the current actuator state. *TempControl*[1] implements the rule of opening the valve in case of overheating. *SensorSWC*[2] and *ActuatorSWC* abstract the access to the peripherals. *CDD*[3] (Complex Device Driver) and *ECUAbstraction*[4] abstract from the low level details of ECUs hardware (i.e. concrete analogue sensors and output stage).

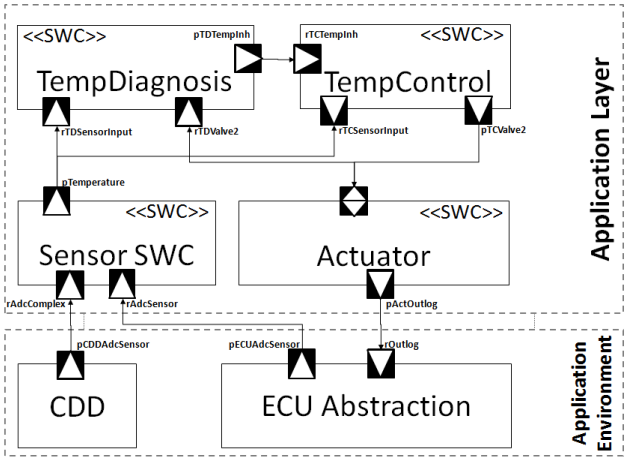



**Figure 11: Function one software architecture**

We model the erroneous behavior of the architectural elements using the SAFE meta-model.

Initially, we describe a library of *MalfunctionTypes*, which are of interest for the system under consideration. In our case, we propose the types *ComputationAnomaly* and *CommunicationAnomaly*. Each type can be hierarchically decomposed. In case of *CommunicationAnomaly*, we further refine this type into the following primitive *MalfunctionTypes*: *value_error* (invalid content of the associated data), *timing_error* (data arrives too late/too early), *omission_error* (data does not arrive) and *commission_error* (data arrives too often).

As a next step, we describe the error model of the considered software. As mentioned in section 6, we distinguish between application layer and application environment. The error model for the application environment includes the malfunctions induced by the complete basic software stack and the hardware components of the ECU. The error model for the application layer includes a hierarchical description of the error propagation within the AUTOSAR software components under consideration. Figure 12 highlights the *ErrorModelType* of the software component *TempControl*. The SAFE meta-model allows to link the model elements of the *ErrorModelType* with the architectural elements. For example, the external failure *ExtFailure_TC_Valve2* (see Figure 12) is associated with the respective data element of port *TempControl.pTCValve2* (seeFigure 11).

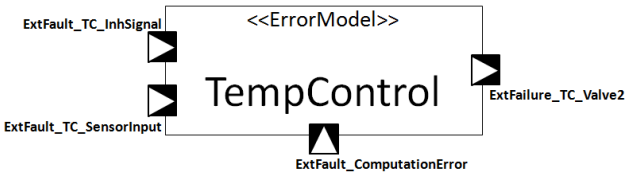



**Figure 12: ErrorModelType for TempControl**

The semantics of the external faults/failures is given through the respective *MalfunctionType* associated with them. For example, *ExtFault_TC_SensorInput* has the type *CommunicationAnomaly* as described above. Moreover, the external faults/failure can be brought into relation, describing the error propagation within the architectural element. Considering *TempControl*, the external Failure *ExtFailure_TC_Valve2* occurs, if a) the sensor input is invalid, or b) the inhibition signal provided by the component *TempDiagnosis* is invalid or c) the application environment exhibits a computation error leading to the erroneous execution of the software component. This can be described via the following expression:

*ExtFailure_TC_Valve2.value_error = ExtFault_TC_InhSignal.value_error OR ExtFault_TC_SensorInput.value_error OR ExtFault_ComputationError.**

For a complete error model of function one, the unintended behavior of all involved architectural elements must be specified and interlinked. In particular, the *ErrorModelType* for the application environment describes the malfunctions of the concrete ECU where function one is deployed to. It includes, amongst others, one external failure describing improper computation. This malfunction is then linked with the external fault *ExtFault_ComputationError* of the error model for *TempControl*.

