Contract number: ITEA2 – 10039

Safe Automotive soFtware architEcture (SAFE)

ITEA Roadmap application domains:

Major: Services, Systems & Software Creation

Minor: Society

ITEA Roadmap technology categories:

Major: Systems Engineering & Software Engineering

Minor 1: Engineering Process Support

WP3 - Deliverable D3.2.2b Proposal for extension of meta model for hardware modeling

Due date of deliverable: 27/12/2013

Actual submission date: 20/12/2013

Start date of the task: 28/11/2011 **Duration:** 25 months

Project coordinator name: Stefan Voget

Organization name of lead contractor for this deliverable: Continental France

Editor: Philippe Cuenot (Continental France)

Contributors: Philippe Cuenot (Continental France), Nico Adler (FZI), Stefan Otten (FZI)

Reviewers: Florent Meurville (Valeo), Martin Hillenbrand (FZI), Nico Adler (FZI), Stefan Otten (FZI)

Revision chart and history log

Version	Date	Reason
0.1	2012-07-17	Initialization of document
0.2	2012-12-14	Revision of template and allocation
0.3	2013-01-23	Initial documentation of contribution to SAFE meta model and inter-dependencies to other work tasks
0.4	2013-02-05	Conti-F contributions chap 4 to 6.
0.5	2013-02-06	Meta model documentation, Hardware modeling scoping, Performing Hardware Modeling based on EAST-ADL
0.6	2013-02-08	Review of Chapter 6 by Florent for WT331 interface
0.7	2013-02-19	Current status EAST-ADL + chap. 7 and 9
0.8	2013-02-21	Update MM description
0.9	2013-02-22	New chapter 8.3, 8.4, 9.2 and completion of chap 10 and 11
1.0	2013-02-28	Final version 1.0 with proof reading (published 28.02.2013)
1.1	2013-12-02	Up to date meta model and clarification of section 9 and section 10
1.2	2013-12-16	Review
2.0	2013-12-20	Final version D3.2.2.b (published 20.12.2013)

1 Table of contents

2 List of figures

3 Executive Summary

The work task WT3.2.2 targets the topics of hardware modeling and evaluation according to ISO 26262. This activity includes the definition of the necessary elements to represent the hardware architecture of the technical safety concept and the hardware parts of electronic schematics. It also comprises the constructs supporting the calculation of hardware quantitative measures demanded by ISO 26262 [1] in terms of the hardware architectural metrics and the evaluation of safety goal violations due to random hardware failures.

Besides giving an overview of relevant sections in ISO 26262, the allocated requirements to WT3.2.2 resulting from an ISO 26262 analysis of WT 2.1 and the needs from use case descriptions in WT2.3 are presented.

In addition to the previous mentioned overview, the methodology for the hardware technical safety concept representation, the hardware component failure modes and classification rating definition in accordance with the needs of ISO 26262 is presented. As it is objective to develop a meta-model for hardware modeling, the current version of EAST-ADL[3] and AUTOSAR[2] is analyzed. Moreover, the contribution of WT3.2.2 to the SAFE meta-model, which is based on EAST-ADL is presented.

The relation of selected hardware meta model constructs with the consumer electronic interchange format IP-XACT [4] from Accelera Organization is discussed. A first overview of proposed links is given.

4 Introduction and overview of document

This document provides information about a methodology for hardware modeling to facilitate the representation and to perform the safety evaluation of the technical safety concept related to hardware components and hardware parts as electronic components. The proposed method will rely on existing automotive standards AUTOSAR and EAST-ADL, and will forecast to elaborate a first approach to connect it to the consumer electronics standard IP-XACT.

4.1 Scope of WT 3.2.2

Work task WT3.2.2 deals with model-based structural and failure description of hardware.

Basis is the hardware design architecture of EAST-ADL [3] and the ECU resource template from AUTOSAR [2] in the hardware element description, both being presented in chapter 8. WT3.2.2 intends to provide a methodology for the hardware architecture component representation and decomposition into a hardware part with respect to safety evaluation related to random hardware failures. The existing current meta-model of EAST-ADL and AUTOSAR will be analyzed to provide proposals for modification of basic standards via change request and to define appropriate safety-related extensions in terms of the described topics.

Additionally, the IP-XACT [4] interchange format will be mapped to AUTOSAR hardware elements, as component part description, in order to derive requirements for a possible automatic transformation to favor hardware model exchange with silicon suppliers.

Therefore, the following artifacts and their interrelations shall be considered:

Hardware Component

The applicable concept of EAST-ADL2.1 for hardware components (type and prototype) allows representing a logical or technical hardware element. This actual construct allows compositional organization of hardware elements, either used to represent logical element or directly as a physical electronic component. The use of logical elements allows a functional abstraction of electronic component, then allocated into one (or several) physical electronic complex component (e.g. FPGA, ASIC) or decomposed into a set of physical electronic component (resistors, capacitors, etc…). The hardware component concept shall enable a direct relation to behavioral representation for functional or dysfunctional modeling and possible simulation. Furthermore, the interconnection of component communication via Pins, Ports and Connectors shall allow the definition of generic abstraction concept for whatever bus interconnection is capable for on low level electronic abstraction features (e.g. SPI, AMBA bus...). The use of hardware components and their interconnections shall also permit flexible and reusable description of hardware characteristics in particular for the ports. This would facilitate the allocation of a hardware component to physical elements based on predefine semi- formal semantic.

Hardware Part

The concept for hardware part shall allow depicting the physical implementation of a hardware component, decomposed by multiple electronic parts, to be able to support the description of an electronic design schematic using concrete electronic components (exemplarily resistors, capacitors and complex components). AUTOSAR R4.0 includes hardware element constructs required for software configuration in AUTOSAR ECU Resource Template. The proposed use of hardware part shall enable the use of AUTOSAR hardware elements and define a clear interrelation with hardware component.

Hardware Architecture

The concept for Hardware Architecture has to comply with the needs of the Technical Safety Concept description of hardware components with regards to software components for the software architecture. The hardware architectural level represents the set of hardware components for the intended features of the system and additionally has to support the introduction of links for safety mechanisms and safety measures to be applied on hardware components. The hardware architecture shall be the perspective to collect the overall random failure information and link them in order to facilitate the calculation for the hardware architectural metrics and evaluation of the failure rate for violation of the safety goal. The hardware architecture is aimed to be based on the hardware component net representation and shall be set on the top of EAST-ADL2.1.

Hardware Electronic Design

The Hardware Electronic Design represents the hardware detailed design as at the level of electrical schematics representing the interconnections between hardware parts composing the hardware components. The hardware electronic design is the perspective where the random failure information of the physical electronic part is available (including value for complex component such as microcontroller or ASIC). An unambiguous relation between hardware part failure information and hardware components failure data shall be defined to permit the quantitative assessment of the hardware architecture level. The hardware electronic design is aimed to be based on the hardware part net representation and shall be set on the top of AUTOSAR 4.0.

Hardware Software Interface

The concept for Hardware Software Interface (HIS), as specified in Part 4 for the product development at system level, shall be explicitly represented in the system architecture composed by hardware and software architecture. Therefore, EAST-ADL2.1 needs to be adjusted to support a clear separation of hardware and software with respective component behavior attached to the component. An explicit element interface between software function and hardware component needs to be defined. This concept shall support continuity of domain flow (e.g. software as sampled physical data and hardware as electrical data) for functional simulation and error propagation. In addition, it shall allow abstraction principle compared to detailed concrete implementation applied at the system level architecture.

Failure Rate and Failure Mode

Hardware failure information such as failure rate and failure mode shall be captured in an unambiguous formalism to enable the data exchanged within supplier chain and to facilitate quantitative assessment of the hardware architecture. Moreover, this concept shall support the allocation and interrelation between logical hardware component and physical hardware part for joined calculations of hardware random failure from different hardware abstraction level (hardware architecture and hardware electronic design).

Fault and contribution to Safety Goal/Malfunction

The contribution of hardware components to the violation of the safety goal shall be supported by tagging safety-related components. The item, identified during hazard and risk analysis, can be decomposed according to sub-system development scenario. The hardware sub-system can exhibit local malfunctions, and their contribution to the top level system malfunction linked to the violation of the safety goal. This shall be incorporated in the meta model. The basic faults related to the top level malfunction, should be classified by the type of fault (e.g. single point fault, latent fault, multiple or residual fault).

Hardware Architecture Metrics and Probabilistic value

Based on the hardware component faults and their relations to safety mechanism including associated coverage rates, the hardware architectural metrics (Single-point fault and Latent-fault metric) need to be allocated first and subsequently verified by calculation. The same proceeding should be applied on probabilistic measures for the evaluation of safety goal violation due to random hardware failure (using Probabilistic Metric for random Hardware Failure PMHF) or for the evaluation of each cause using Failure Rate Class (FRC) method. The meta model extension developed in this work task shall enable to store the respective results of the calculation steps. Additionally, this provides documentation of results together with their respective parameters or assumptions. It shall also be able to express relation over the assumption of the logical hardware component and physical hardware part, to offer basic repository for the complete failure analysis methodology defined in WT3.3.1. Model-based quantitative evaluation in terms of hardware design assessment at different abstraction levels is described in the public deliverable D3.3.3b.

4.2 Structure of document

The document is structured as follows:

Subsequent to the introduction an overview on the parts of ISO 26262, which are relevant for the hardware development with its relation and assessment to the system development, is given in section 5.

Within section 6, the interface with WT3.3.1 safety analysis methodology will be clarified and defined according to the analysis of the impact from the hardware abstraction view and representation (system, component, part) in 6.1, and to the definition of the element to be interfaced in 6.2.

The section 7 deals with the coverage of the hardware requirements from the initial ISO26262 analysis, with the description of the organization and the topics selected from this WT3.2.2 requirement analysis. Notice that initial and derived requirements are available in an external document traced from WT2.1 activities.

Section 8 deals with hardware modeling using EAST-ADL2 and AUTOSAR 4.0. On the one hand, the current version of EAST-ADL2.1 in particular for the hardware description is highlighted and described in 8.1. On the other hand in 8.2, some proposed extensions to this current version are explained which enhance the possibility to perform complete hardware components development and quantitative safety analysis. Moreover the ECU Resource Template of AUTOSAR R4.0 will be exhibited in 8.3 showing how to use it for hardware part modeling. In section 8.4 we will briefly discuss a proposal for change of existing constructs.

The contribution of WT 3.2.2 to the SAFE meta-model is described in section 9. As introduced in section 9.1 the organization of change request and extension is presented. Section 9.2 gives a detailed description of the proposed change request for the current EAST-ADL meta model regarding classes and links. Our extension for EAST-ADL is described in section 9.3. Moreover, an example for the application of the meta-model for hardware modeling is presented in section 10.

In section 11 the preliminary relation between the hardware part elements as proposed in AUTOSAR R4.0 ECU Resource template and the existing construct of IP-XACT is proposed.

Finally, in section 12 a conclusion and discussion is given.

5 Overview on ISO 26262

Within this section, an overview of the relevant parts of ISO 26262 with regard to hardware modeling and safety-related activities are given. The selection of the presented parts is based on the SAFE requirements elicited in WT 2.1 which are allocated to WT 3.2.2.

Addressing the development process of electric / electronic components for passenger cars, the ISO 26262 "Road vehicles – Functional safety" came into effect in November 2011. This standard introduces a safety lifecycle which "encompasses the principal safety activities during the concept phase, product development, production, operation, service and decommissioning" ([1], part 2, p.3). This can be seen as a guideline that demands a risk-based development approach with seamless traceability. In Figure 1 an overview on the different parts of ISO 26262 is given.

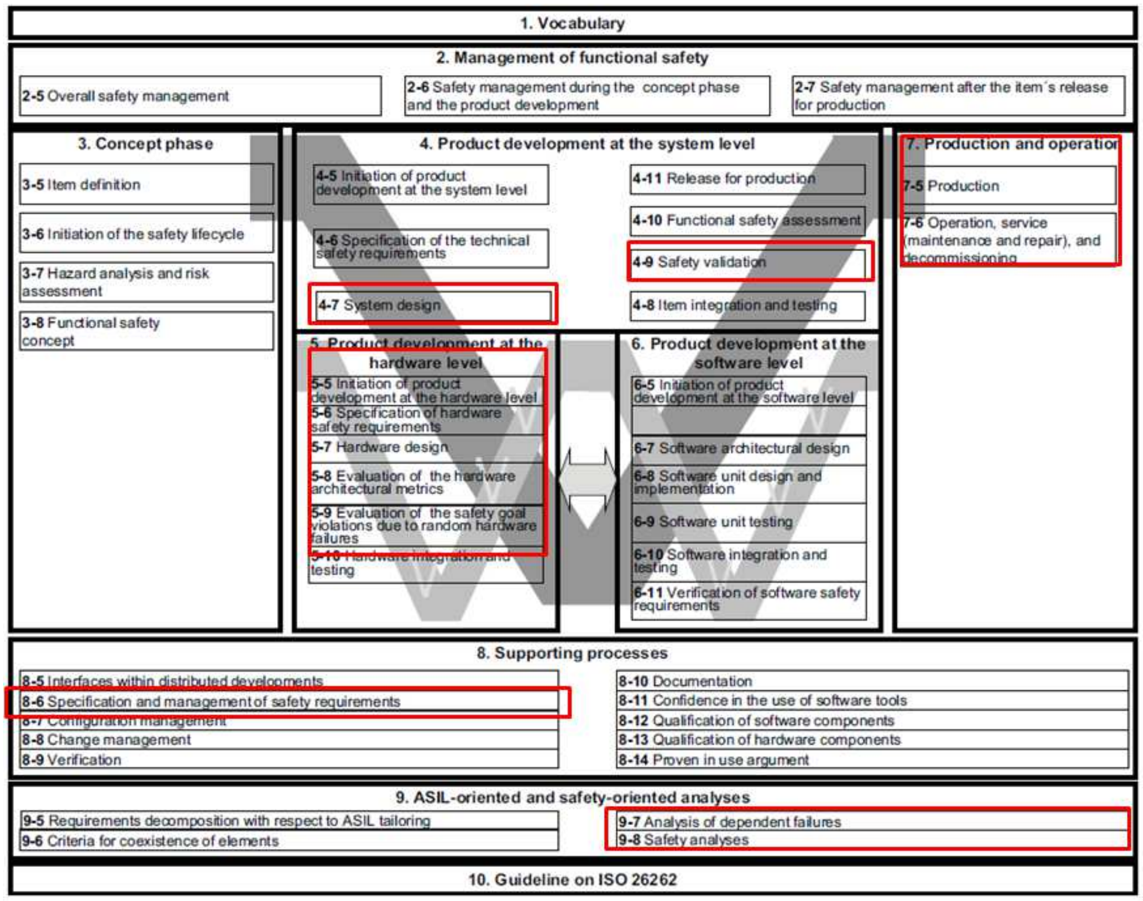

1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Safety management during the concept phase and the product development
2-7 Safety management after the item's release for production
3. Concept phase
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Functional safety concept
4. Product development at the system level
4-5 Initiation of product development at the system level
4-6 Specification of the technical safety requirements
4-7 System design
4-11 Release for production
4-10 Functional safety assessment
4-9 Safety validation
4-8 Item integration and testing
7. Production and operation
7-5 Production
7-6 Operation, service (maintenance and repair), and decommissioning
5. Product development at the hardware level
5-5 Initiation of product development at the hardware level
5-6 Specification of hardware safety requirements
5-7 Hardware design
5-8 Evaluation of the hardware architectural metrics
5-9 Evaluation of the safety goal violations due to random hardware failures
5-10 Hardware integration and testing
6. Product development at the software level
6-5 Initiation of product development at the software level
6-6 Specification of software safety requirements
6-7 Software architectural design
6-8 Software unit design and implementation
6-9 Software unit testing
6-10 Software integration and testing
6-11 Verification of software safety requirements
8. Supporting processes
8-5 Interfaces within distributed developments
8-6 Specification and management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8-10 Documentation
8-11 Confidence in the use of software tools
8-12 Qualification of software components
8-13 Qualification of hardware components
8-14 Proven in use argument
9. ASIL-oriented and safety-oriented analyses
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistence of elements
9-7 Analysis of dependent failures
9-8 Safety analyses
10. Guideline on ISO 26262

Figure 1: Overview on ISO 26262 (Relevant parts highlighted)

The relevant requirements for the hardware related development are mainly provided in ISO 26262:2011, Part 5 in "Product development at the Hardware Level". As a consequence of the exclusion from Part 5 chapter 10 "product integration and test" as a SAFE project decision, this chapter was considered as non relevant for this analysis. However, the Part 4 (Product development at System level) is strongly interlaced with respected to hardware development. Moreover, also in other parts, namely Part 7 (production and Operation), Part 8 (Supporting processes), and Part 9 (Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses) requirements are provided that affect directly or also indirectly the hardware development. In the following, an overview on the relevant aspects from the respective parts is given.

Part 4: Product Development – System Level

During this phase the development of the item from the system level perspective takes place. The process is based on the concept of a V-model. Starting point (on the upper left side) is the specification of the technical safety requirements which is followed by the development of the system architecture and the system design.

Safety mechanisms

During the system development the technical safety requirements specify the necessary safety measures to define mechanisms to detect and control the fault in the system, and their interactions with the system design in order to reach a safe state within a fault tolerant time interval. The safety mechanism shall be specified to prevent latent or multiple point faults with consideration of the given architecture and in particular for the one implemented by hardware components.

System Design – Technical Safety Concept

The system design shall implement the technical safety requirements by defining the technical capability of the intended hardware and software design with regard to the safety achievement. Measure to avoid systematic failure shall be introduced according to safety analysis in order to avoid system failure, via introducing of safety mechanism for component failure mitigation. According to analysis, specific measure to control random hardware failure during operation shall be specified. The target value for the hardware architecture shall be defined according to single-point fault and latent-multiple fault metric, and for quantification of avoidance of safety goal violation due to random hardware failures.

System Design – Allocation to Hardware and Software

As introduce above the system design shall include the hardware and software partitioning via allocation of technical requirements.

System Design – Hardware Software Interface Specification

The interaction between hardware and software component shall be defined to allow specification of component hardware devices controlled by software. Additionally, hardware resources, configuration and error mechanism shall be specified.

System Validation

The validation with hardware metrics shall be carried out at the item via evaluation of criteria for the evaluation of safety goal violation due to random hardware failures and for hardware architectural metrics as single-point fault and latent-multiple fault metrics (calculation of results versus targets).

Part 5: Product Development – Hardware Level

During this phase, the development of the item from the hardware perspective is performed. The process is again based on a V-model, going down with the specification of hardware safety requirements as well as hardware design and implementation.

Hardware Design

The hardware design shall be performed in accordance to system design and hardware safety requirements. It starts from the hardware architecture down to hardware detailed design at the level of electronics schematic describing interconnected hardware parts. The traceability of safety requirements shall be provided down to the lowest level of hardware components. The environmental conditions and potential causes of failures of hardware components shall be considered during design of hardware component.

Safety Analysis

The safety analysis of hardware design identifies the causes of failure and effect of faults regarding overall system failure behavior. The effectiveness of safety mechanisms shall demonstrate to avoid single-point fault, to maintain the system in safe sate and to validate coverage with respect to residual and latent faults. This WT3.2.2 will not propose methodology for fault propagation and failure identification as this is includes in WT3.3.1, but will provide necessary element to describe the fault and safety constraints to the respective hardware components and hardware parts.

Evaluation of Hardware Metrics

The hardware architectural metrics shall be computed to evaluate the effectiveness of the architecture to cope with random hardware failures. They have to be computed, for each violation of each safety goal on respective item of ASIL B to D, and to be applied iteratively from hardware architecture down to hardware detailed level. Similar to safety analysis, WT3.2.2 will only cope with the elements to capture component failure information, metrics targets and results for relation to failure of hardware parts. The model-based methodology is presented in D3.3.3, the overall safety analysis methodology is described in WT3.3.1.

Evaluation of Safety Goal Violations due to Random Hardware Failure

The evaluation of the residual risk of safety goal violation due to random hardware failures regarding single-point faults, residual faults and possible dual-point (multiple) faults shall be evaluated for each violation of each safety goal on respective item of ASIL B to D. Two methods can be used - either Probabilistic Metrics for random Hardware Failures (PMHF) which can be build by a quantification of a fault tree, or Failure Rate Class (FRC) method which evaluates each fault individually. Similar to safety analysis, WT3.2.2 will only cope with elements to capture component failure information, metrics target and results for relation to failure of hardware parts. The model-based methodology is presented in D3.3.3, the overall safety analysis methodology is described in WT3.3.1.

Part 7: Production and Operation

The relevant requirements for WT 3.2.2 arise from two sections of part 7, namely “Production” and “Operation Service”. As for this product cycles the requirement encompasses largely the hardware development, only the requirement related to hardware safety measure initiated during hardware product development will be considered.

Part 8: Supporting Processes

The relevant requirements for WT 3.2.2 arise from Part 8 “Supporting processes”, section 6 namely “Specification and management of safety requirements” and section 9 “Verification”. Section 13 “Qualification of hardware component” is in focus of work task WT3.2.4.

Specification and Management of Safety Requirements

The objective of this section of ISO 26262 is to ensure that all safety requirements are specified correctly with respect to their attributes and characteristics. In addition the management of the safety requirements and tracing during the entire safety lifecycle has to be consistent, in particular for hardware development as context of this task.

Part 9: Automotive Safety Integrity Level (ASIL)-oriented and Safety-oriented Analyses

The relevant requirements for WT 3.2.2 arise from three sections of part 9 “Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses”, namely section 7 “Analysis of dependent Failures” and section 8 “Safety Analyses” as reference for hardware element as introduced in System Design part 4 and Hardware development part 5. The section 4 related to “Criteria for co-existence of elements” and section 5 related to “requirement decomposition with respect to ASIL tailoring” is ensured WT3.1.1. Therefore, only from the first two sections an overview is given.

Analysis of Dependent Failure

The analysis of dependent failures on the architecture induces to introduce specific measure to be applied to the architecture element (e.g. such as redundancy, dissimilar development, safety mechanism, physical barrier, etc). A common cause failure and cascading failure analysis shall be performed for the architecture considering operational life of the product. This evaluation shall be performed for systematic faults, random hardware failures according to adequate required methods.

Safety Analyses

With the help of the safety analyses consequences of faults and failures on functions, behavior and design of items and elements shall be examined. The context of hardware elements is targeted in this task. Moreover, the analyses provide information on causes and conditions that could lead to the violations of a safety goal or safety requirement.

6 Safety Analysis Methods Interface

After presenting the relevant parts of ISO 26262 for hardware modeling and in addition to the primary goal of the representation of the Technical Safety Concept, the calculation of the hardware metrics and probabilistic value on hardware element shall be performed. It is essential that abstraction level of the hardware development is considered; meaning capability for separation of Hardware function and electronic component packaging during development and modeling. Furthermore, these models shall allow to perform safety analysis methods by first qualitative and then quantitative value for hardware element. It has been stated that the hardware package will include construct for hardware modeling, necessary constructs to perform quantitative measurement, such as failure mode and rate, and constructs to allocate or store results of the quantitative hardware analysis, such as Single Point Fault metric or Probabilistic Metric for random Hardware Failures.

The following chapter defines the boundary of the safety analysis methods interface, and interface element in detail.

6.1 Interface Methodology for Safety Analysis

The model based methods to perform safety analysis, in particular on hardware design to the failure and effect of faults as defined in ISO 26262-9:2011-Clause 8, is defined in the context of WT3.3.1 formally work task “Safety Analysis”. The outputs of an analysis per safety goal are: the identification of safety related attribute of the hardware component; the relation of the hardware component to the context of analysis as the safety goal or the sub-system malfunction in case of decomposition of the system; the typing of the elementary component fault as safe fault, single-point or residual fault and multiple-point latent; the identification of the safety mechanism covering the component fault. These outputs are required to enable the calculation of the hardware architecture metrics and the residual risk of violation of safety goal due to random hardware failure.

In addition the model-based development process foreseen by SAFE takes into account all the elements / attributes that potentially contribute to a safety risk on vehicle level. So, from vehicle items, all elements are decomposed according to engineering phase defined by the ISO26262 standard, being represented by the Functional Safety Concept and by the Technical Safety Concept. Then, according to the hardware development requirement from Part 5, the hardware architecture and detailed hardware design shall be captured to allow then further iterative safety analysis.

The architecture principle selected for the consideration of these needs is based on abstraction view and viewpoint, capable to capture and interconnect all relevant artifacts. The resulting architecture which is used is presented in the Figure 2.

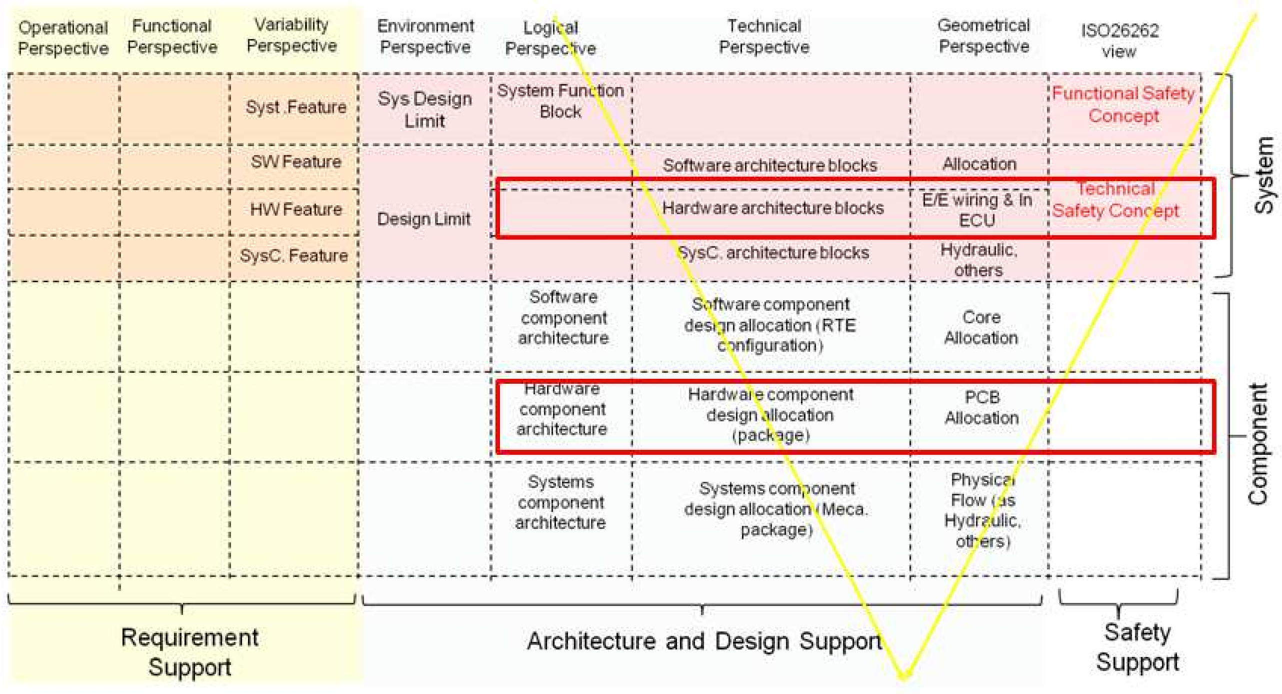

Operational Perspective
Functional Perspective
Variability Perspective
Environment Perspective
Logical Perspective
Technical Perspective
Geometrical Perspective
ISO26262 view
Syst .Feature
Sys Design Limit
System Function Block
Functional Safety Concept
SW Feature
Software architecture blocks
Allocation
HW Feature
Design Limit
Hardware architecture blocks
E/E wiring & In ECU
Technical Safety Concept
SysC. Feature
SysC. architecture blocks
Hydraulic, others
System
Software component architecture
Software component design allocation (RTE configuration)
Core Allocation
Hardware component architecture
Hardware component design allocation (package)
PCB Allocation
Systems component architecture
Systems component design allocation (Meca. package)
Physical Flow (as Hydraulic, others)
Component
Requirement Support
Architecture and Design Support
Safety Support

Figure 2: Overview on structure of architecture (Relevant parts highlighted)

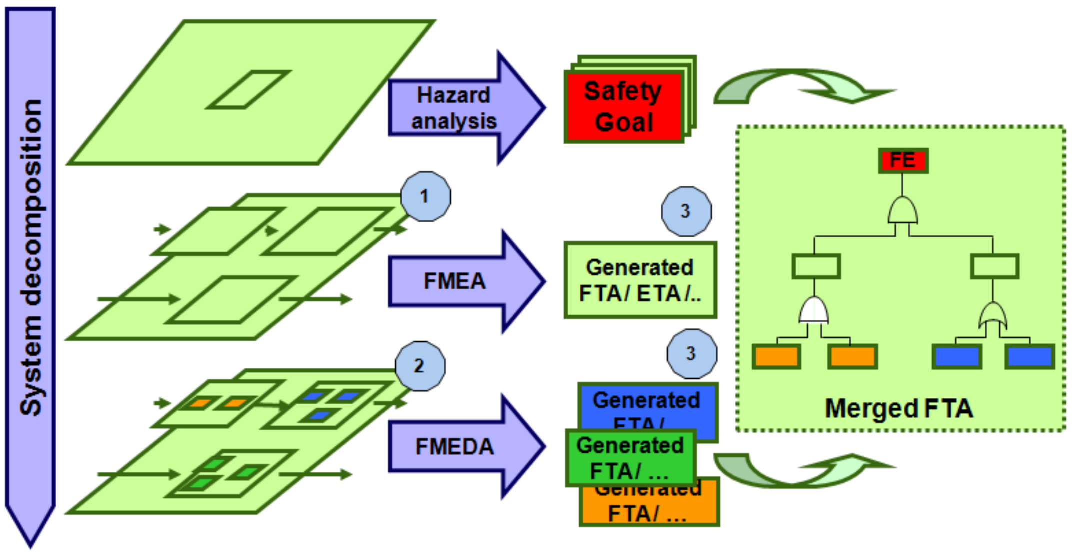
As introduced in task description, the hardware description is mapped to the existing language EAST-ADL. The EAST-ADL is structuring functional decomposition and architectural element definition in the Design abstraction view of EAST-ADL and the Implementation view AUTOSAR. The mapping of view point for hardware development in accordance to Figure 2 is conform to

- Hardware architecture is represented by EAST-ADL Hardware Design Architecture
- Hardware detailed design is represented by AUTOSAR HW Element from ECU Resource Template

It can be noticed that as Hardware Design Architecture of EAST-ADL is also capable to represent Hardware Detailed Design, methods proposed shall allow the support of compatible interface required by Safety Analysis.