This way, a complete failure net can be established, which can then be used as input for safety analysis tools to automatically derive safety-relevant

[1] Realizes ASW "Temp Comparison" of Figure 9
[2] Includes ASW "Temp Conversion" of Figure 9
[3] Realizes BSW "Current ADC Driver" of Figure 9
[4] Realizes BSWs "Current ADC Driver" and "Temp ADC Driver" of Figure 9

conclusions like Single Point Faults leading to the violation of a safety goal. In the current state, no direct link between the SAFE tool platform and known safety analysis tools like Hip-Hops [16] or SafetyDesigner [10] is available. However, experimental attempts have shown that SAFE error models can be transformed into the respective representation of the mentioned tools.

**Hardware Safety Analysis**

Based on the technical safety concept in terms of hardware components as shown in Figure 8, the impact of random hardware failures has to be assessed according to the quantitative evaluations claimed by ISO 26262 Part 5.

The determination of the classification is described exemplarily based on the fault tree shown in Figure 10. Failure mode *OutputF1.PowerStage.OpenCircuit* is directly affiliated via one OR-gate to the top-level event *Valve2.PhysicalOutV2*. This represents a minimal cut-set of order 1. Therefore, the failure mode is classified as a single-point fault. Failure modes which are contained in the branch of *ECU.uController.Out_V2* are affiliated with the top-level event via at least one AND-gate. Thus, they are represented in a minimal cut-set of order greater than or equals 2 and classified as latent multiple-point faults.

To complement the initial safety evaluation of hardware architectural designs, the hardware component failure rates values represent assumptions taken from previous designs or expert knowledge. In later phases of development, the hardware architectural design is refined at the level of electronic schematics, exemplarily shown in Figure 13.

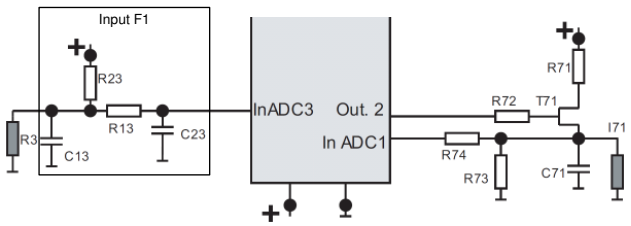



**Figure 13: Hardware detailed design for function one of [1] Part 5 Annex E**

The failure rate assumptions can now be confirmed based on an electronic FMEA using logical expressions and the hardware part failure data taken from exemplarily recognized industry sources or company-specific databases [1]. This is exemplarily presented for the failure modes of hardware component *Input F1*.

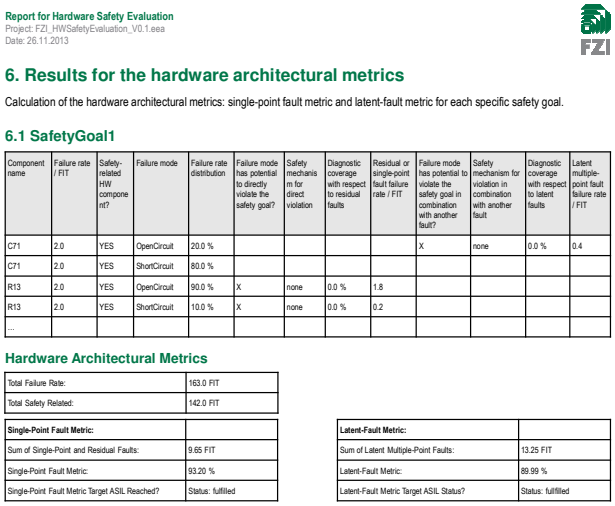
*InputF1.SCG = C13.SC OR C23.SC*

*InputF1.SCB = R23.SC*

*InputF1.OC = R23.OC OR R13.OC*

To quantify the expressions, the failure rates of the corresponding failure modes have to be transformed into probabilistic values based on system lifetime, according to ISO 26262 Part 10 Annex B.4. Alternatively to the hardware architectural design, safety evaluation could also be performed directly at the level of electronic schematics [20].

For the quantitative evaluations according to [1], FZI has set up a research prototype implementation which provides classification of failure modes based on fault tree generation and analysis. Hardware structural and failure data for architectural and detailed designs can be imported using a XML-based file format. Results of the quantitative evaluations are automatically verified against target values, which can be user-defined or taken from recommendation of [1].


Report for Hardware Safety Evaluation
Project: FZI_HWSafetyEvaluation_V0.1.eea
Date: 26.11.2013

FZI

6. Results for the hardware architectural metrics

Calculation of the hardware architectural metrics: single-point fault metric and latent-fault metric for each specific safety goal.