Finally, the safety analysis analyzing hardware component failure and identifying their fault classification (single-point or residual…) shall be visible at the hardware architecture level. This iterative process of failure analysis allows to iteratively introduce safety mechanism and mitigation effect, and to validate their impact and efficiency. The process is not intended to be detailed here, but simply showing that hardware architecture will evolve according to safety analysis and technical safety requirement management and refinement. The Figure 3 below, represent a general overview of the iterative process that will be considered in WT3.3.1 according concrete method selection.

The given assumption for WT3.2.2 is that component fault classification, the safety related component tag and the relation of the component to the safety/malfunction is given from this safety analysis. In addition this analysis is also built on the top of the hardware architecture composed of hardware element and hardware safety mechanism, the traceability of safety mechanism to the component fault mitigates, and finally by the fault propagation methodology.



Step 1: Elementary block failure mode analysis (Dysfunctional behavior)

Step 2: Tag of each block safety contribution (function, diagnosis, mechanism…)

Step 3: Generation of propagation for Qualitative analysis (FTA)

Deductive methods : FTA: Fault Tree Analysis
Inductive Methods : FMEA: Failure Mode and Effect Analysis / FMEDA : Failure Mode, Effect and Diagnosis Analysis

Figure 3: Overview on iterative safety analysis methods

Moreover, the hardware development process may then, depending of industrial process, perform allocation of Hardware Component in Hardware Part in consideration of electronic industrial process (e.g. see example in Figure 4 below). Such separation of concern shall then consider the inter-relation between fault characteristic at architecture level and origin from fault at design level.

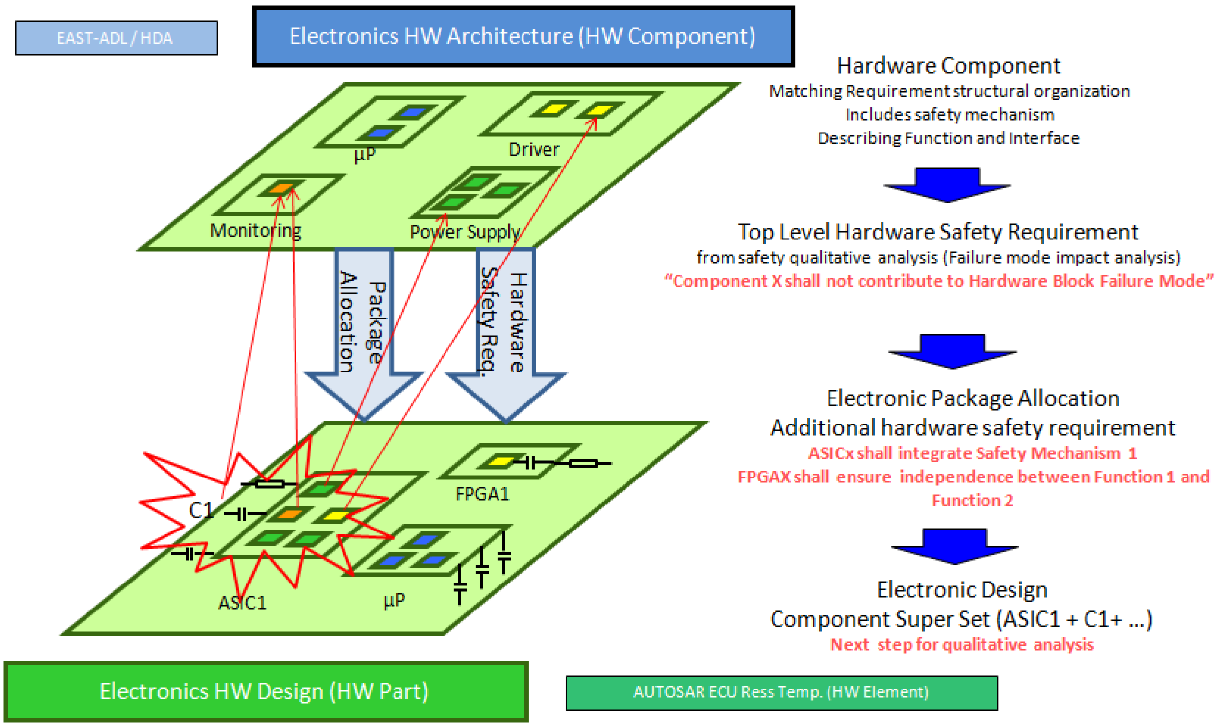

EAST-ADL / HDA
Electronics HW Architecture (HW Component)
µP
Driver
Monitoring
Power Supply
Package Allocation
Hardware Safety Req.
C1
FPGA1
ASIC1
µP
Electronics HW Design (HW Part)
AUTOSAR ECU Ress Temp. (HW Element)
Hardware Component
Matching Requirement structural organization
Includes safety mechanism
Describing Function and Interface
Top Level Hardware Safety Requirement
from safety qualitative analysis (Failure mode impact analysis)
"Component X shall not contribute to Hardware Block Failure Mode"
Electronic Package Allocation
Additional hardware safety requirement
ASICx shall integrate Safety Mechanism 1
FPGAX shall ensure independence between Function 1 and Function 2
Electronic Design
Component Super Set (ASIC1 + C1+ ...)
Next step for qualitative analysis

Figure 4: Hardware allocation and quantitative analysis

So, the safety analysis is performed on the impact analysis of the failure mode of the Hardware Component. In the context of the architecture, the hardware components are tagged as safety related and its failures modes are characterized as safe fault, single-point or residual fault and multiple-point latent fault. The corresponding failure modes of the hardware component are considered as malfunction for the electronic design. The quantitative values are computed from this fault consideration and from the diagnostic coverage of hardware element identified as safety mechanism. Such measures are the hardware architectural metrics as Single-Point Fault metric or Multiple Point-Fault and Latent metric, plus the Probabilistic Metrics for random Hardware Failure or the individual Failure Rate Class evaluation.

The necessary failure rate and distribution, only available at the hardware part level, shall then be combining to retrieve computed failure rate at the architecture level for each failure mode of the hardware component considered. The correspondence will be performed by the quantification of the hardware component malfunction. SAFE meta model constructs shall allow to store this different failure information and calculation relation using self define formula. It shall also permit to define target values and store results of the quantitative hardware analysis. We propose to store in constructs by WT3.2.2: the definition of formula for quantitative measurement as relevant failure information is store in modeling element. From this interface defined in WT3.2.2, the tools and methods specification of WT3.3.1 as D3.3.1.b deliverable will validate the initial formula for calculation on the top of actual information provided in this chapter and related SAFE model element in chapter 9.3. Moreover WT3.3.3 as architecture benchmark analysis makes use failure and metric, and will provide a context for validation (see specification D3.3.3[8])

Key Steps of Hardware modeling and analysis

Based on the considerations described above the key steps of the methodology for hardware modeling and safety analysis can be formulated as below, and shall consider assumption for WT3.3.1 work task in the overall detailed methodology. The key steps are identified as:

- Capture Hardware Technical Safety Concept (with Hardware Component)
- Complete Hardware Component Failure Propagation (Iterative process for Safety Mechanism validation)
- Define (or Reuse) initial failure rate data for hardware components and calculates metrics
- Define Hardware Component allocation and Malfunction (from Hardware Component into complex parts such as ASIC, FPGA)
- Develop Electronics Schematic and capture (or reuse existing) Hardware Part
- Perform/Reuse Electronic part detailed failure analysis (e.g. FMEA) and contribution to Hardware component malfunction
- Verify hardware component Metrics and Probabilistic value

6.2 Interface Element

The split decided in the work task organization between safety analysis methods from WT3.3.1, hardware architecture assessment from WT 3.3.3 and hardware meta model from WT3.2.2, was that, in addition to hardware component and hardware part, the SAFE construct for hardware modeling will include: hardware failure related information, calculation constructs necessary for hardware architectural metrics and for the two methods for evaluation of the residual risk for violation of the safety goal.

Moreover, constructs shall provide the relationship of the formula calculation for computation of Hardware Component failure rates from Hardware Part failure rate and distribution value from industry source).

The list of artifacts, consolidated by WT2.1 analysis and derivation requirement synthesized in 7, is initiated from following concept:

- Failure Mode, Failure Rate and Distribution for Hardware Part (to be imported from industry source)
- Failure Mode and Failure Rate of Hardware Component
- Fault Enumeration to allow Failure Mode classification of a Hardware Component in the type in context of an overall hardware architecture
- Identification of Safety Related impact of the Hardware Component
- Formula to provide relation and perform calculation from Hardware Part to Hardware Component in the context of an electronic design and the given hardware malfunction for the design element
- Hardware architectural metric target values and results for Single-Point Fault Metric and Latent-Fault Metric
- Probabilistic Metrics for random Hardware Failure (today simplified approach) target values and results
- Failure Rate Class target values , values for each Hardware Component and defined measures
- Formula to perform calculation required for architectural metrics, probabilistic metrics and failure rate class, depending of Hardware Component Failure Rate, potential Diagnostic Coverage of the selected Safety Mechanism
- Relation to the top level malfunction (linked to the Safety Goal of the item) of the hardware architecture , to allow evaluation for each Safety Goal (direct or indirect evaluation)

The concrete details of the meta model elements is defined in section 9.3.

Notice that as defined in previous section, thanks to the expressiveness of Hardware Design Architecture from EAST-ADL capable to represent Hardware Detailed Design, the constructs provided could allow completing the calculation directly from Hardware Component model, and so preventing using elements of Hardware Part if convenient.

7 Hardware modeling scoping

In the work of WT2.1 the ISO 26262 was analyzed into detail. Requirements were elicited from each part of the standard and textually described with the corresponding ISO references. For WT3.2.2 - hardware modeling – requirements out of part 4, 6, 7, 8, 9 and in particularly Part 5 had to be considered. Derived work task specific requirements describe all necessary characteristics for the meta model extension of WT3.2.2, to provide hardware modeling on hardware architecture level and detailed level of hardware electronic design for hardware safety evaluation. To provide structure and traceability in managing the work task specific requirements, the relevant ones were categorized by their impact on the hardware model for the SAFE meta models extension. Based on the requirements elicitation five categories were derived and introduced: requirements for hardware components, hardware failures, hardware architectural metrics, safety goal violation and traceability. The scope of the work task hardware modeling regarding meta model constructs is to provide all necessary information for structural and failure description of hardware components as well as constructs for the evaluation of hardware with regard to hardware architectural metrics and evaluation of safety goal violation according to ISO 26262 Part 5, Clause 8 and 9.

The presented categories contain all requirements for SAFE meta model extension and are explained into detail in the next sections. Please notice that the refined requirements are not reported below, as these categories where build to provide an initial structure for the SAFE meta model contribution as detailed in section 7.6.

7.1 Requirements Package: Hardware Components

Requirements regarding the structure of hardware components and parts for hardware architecture and hardware electronic design were collected in the category hardware component. To facilitate safety evaluation of a hardware design, the hardware components and their interference have to be described into detail according to the needs in ISO 26262, Part 5. The requirements for hardware component structure are partially related with existing EAST-ADL and AUTOSAR constructs. As the requirement collected for Design Environmental Condition and Special Characteristics deals with constraints description for design operation and then production, operation, decommissioning and maintenance, they can be express through Requirement EAST-ADL constructs and so are not considered in additional meta modeling artifacts.

The package hardware component addresses the description of hardware components and parts as well as composition of components or parts including port and pin connections. Hardware/Software-Interfaces facilitate the presentation of hardware which is controlled by software. The representation of elementary hardware components and the categorization of hardware components are also included.

7.2 Requirements Package: Hardware Failure

The category hardware failure groups all requirements of the ISO 26262 regarding the relevant failure description of hardware components and parts. A meta model extension for the failure description is related to capture all requirements.

The package hardware failure captures the description of different failure modes and a failure rate of hardware components and parts including potential causes of the failure mode, the failure rate distribution of the failure mode and contribution to the malfunction (linked to violation of a safety goal). Safety mechanisms with their diagnostic coverage are also addressed.

7.3 Requirements Package: Hardware Architectural Metrics

The hardware architectural metrics, described in ISO 26262 Part 5 Clause 8, provide the first safety evaluation of the hardware architecture claimed by the ISO. All requirements to perform this evaluation as well as the methodology, calculation and results are collected in this requirements package.

The package hardware architectural metrics captures the single contribution of each violating failure mode as a specific failure rate, according to its classification. Target values for the architectural metrics are provided.

7.4 Requirements Package: Safety Goal Violation

The evaluation of residual risk of safety goal violation is the second safety evaluation claimed by the ISO 26262 and is described into detail in Part 5 Clause 9. All requirements which are relevant for both methods, the *Probabilistic Metric for Random Hardware Failure* (PMHF) and the *Failure Rate Class* (FRC) approach, are grouped in this category.

This requirement packages addresses all necessary calculations for the evaluation of safety goal violation as well as target values. Exposure time for dual-point faults and required dedicated measures are included. Additionally, diagnostic coverage on hardware component level is described.

7.5 Requirements Package: Traceability

The traceability of safety requirements such as safety goals regarding the evaluation of the hardware architecture is provided by the requirements in the category traceability. These requirements are in focus work task WT3.1.2 for the “Safety Requirement Expression”.

The package traceability addresses the dependency of technical and functional requirements. Additionally, the links of hardware components to hardware safety requirements and the traceability from a preliminary design to hardware components at electronic level are captured.

The modeling elements used for traceability have been centralized into the SAFE meta model Requirements package. For more details on constructs please refers to [10].

7.6 Allocation of the requirements packages to derived meta model structure

A structure for the meta model was derived from the structure of the requirements categorization. Therefore, the meta model contains the following sub-packages in the package *Hardware*:

- Sub-Package *Structure*, according to the requirements category hardware components as change request for EAST-ADL and AUTOSAR
- Sub-Package *Failure*, according to the requirements category hardware failure related to hardware components. Additionally, the quantitative assessment for the calculation of single contribution for each failure mode for hardware components is included.
- Sub-Package *FailurePart*, according to the requirements category hardware failure related to the hardware part. Additionally, the quantitative assessment for the calculation of single contribution for each failure mode for hardware parts is included.
- Sub-Package *HWQuantitativeMeasure* for the classification of the assessments to the architectural metrics or probabilistic methods for hardware safety evaluation.
- Sub-Package *HWArchitecturalMetrics*, according to the requirements category Hardware Architectural Metrics

- Sub-Package *ProbabilisticMethods*, according to the requirements category Safety Goal Violation
- An additionally package *FailureFormula* contains all formula expressions required for the evaluation of hardware. This has to include the quantitative measures and the previous calculations exemplarily, of the single failure mode contributions.

8 Performing Hardware Modeling based on EAST-ADL

In this section the current status of the architecture description language EAST-ADL regarding hardware is described. Based on the investigation a proposal for adaption and extension of existing constructs is provided to facilitate an evaluation of detailed hardware architectures regarding functional safety in accordance with ISO 26262.

8.1 Current status of EAST-ADL

EAST-ADL provides the description of an automotive architecture on different levels of abstraction. This namely is the vehicle level, analysis level, design level, implementation and operational level. This architecture description language was developed in various projects together with Original Equipment Manufacturers (OEMs), suppliers and research institutes. Current published version of EAST-ADL is version 2.1, see also www.east-adl.info.

The class diagram *PackageDependencies* of EAST-ADL V2.1[5] gives an overview of the dependencies of the package and is presented in Figure 5. Beside the described abstraction layers, especially the sub-package *HardwareModeling* and the package *Dependability* are in special interest for hardware and failure modeling. This has to be related with the hardware evaluation including the architectural metrics and the probabilistic methods.

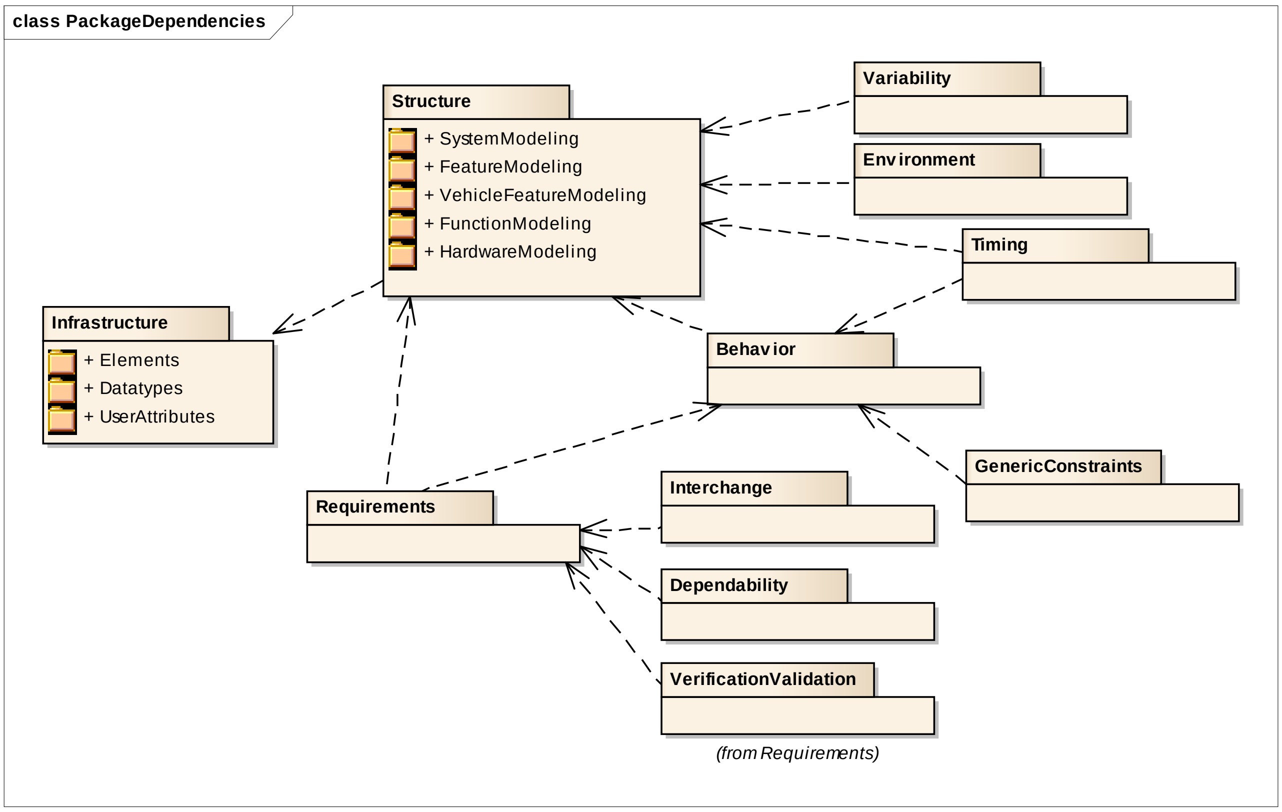


Figure 5: Class diagram for Package Dependencies

In the sub-package *HardwareModeling* of the package *Structure*, EAST-ADL V2.1 describes the hardware modeling in the corresponding diagram. The construct HardwareComponentType and HardwarComponentPrototype provides a structural entity that defines a part of an electrical architecture [5], as shown in Figure 5. Further class of interest are the *HardwareConnector*, *HardwarePin* and *HardwarePinGroup*, as the can be used for the description of the electrically connection of hardware components regarding their logical bus between ports of the hardware component.

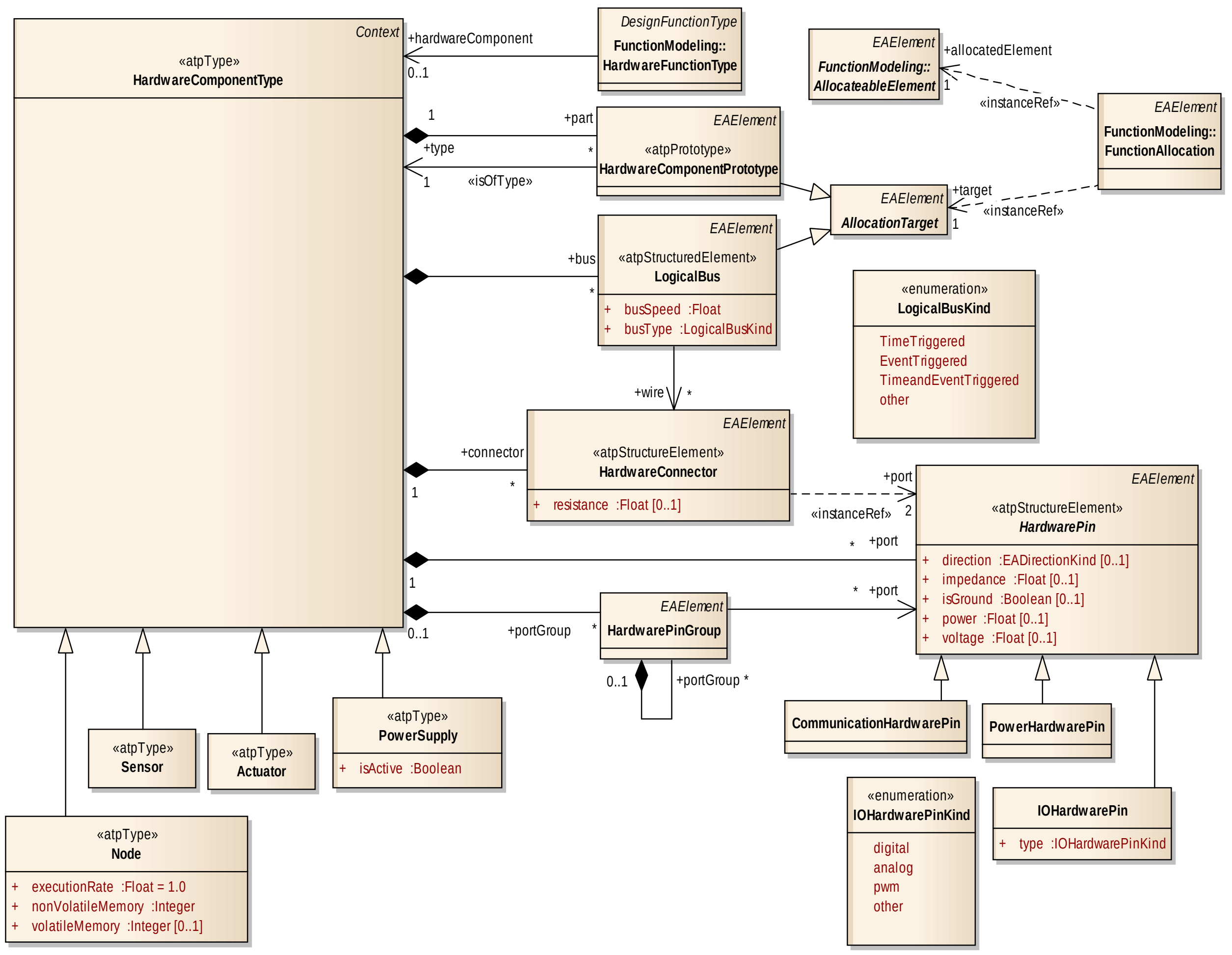


Figure 6: Class diagram for Hardware Modeling in the EAST-ADL2

The proposed use of hardware construct HardwareComponentType in Design Level of EAST-ADL2.1 methodology is to build the hardware node and topology including sensors and actuators, to define the allocation of functional block as *DesignFunctionType*. Notice that the HardwareComponentType allows further decomposition to be able to decompose an ECU *node*. But the *DesignFunctionType* can be specialized, as visible in the Figure 7, as hardware via *HardwareFunctionType* or software with *DesignFunctionType* or *LocalDeviceManager* to interface a *Sensor* or *BasicSoftwareFunctionType* as a general basic software module. Moreover, the be-havior of the function *FunctionBehavior* is associated to the *FunctionType*. So the top level *FunctionType* represent functional chain of hardware and software element, as *DesignFunctionType*, where *HardwareComponentType* are simply a container, via *allocation* link, for *HardwareFunctionType*. So the use of Design Level is still a functional approach, as software and hardware and not completely split.

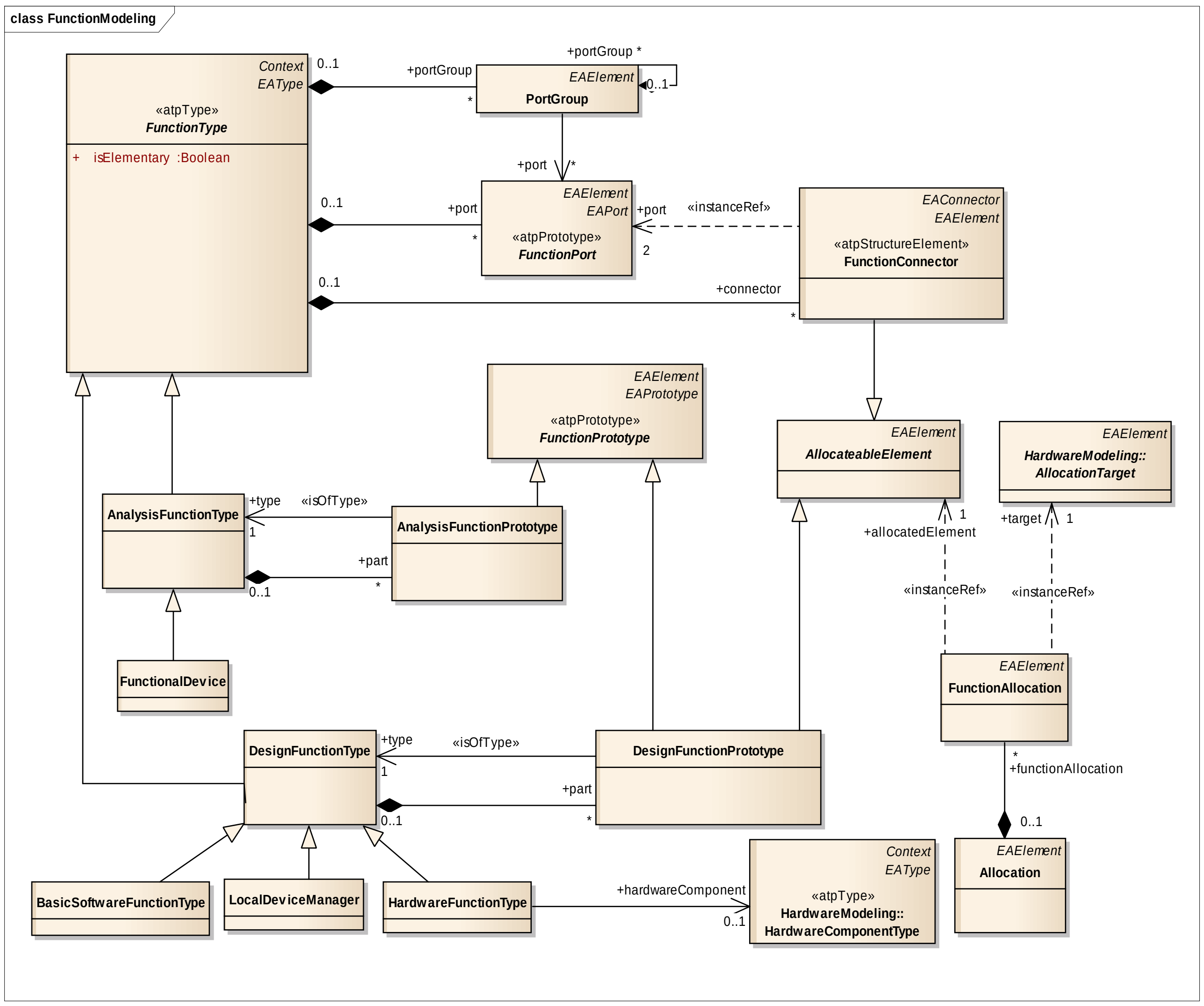

class FunctionModeling
Context EAType «atpType» FunctionType
+ isElementary :Boolean
EAElement PortGroup
+portGroup
+portGroup *
EAElement EAPort «atpPrototype» FunctionPort
+port
«instanceRef»
EAConnector EAElement «atpStructureElement» FunctionConnector
+connector
EAElement EAPrototype «atpPrototype» FunctionPrototype
EAElement AllocateableElement
EAElement HardwareModeling:: AllocationTarget
AnalysisFunctionType
AnalysisFunctionPrototype
«isOfType»
+type
+part
+allocatedElement
+target
FunctionalDevice
EAElement FunctionAllocation
DesignFunctionType
DesignFunctionPrototype
+functionAllocation
EAElement Allocation
BasicSoftwareFunctionType
LocalDeviceManager
HardwareFunctionType
+hardwareComponent
Context EAType «atpType» HardwareModeling:: HardwareComponentType