6.1 SafetyGoal1

| Component name | Failure rate / FIT | Safety-related HW component? | Failure mode | Failure rate distribution | Failure mode has potential to directly violate the safety goal? | Safety mechanism for direct violation | Diagnostic coverage with respect to residual faults | Residual or single-point fault failure rate / FIT | Failure mode has potential to violate the safety goal in combination with another fault? | Safety mechanism for violation in combination with another fault | Diagnostic coverage with respect to latent faults | Latent multiple-point fault failure rate / FIT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C71 | 2.0 | YES | OpenCircuit | 20.0 % | | | | | X | none | 0.0 % | 0.4 |
| C71 | 2.0 | YES | ShortCircuit | 80.0 % | | | | | | | | |
| R13 | 2.0 | YES | OpenCircuit | 90.0 % | X | none | 0.0 % | 1.8 | | | | |
| R13 | 2.0 | YES | ShortCircuit | 10.0 % | X | none | 0.0 % | 0.2 | | | | |
| ... | | | | | | | | | | | | |

Hardware Architectural Metrics

| | |
|---|---|
| Total Failure Rate: | 163.0 FIT |
| Total Safety Related: | 142.0 FIT |

| Single-Point Fault Metric: | |
|---|---|
| Sum of Single-Point and Residual Faults: | 9.65 FIT |
| Single-Point Fault Metric: | 93.20 % |
| Single-Point Fault Metric Target ASIL Reached? | Status: fulfilled |

| Latent-Fault Metric: | |
|---|---|
| Sum of Latent Multiple-Point Faults: | 13.25 FIT |
| Latent-Fault Metric: | 89.99 % |
| Latent-Fault Metric Target ASIL Status? | Status: fulfilled |

**Figure 14: Excerpt of report for hardware safety evaluation**

All analysis results are included in corresponding reports as a PDF file for documentation purposes, as exemplarily shown in Figure 14 for the hardware detailed design of ISO 26262 Part 5 Annex E. The presented hardware safety evaluations and modeling of hardware designs including failure data are also integrated as prototype plug-ins in the model-based tool environment PREEvision to ease consistent modeling and evaluation [25][26].

## 8. Conclusion and future work

In this paper we presented a methodology for model based safety analysis of automotive systems. Based on a case study, we demonstrated its capability to perform early assessment of safety concepts. The different evaluations comply with the ISO 26262 “Road Vehicle - Functional Safety” and facilitate the design of automotive products via qualitative evaluation of the architecture, from functional abstraction down to implementation layer (e.g. AUTOSAR software architecture). In addition, it provides a proposal to control the complexity of electronic design to compute the quantitative measures required by the ISO 26262 standard.

The automotive domain, quite conservative for methods but subject to competitive market, requires a fully integrated environment to ease modeling and to abstract the complexity of safety analysis constructs. Despite actual maturity of specialized analyzer, this is still a challenge. In particular, today's tools only partly support the automotive design engineer adequately in its activity to conduct multi domain safety analysis with short iteration loops.

The assessment of AUTOSAR infrastructure related to hardware domain, as middleware architecture, hardware element and HSI is still under investigation. This is part of future work.

## 9. Acknowledgement

This work was supported by contribution of all the partners of the SAFE project consortium. This SAFE project in running the framework of the ITEA2, EUREKA cluster program Σ! 3674. The work has been funded by the German Ministry for Education and Research (BMBF) under the funding ID 01IS11019, and by the French Ministry of the Economy and Finance (DGCIS).

## 10. References

[1] International Organization for Standardization, "Road vehicles - Functional safety," ISO Standard 26262, Rev. Nov. 2011.

[2] International Electro-technical Commission, "Functional safety of electrical /electronic/ programmable electronic safety-related systems," IEC Standard 61508, Rev. Apr. 2010.

[3] AUTOSAR Development Partnership, http://www.autosar.org

[4] EAST-ADL Association supporting EAST-ADL2 Language, http://east-adl.info/

[5] ARP4754A, Guidelines for Development of Civil Aircraft and Systems, Aerospace Recommended Practice, SAE International Standard http://www.sae.org/standards/

[6] ARP4761, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment is a standard, aerospace recommended Practice, SAE International Standard http://www.sae.org/standards/

[7] M. Bozzano and all, "ESACS: an integrated methodology for design and safety analysis of complex systems", in proceedings of ESREL 2003, Balkema publisher.

[8] A. Arnold, A. Griffault, G. Point, A. Rauzy, "The AltaRica formalism for describing concurrent system", Fundamenta Informaticae, Vol. 40, n°2-3n pp 109-124, IOS Press, 2000.

[9] AltaRica project https://altarica.labri.fr/forge/

[10] Safety Designer V9, Dassault Systèmes, http://www.3ds.com/

[11] P. Bieber, and all, "Safety Assessment with AltaRica - Lessons learnt based on two aircraft system studies" 18th IFIP World Computer Congress, Topical Day on New Methods for Avionics Certification, August 26th, 2004, Toulouse (France).

[12] R. Adeline, P. Darfeuil, S. Humbert, J. Cardoso, C. Seguin, "Toward a methodology for the AltaRica modelling of multi-physical", In Proceedings of European Safety and Reliability Conference, ESREL 2010. Rhodes (Greece). September, 2010.

[13] ASSERT Project final publication, http://www.assert-project.net/Publications

[14] AADL, Architecture Analysis & Design Language http://www.aadl.info/, SAE International Standard.

[15] Ana-Elena Rugina, Karama Kanoun, Mohamed Kaâanich, "A System Dependability Modeling Framework Using AADL and GSPNs", Architecting Dependable Systems IV, Lecture Notes in Computer Science Volume 4615, 2007, pp 14-38

[16] HiP-HOPS from University of Hull, http://www2.hull.ac.uk/

[17] Y. Papadopoulos, Y., Walker, M., Parker, D., Rüde, E. et al., "Engineering failure analysis and design optimization with HiP-HOPS," Engineering Failure Analysis 18(2):590–608, 2011.

[18] Jeon, S.-H., Cho, J.-H., Jung, Y., Park, S. and Han, T.-M., "Automotive hardware development according to iso 26262," 13th International Conference on Advanced Communication Technology (ICACT), 2011.

[19] K. Svancara and all, "Experience with the second method for eps hardware analysis: "evaluation of each cause of safety goal violation due to random hardware failures", " VDA Automotive SYS Conference on Quality and Functional Safety Management for Automotive software-based Systems, 2012.

[20] Adler, N., Otten, S., Cuenot, P., and Müller-Glaser, K. D. , "Performing Safety Evaluation on Detailed Hardware Level according to ISO 26262", SAE Int. J. Passeng. Cars – Electron. Electr. Syst. / Volume 6, Issue 1 (May 2013)

[21] SAFE project, www.safe-project.eu

[22] ARTOP, AUTOSAR Tool Platform User Group, http://www.artop.org/

[23] EATOP, EAST-ADL Tool Platform from Eclipse Automotive Industry Workgroup Lab, https://code.google.com/a/eclipselabs.org/p/eclipse-auto-iwg/

[24] PREEvision, Vector Informatik GmbH, http://vector.com/vi_preevision_en.html

[25] N. Adler, M. Hillenbrand, K. Müller-Glaser, E. Metzker, and C. Reichmann, "Graphically notated fault modeling and safety analysis in the context of electric and electronic architecture development and functional safety," in Rapid System Prototyping (RSP), 2012 23rd IEEE International Symposium on, 2012, pp. 36–42.

[26] N. Adler, S. Otten, M. Mohrhard, and K. Müller-Glaser, "Rapid Safety Evaluation of Hardware Architectural Designs Compliant with ISO 26262," in Rapid System Prototyping (RSP), 2013 24rd IEEE International Symposium on, 2013, pp. 66–72.

## 11. Glossary

| | |
|---|---|
| SCB | Short Circuit to Battery |
| SCG | Short Circuit to Ground |
| DC | Diagnostic Coverage |
| HSI | Hardware Software Interface |
| IMA | Integrated Model Avionic |
| ECU | Electronic Control Unit |
| E/E | Electrical/Electronic |
| FRC | Failure Rate Class |
| FMEA | Failure Mode and Effect Analysis |
| FTA | Fault Tree Analysis |
| LFM | Latent Fault Metric |
| MBD: | Model Based Development |
| MPFL | Multiple Point Fault Latent |
| OC | Open Circuit |
| PMHF | Probabilistic Metric for Random Hardware Failure |
| RTE | Run Time Environment |
| SC | Short Circuit |
| SPF | Single-Point Fault |
| SPFM | Signe-Point Fault Metric |