Figure 7: Class diagram for Function Modeling in the EAST-ADL2

Then for the failure part, in the sub-package *ErrorModel* of the package *Dependability*, EAST-ADL2.1 describes the error modeling in the corresponding diagram, as shown in Figure 8. Propagation points for faults can be described by the class *FaultInPort* and *FailureOutPort*, while the *FaultFailurePort* describes an abstract port for faults and failures and depends on a hardware pin. The constructs *ErrorModelType* and *ErrorModelPrototype* provides a hierarchical composition of error models. The connection of the *ErrorModel* with the structural element *FunctionType* and *HardwareComponentType* is made via respective allocation link as *errorModelPrototype_hwTarget* for *HardwareComponentPrototype* and *errorModelPrototype_functionTarget* for *DesignFunctionPrototype* (with relevant specialization from Figure 7).

A typical target of the *ErrorModelType* is exemplarily a system/subsystem, a function or a hardware device and represents the internal faults and the fault propagation of the targeted element. From the EAST-ADL2.1 Design Level modeling methodology, as introduce above, the functional approach applied to *ErrorModel* for safety analysis constraints the use of *ErrorModel* for *HardwareComponent* to describe hardware fault that propagates Failure to *DesignFunction* (hardware or software functional behavior) as a hardware resource failure. The signal fault propagation is supported by the *ErrorModel* of *HardwareFunctionType*. In the physical electrical domain this split of concern is not visible.

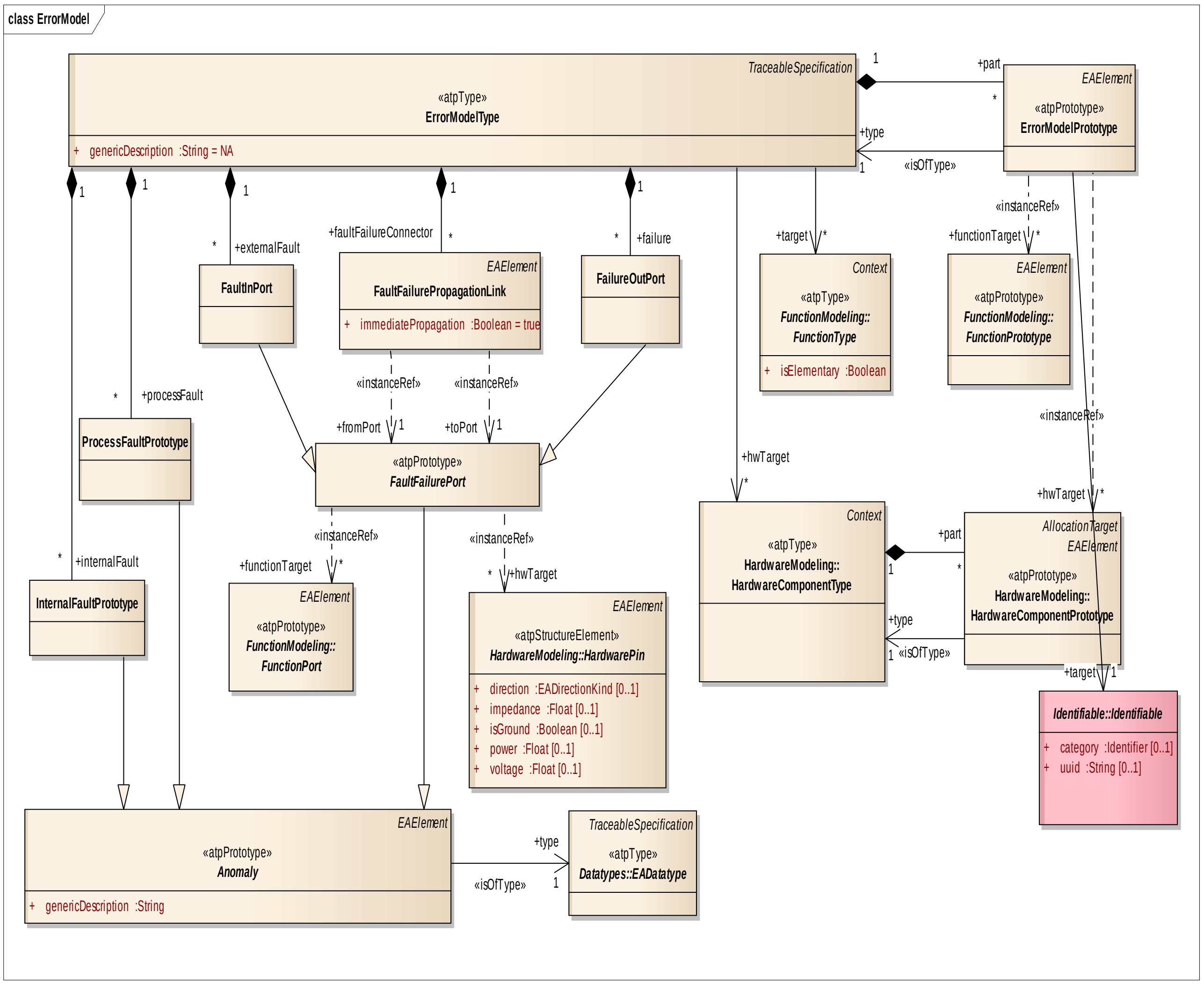


Figure 8: Class diagram for Error Modeling in the EAST-ADL2 Dependability

In the sub-package *ErrorModel* of the package *Dependability* EAST-ADL 2.1, describes the error behavior in the corresponding class diagram *ErrorBehavior*, as shown in Figure 9. The presented different faults can have the following different roles: external, internal or process faults. While class *FailureOutPort* and *FaultInPort* represent the described propagation points, the *InternalFaultPrototype* represents an internal condition of the target that concerns the components faults/failure definition.

For the stake of fault of hardware part, the internal fault as *InternalFaultPrototype* represents the failure mode of the *HardwareComponent*. The others relevant information for quantitative assessment as failure rate and distribution are not clearly defined. A construct *QuantitativeSafetyConstrainst* is present but only associate to a *FaultFailure* as an instance reference of an *Anomaly*, as the top level failure effect of an *ErrorMode*l as typed *FailureOutPort*.

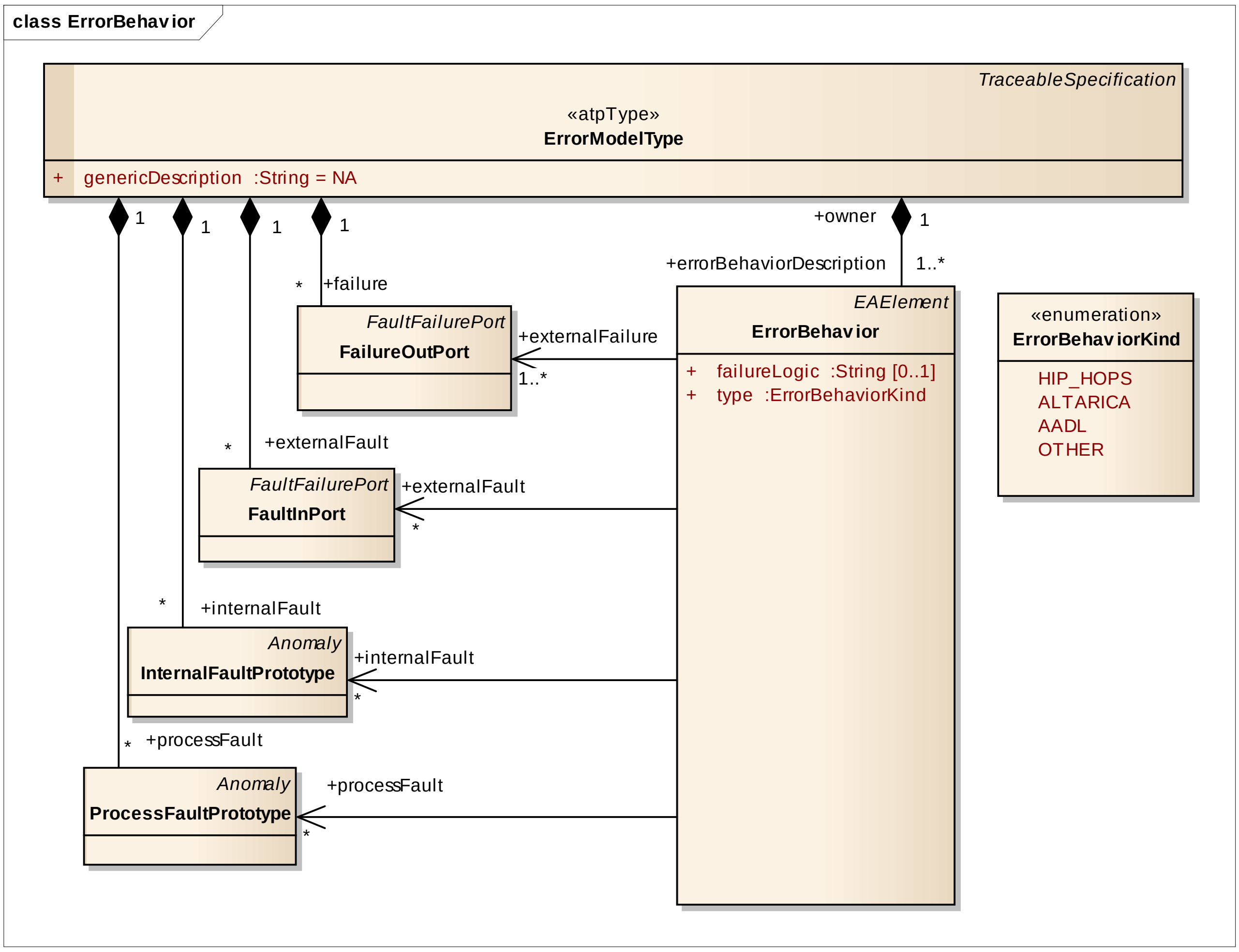


Figure 9: Class diagram for Error Behavior in the EAST-ADL2 Dependability

8.2 Proposed extensions to EAST-ADL

Basic constructs needed for structural description of hardware exists in EAST-ADL V2.1, as shown in Figure 6. With regard to the elicited requirements of ISO 26262 these concepts and constructs can cover and fulfill high level description of hardware node and sensors/actuators. Inconveniences exist for the interconnection of hardware components on the abstraction of low level electronics. To model hardware architectures on detailed level to perform the demanded metrics, constructs for the structural description has to be provided, exemplarily for hardware ports, pin and their specific connectors. Additionally, a Hardware-Software-Interface (HSI) has to be introduced, claimed by the ISO 26262. Therefore, an adaption of the structural part for the hardware modeling has to be provided. Existing artifacts in EAST-ADL shall be referenced and linked, as it should be objective to reuse as much as possible of the existing structural constructs for the SAFE meta model extension. We propose for the structural part a change request of EAST-ADL. The corresponding meta model adaption is presented in Section 9.2.

Beside the structural part, the specific requirements for hardware modeling presented in Section 7 claim the description of hardware failure information and the metrics for qualitative and quantitative analysis. Beside the concepts for error modeling with the definition of propagation points the EAST-ADL V2.1 provides no constructs for failure information. To provide failure modes, failure rates of hardware components etc. the existing constructs have to be extended. For the qualitative and quantitative assessment of the hardware failure expressions have to be formulate and constructs for storage of the results,

These potential extensions together with their rational are described in the Section 9.3. However, as this task is still going on in future also the potential extensions will be elaborated in more detail.

8.3 Current status of AUTOSAR

As proposed by EAST-ADL abstraction view, AUTOSAR provides the implementation view that represents the software oriented implementation. For the hardware related part, in particular in AUTOSAR the ECU Resource Template, main elements capable to represent hardware design element are available. As it is depicted in Figure 10, the basic class *HwElement* exists. This element can be composed of *HwPin* through the intermediate class *HwPinGroup*. Then a connector can connect two *HWElement* by a *HwElementConnector* and then connect *HwPin* via *HwPinConnector* or *HwPinGroup* via *HwPinGroupConnector*.

So, we can represent a nested composition like of *HwElement* by using the *nestedElement* relationship, knowing that in term of semantic this is not a strict composition.

By such means an ECU can be defined as nested *HwElement*, connected together by their *HwPin*, *HwPinGroup*, to represent all the electronics Hardware Part and to define a complete ECU electronic schematic as hardware electronic design level. As explain in the next section, there is place for improvement in order to align concept with HW Component and compositional organization of an ECU organization.

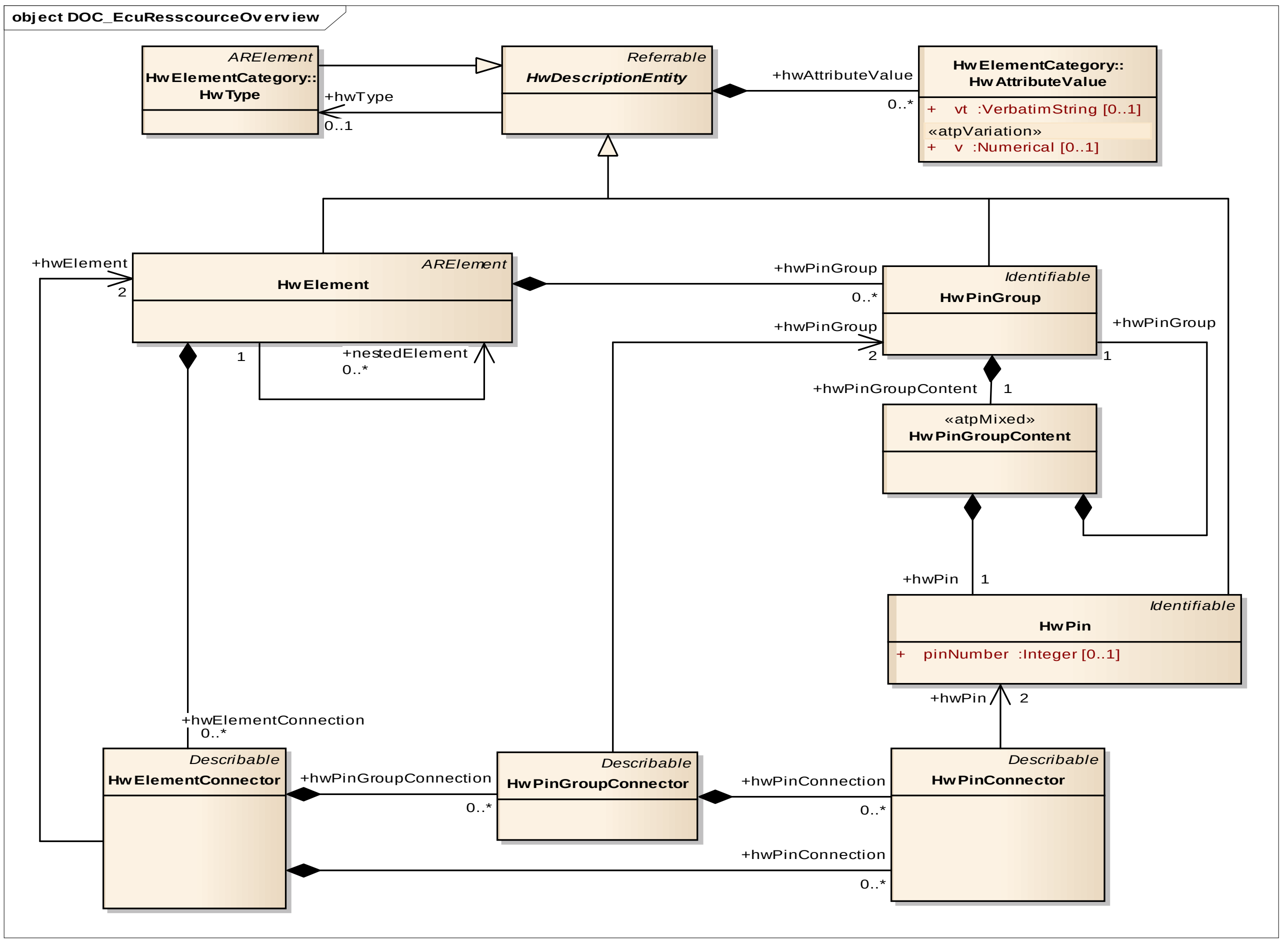


Figure 10: AUTOSAR ECU Resource overview

8.4 Proposed extensions to AUTOSAR

As Introduced in the previous section, to facilitate hardware part representation and compositional aspects, the ECU resource template requires some improvement. Due to AUTOSAR IPs, we will only express needs and then propose to submit this subject to the AUTOSAR consortium as a potential improvement area for a future official change request.

The draft of the main features to be change in ECU Resource template is the introduction of compositional capability by the creation of HwElementType composed of part from *HwElementPrototype*. Another possible of change would be to revise *HwPinGroup* definition in order to introduce the concept of Bus, in order to be more restrictive in the *HwPin* composition.

9 WT 3.2.2 Contribution to SAFE Meta-Model

Within this section the contribution of WT 3.2.2 to the SAFE meta-model is described. At the beginning an overview about the model is given which is followed by the detailed description of the classes and interconnections. Moreover, in another section the meta-model is described by means of an example.

9.1 Overview

The structuring of the meta model extension regarding hardware is done according to the categories defined in Section 7.6 as shown in Figure 11.

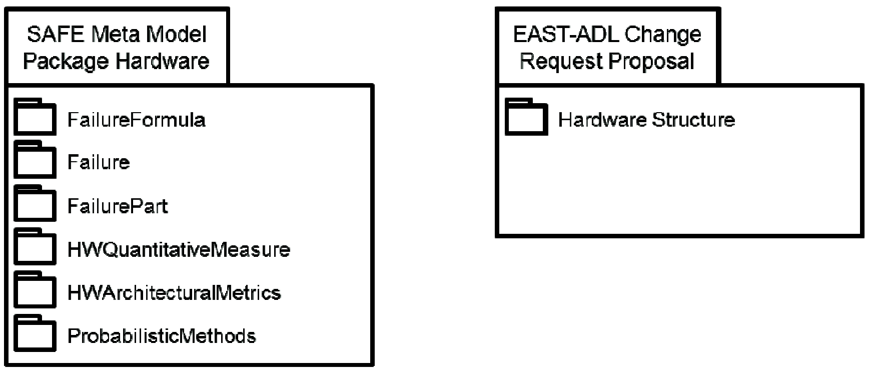


Figure 11: Overview on WT 3.2.2-contribution to SAFE meta-model

The top-level package *Hardware* of the SAFE meta model, developed in Enterprise Architect, contains all meta model extension of WT 3.2.2, except for the structural part. The meta model adaption for EAST-ADL capturing the structural part is described in Section 9.2, as the decided choice was to shift it away from the package *Hardware* and make proposal for EAST-ADL2.1 adaptation in *HardwareStructure.*

The package *Hardware* with its sub-packages FailureFormula, Failure, FailurePart, HWQuantitativeMeasure, HWArchitecturelMetrics, ProbabilisticMethods is described in Section 9.3.

Due to the fact, that the meta model regarding hardware is partially based on the existing constructs of EAST-ADL, a lot of references are included. Figure 12 gives an overview of the references to EAST-ADL which are used in the package Hardware. In case of a reference, all attributes from the EAST-ADL class are inherited. For some classes of EAST-ADL adaptations are required, described in Section 9.2.

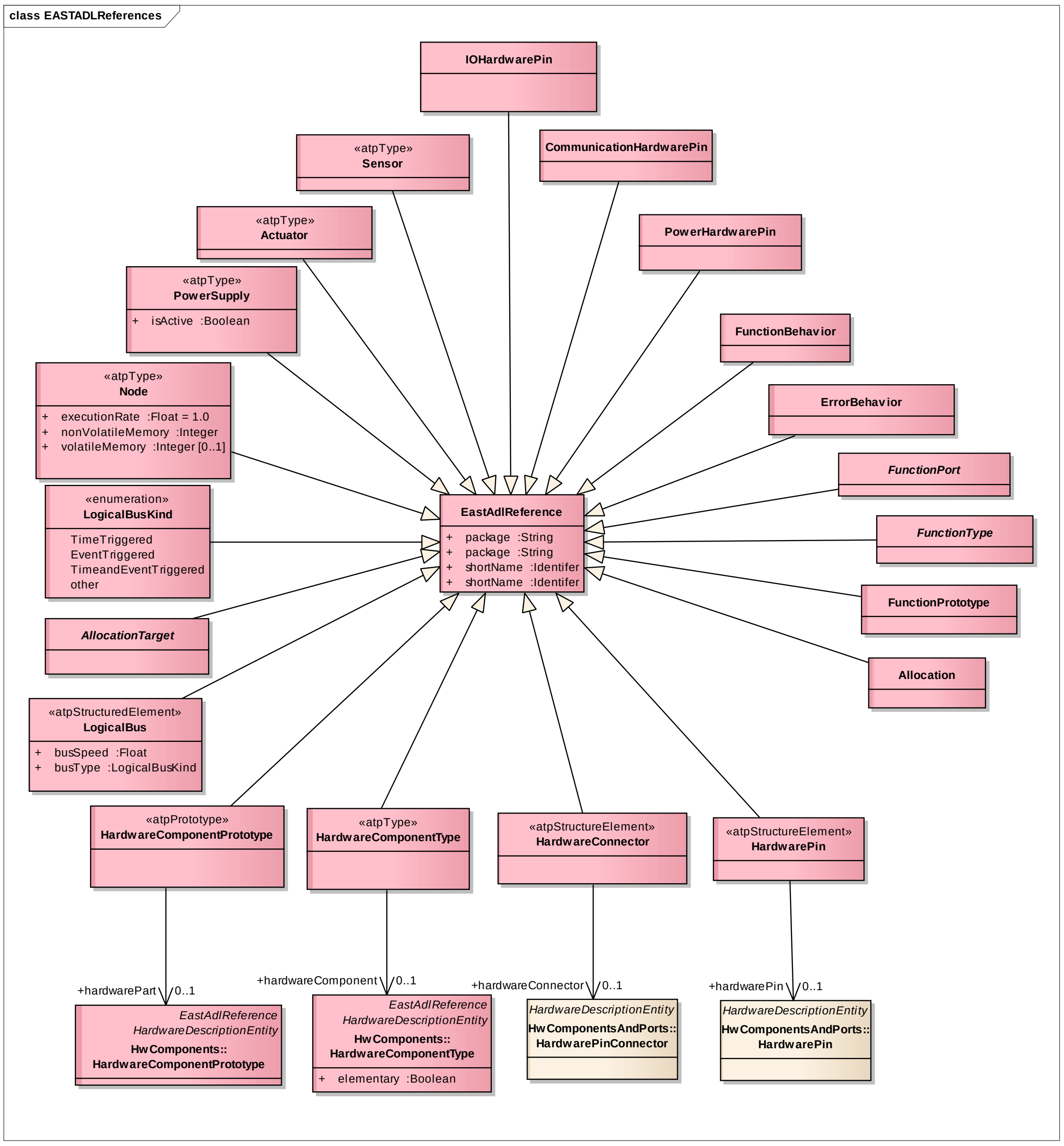

class EASTADLReferences
IOHardwarePin
«atpType» Sensor
CommunicationHardwarePin
«atpType» Actuator
PowerHardwarePin
«atpType» PowerSupply
+ isActive :Boolean
FunctionBehavior
«atpType» Node
+ executionRate :Float = 1.0
+ nonVolatileMemory :Integer
+ volatileMemory :Integer [0..1]
ErrorBehavior
FunctionPort
«enumeration» LogicalBusKind
TimeTriggered
EventTriggered
TimeandEventTriggered
other
EastAdlReference
+ package :String
+ package :String
+ shortName :Identifer
+ shortName :Identifer
FunctionType
FunctionPrototype
AllocationTarget
Allocation
«atpStructuredElement» LogicalBus
+ busSpeed :Float
+ busType :LogicalBusKind
«atpPrototype» HardwareComponentPrototype
«atpType» HardwareComponentType
«atpStructureElement» HardwareConnector
«atpStructureElement» HardwarePin
+hardwarePart 0..1
+hardwareComponent 0..1
+hardwareConnector 0..1
+hardwarePin 0..1
EastAdlReference HardwareDescriptionEntity HwComponents::HardwareComponentPrototype
EastAdlReference HardwareDescriptionEntity HwComponents::HardwareComponentType
+ elementary :Boolean
HardwareDescriptionEntity HwComponentsAndPorts::HardwarePinConnector
HardwareDescriptionEntity HwComponentsAndPorts::HardwarePin

Figure 12: References of package Hardware to EAST-ADL

9.2 Proposal for change request on EAST-ADL

This following section will describe the details of the proposal for change request in EAST-ADL2.1. It covers the core feature of EAST-ADL in the structural part of the hardware element.

The first main change represents the introduction of the *HardwarePort*, for substitution on the long run the *LogicalBus* meta class. This *HardwarePor*t can then be composed by *HardwarePin*, and *HardwarePort* will represent a transactional description of internal or external bus communication, similar to a concept available in IP-XACT (and in AUTOSAR *HwPinGroup*). As a consequence the *HardwareConnector* will be revised (see next section for details). Linked by the *HardwareElementEntities* generalization, the description of the electrical characteristics of the *HardwarePin* or any other hardware elements need to more flexible expressed. Our proposal is to reuse the *HwCategory* modeling concept from AUTOSAR (see next section and in AUTOSAR document for more details)

The second important change is the creation of the means for a separation at the Design level between hardware and software elements, as required by the ISO26262 requirement. The software architectural element could l be represent by design function (*DesignFunctionType*) and the hardware architectural element by hardware component (*HWComponentType*). As consequence, first a dedicated element shall be added to represent the hardware software interface, a *HwSwInterface* element representing the hardware abstraction (*HWAbstractionFunction*). Moreover them to complete the split, a behavior of the HW component shall be directly attached (*FunctionBehavior*), similar to the behavioral that is attached to *DesignFunction*. For example in hardware domain these behavior may be link to SystemC modeling element including the hardware behavior description for simulation capabilities.

In the following subsections, the detailed description of the classes and interconnections is detailed. Name of the top-level package is “Hardware Structure”. This on the other hand contains 6 sub-packages, as following

- HwCategory
- HwComponentBehavior
- HwComponent
- HwComponentsAndPorts
- HwSwInterface
- _instanceRef

9.2.1 Package Hardware Structure

The package *HardwareModeling* contains the elements to model physical entities of the embedded electrical/electronic system. These elements allow the hardware to be captured in sufficient detail to allow preliminary functional allocation decisions. It also allow to define the hardware architecture description based on hardware component and associated behavior.

Conversely, the Functional Analysis Architecture and the Functional Design Architecture may be revised based on analysis using information from the Hardware Design Architecture. An example is control law design, where algorithms may be modified for expected computational and communication delays and then finally attached to hardware component. Thus, the Hardware Design Architecture contains information about properties in order to support, e.g., timing analysis and performance in these respects. Finally, it includes behavioral description of the control law when decision for hardware implementation is made.

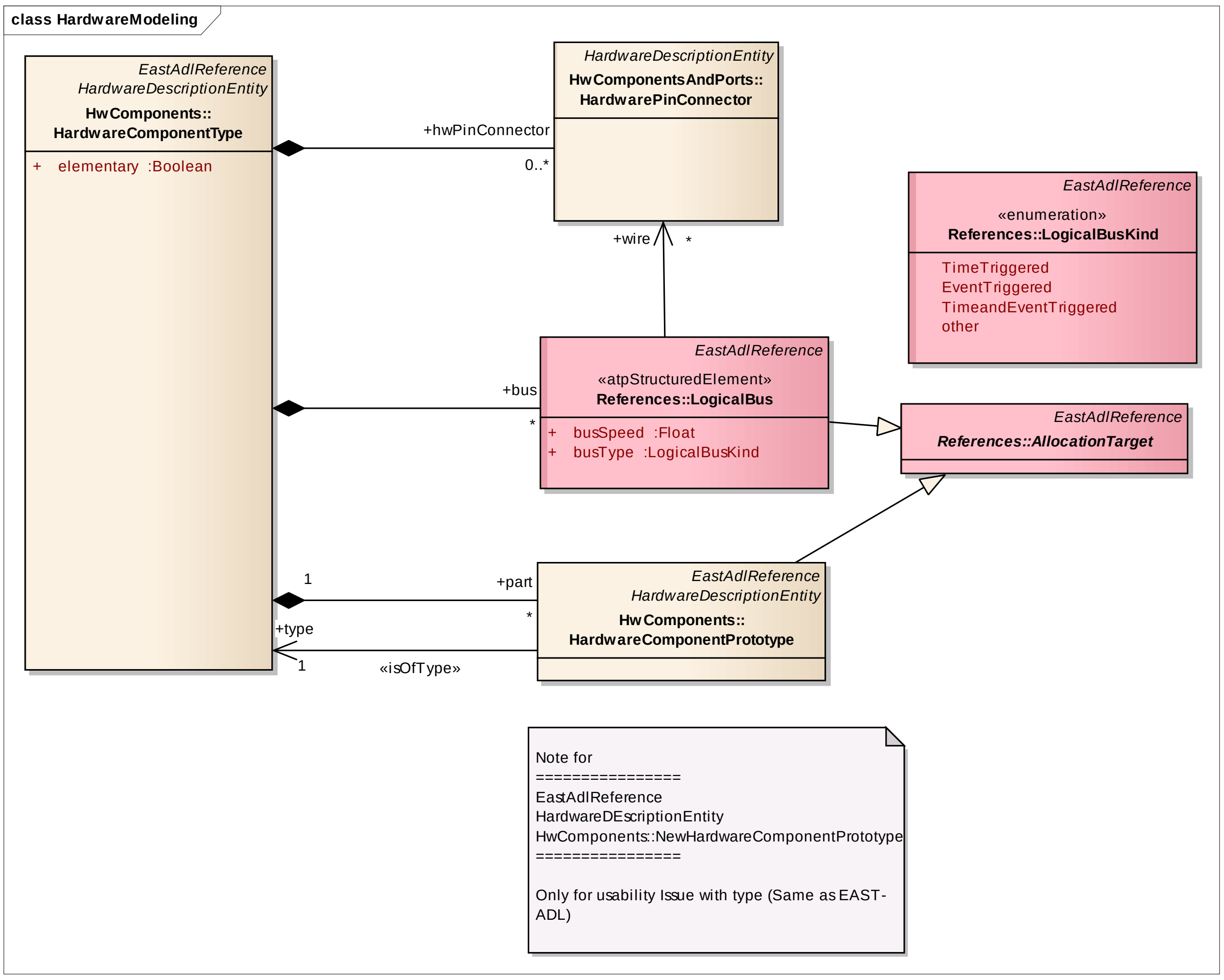


Figure 1: **HardwareModeling** - *(Class diagram)*

This diagram shows an overview of the basic element of *HardwareModeling* as *HardwareComponentType* and *HardwareComponentPrototype*. It depicts the conservation of *LogicalBus* for backward compatibility of EAST-ADL. It is now proposed to be replaced by a more flexible concept the *HardwarePort* as shown in section 9.2.1.4.

9.2.1.1 Package HwCategory

This package represents the *HwCategory*, similar use as in AUTOSAR, to allow definition of specific attributes to all hardware entities of the Hardware Structure package.

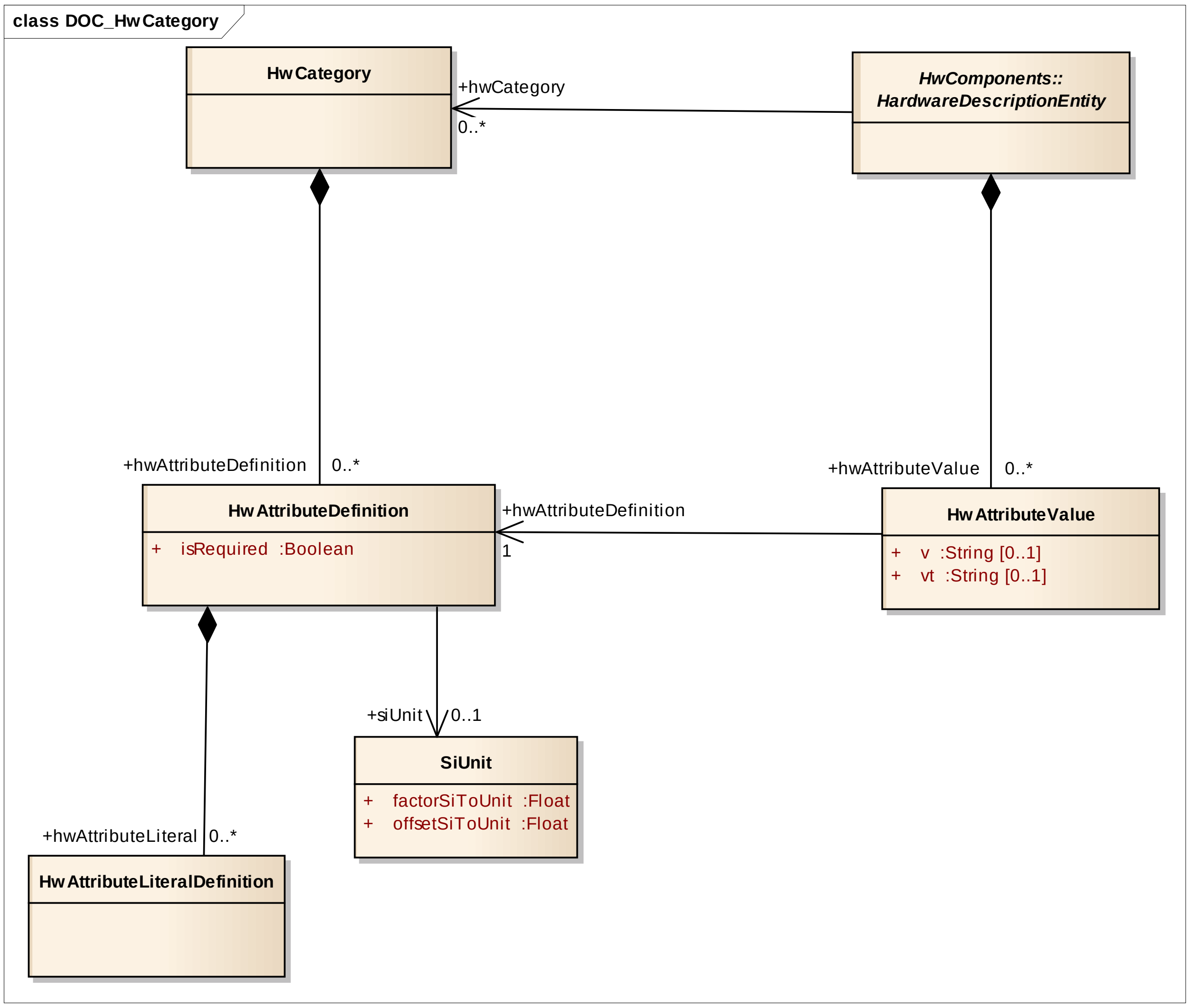


Figure 2: **DOC_HwCategory** - *(Class diagram)*

This class diagram represents a flexible definition of attributes, attached to any hardware entity of the Hardware Structure package, using meta-class generalization *HardwareDescriptionEntity*. This modeling style is the same as the one in use in AUTOSAR to facilitate reuse, refinement and linkage of element between EAST-ADL and AUTOSAR.

The *HwCategory* class is composing of one or several *HwAttributeDefinition* representing the ability to define a particular hardware attribute. This Category can be associated to any *HardwareDescriptionEntity*, in particular to *HardwarePin* to define electrical characteristics, to *HardwarePort* to define communication parameter (e.g. speeds...), to *HardwarePinConnector* to define electrical feature (e.g. resistance) or to *HardwarePortConnector* (e.g. bandwidth or any limitation). The category of this element defines the type of the attribute value. If the category defined by *HwAttributeValue* is Enumeration the *hwAttributeEnumerationLiterals* specify the available literals. This *HwAttributeLiteralDefinition* play the role of *HwAttributeLiteral* for *HwAttributeDefinition* as the definition of the Enumeration. It is only applicable if the category of the *HwAttributeDefinition* equals Enumeration.

The *HwAttributeValue* class represents the ability to assign a hardware attribute value. Note that v and vt are mutually exclusive. The *SiUnit* class represents the physical measurement unit. All units that might be defined should stem from SI units. In order to convert one unit into another factor and offset are defined. For the calculation from SI-unit to the defined unit the factor (*factorSiToUnit*) and the offset (offsetSiToUnit) are applied: Unit = siUnit * factorSiToUnit + offsetSiToUnit. For the calculation from a unit to SI-unit the reciprocal of the factor (factorSiToUnit) and the negation of the offset (offsetSiToUnit) are applied: SiUnit = (unit - offsetSiToUnit) / factorSiToUnit

9.2.1.2 Package HwComponentBehavior

This package describes the behavior of a hardware component. The proposed adaptation of the *HardwareComponentType* is now the representation of the physical entity of the embedded hardware electrical/electronic component including a hardware behavior. This behavior can be defined by language used during hardware architecture development as SystemC, Modelica, VHDL-AMS or Verilog-AMS.

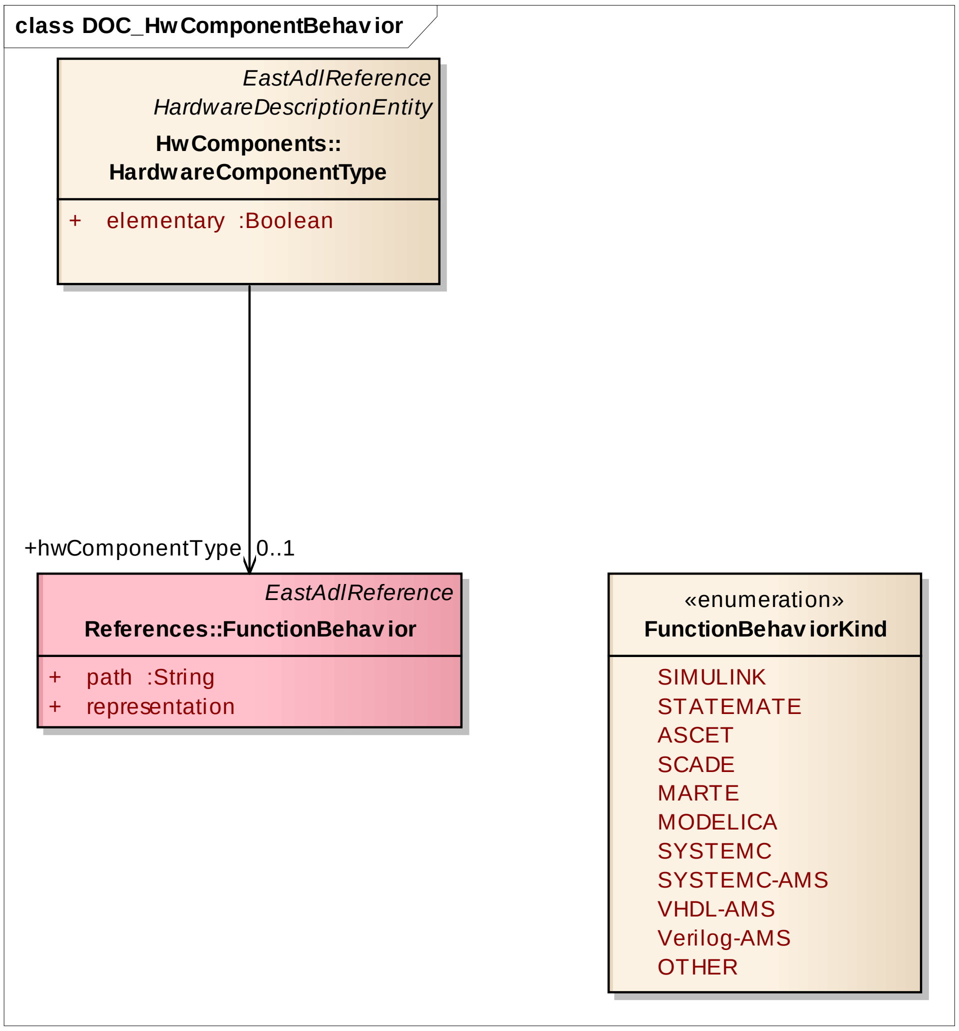


Figure 3: **DOC_HwComponentBehavior** - *(Class diagram)*

This diagram shows the relation of *HardwareComponentType* with a *FunctionBehavior* to map the behavior of the hardware to a function. Each *HardwareComponentType* can reference its behavior via the *FunctionBehavio*r, owning attributes to define a *path* as string path of the file and *FunctionBehaviorKind* is an enumeration which lists the various representations. Hardware modeling languages are added to represent the change on behavior attached *HardwareComponentType*. Several representations are listed; however, one can always extend this list by using the literal OTHER.

9.2.1.3 Package HwComponents

This package represents the description of the *HardwareComponentType* and its specializations for precise use, and a compositional approach for hardware component.

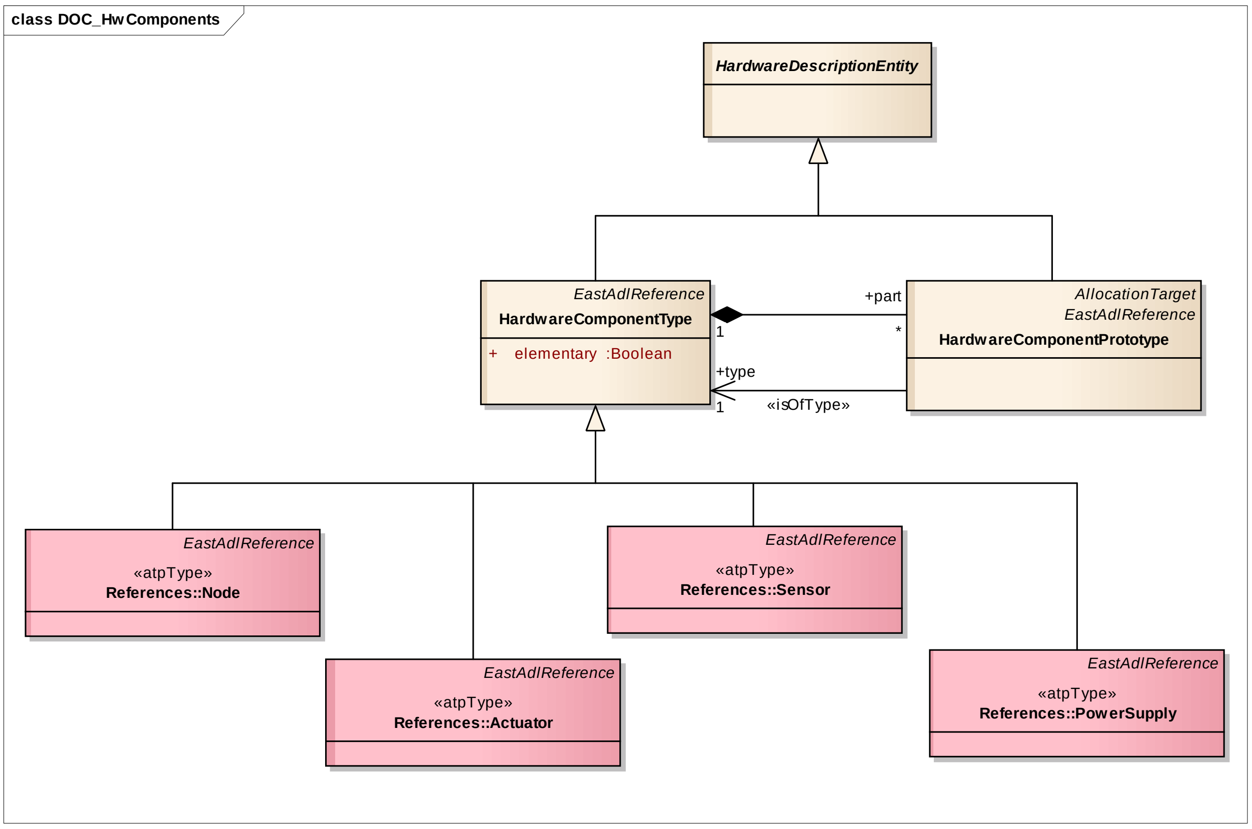


Figure 4: **DOC_HwComponents** - *(Class diagram)*

This class diagram represents the definition of hardware component and its composition thanks to *HardwareComponentType* and *HardwareComponentPrototype*. In addition it includes the list of the class specialized for the use at design level of the hardware component. The *HardwareComponentType* represents hardware element on an abstract level, allowing preliminary engineering activities related to hardware. Through its ports or pins it can be connected to electrical sources and sinks. It is typically connected through its ports to the environment model to participate in the end-to-end behavioral definition of a function. *HardwareComponentProtoype* and HardwareComponentType are specializations of *HardwareDescriptionEntity* as generic lass for hardware relationship definition in the EAST-ADL meta model. *HardwareComponentPrototype* can be typed by a *HardwareComponentType*, and has a composite relation name part to *HardwareComponentType*. This allows for a reference to the occurrence of a *HardwareComponentType* when it acts as a part. The purpose is to support the definition of hierarchical structures, and to reuse the same type of Hardware at several places. For example, a wheel speed sensor may occur at all four wheels, but it has a single definition.

The *HardwareComponentType* can be specialized to represent specific element of the electric/electronic architecture as a *Node* (e.g. typically an ECU), *a Sensor*, and *Actuator* or a *PowerSupply* element.

9.2.1.4 Package HwComponentsAndPorts

This package describes the interface of the hardware component. Such organization is aimed to define low level electrical signal definition and abstraction concept to communication bus with electrical signal grouping.

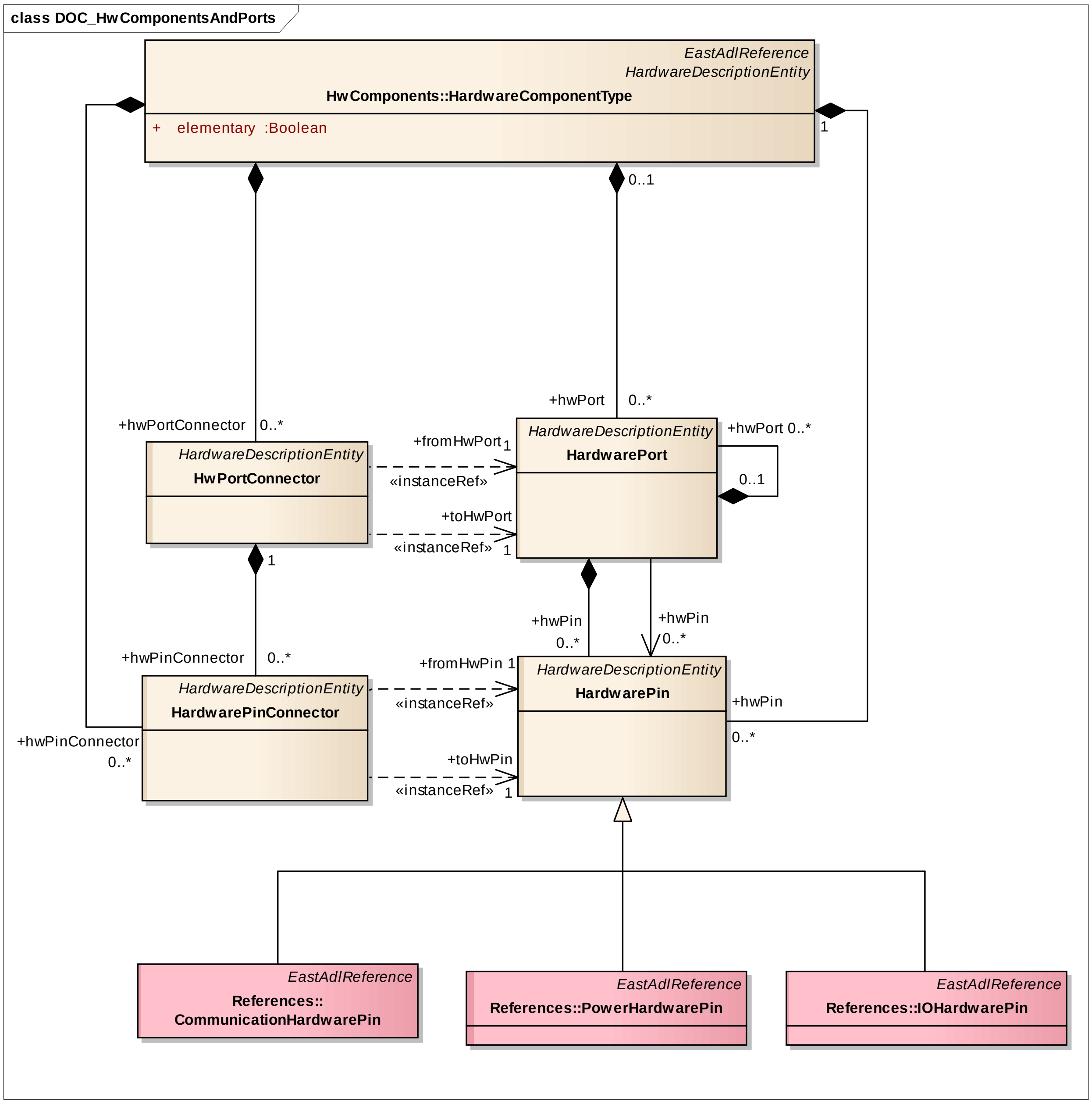


Figure 5: **DOC_HwComponentsAndPorts** - *(Class diagram)*

This class diagram represents the interface of the hardware component made by *HardwarePin* and/or *HardwarePort*. The relation between *HardwarePort* and *HardwarePin* is defined precisely. The *HardwarePort* provides means to organize hardware pins by composing *HwPin* . It can be used to define external/internal communication bus down to the level of communication transactor for hardware bus. It represents a logical connection that carries data from any sender to all receivers. Senders and receivers are identified by the wires of the HardwarePort, i.e. the associated HardwareConnectors. The parameter of *HardwarePort* can be defined with flexible mechanism of HardwareCategory applicable to all hardware entities. Notice that a *HardwarePort* can be also compose *HardwarePort* for larger representation or abstraction (e.g. address/data/control by a simple transaction). Therefore it has two objectives: abstraction of hardware pin(s), and definition of internal/external communication bus; visualization: schematic entry tools busses, like address, data, and control bus. *HardwarePort* can be connected by *HwPortConnector*. *HardwarePortConnector* connectors represent port wires that electrically connect the hardware components through its ports. The connector joins the two referenced ports electrically. *HardwarePin* represents electrical connection points in the hardware architecture. Depending on modeling style, the actual wire or a logical connection can be considered if required. Another use is to compose *HardwarePin* in *HarwdarePort*, for the stake of communication bus interface. Others use case of *HardwaredPin* are the declared specialization as a *CommunicationHardwarePin* (any type of communication busses), a *PowerHardwarePin* (for power supply and ground) or a *IOHardwarePin* (any basic Input output of a component as digital, analog, frequency etc…) .

9.2.1.5 Package HwSwInterface

This package describes the hardware software interface element. Such element shall allow to link unambiguously by a unique element, the hardware component interface with the software element interface.

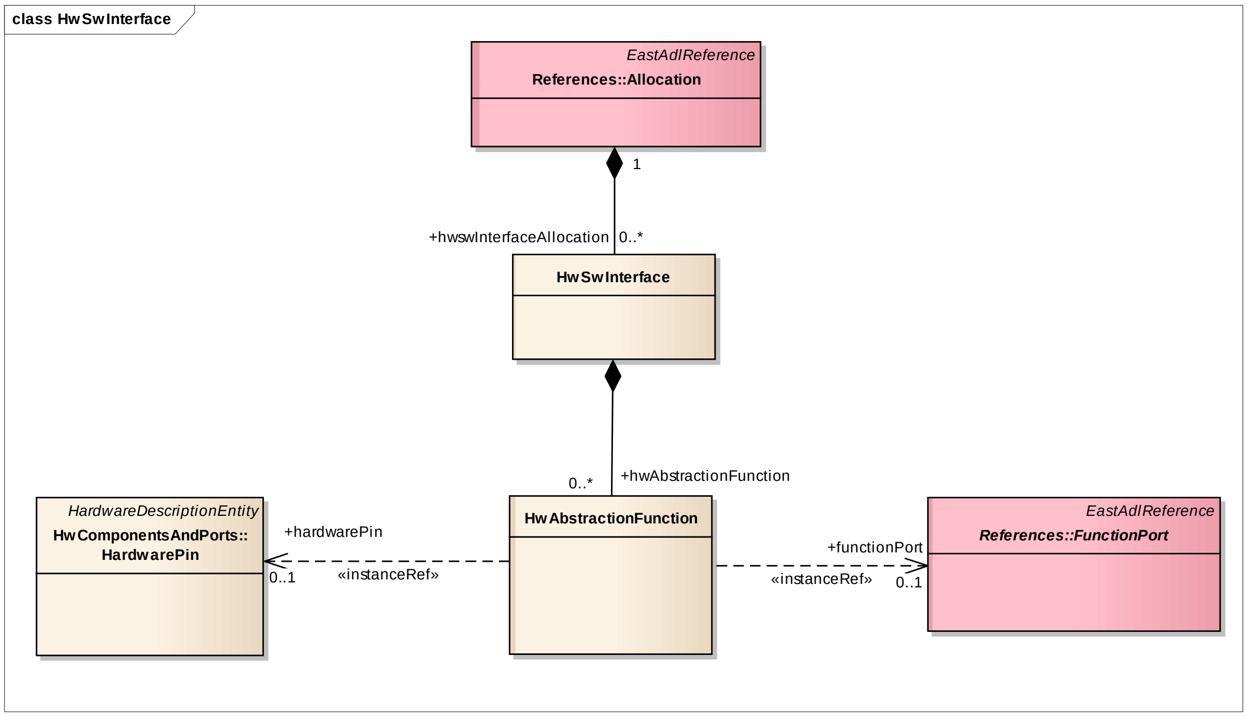


Figure 6: **<u>HwSwInterface</u>** - *(Class diagram)*

This class diagram represents the definition of the *HwSwInterface*. A software element is represented by a *DesignFunction* and a hardware element by a *HardwareComponent*. The *HwSwInterface* class represents the HW-SW interface on the EAST-ADL abstraction Level "Design Level". It is contained into *Allocation* elements that originally bundles all function Allocations, and now bundle the *Hw-SwInterface* elements. *HwSwInterface* is capable to be independent of implementation but allocated into a dedicated hardware element for application purpose (build from *HwSwInterface* abstraction principle). *HwSwInterface* is composed by one or several *HwAbstractionFunction* that allow defining precise interface between hardware and software element of the architecture. As these two elements have heterogeneous interface, as *FunctionPort* and *HardwarePin* as dedicated construct was necessary to represent this inter-relation. It is an abstraction for accessing hardware data by a software element. For software architecture, the abstraction can be defined according to company needs, with our without use of *BasicSoftwareDriverType* for precise definition of interface to the middleware. For hardware architecture, it is can linked to the *upper HardwareComponent* interface as pin , or it could be attached to an internal pin in context of *HardwareComponent* composition (for more precise interface). The *HwAbstractionFunction* has the semantic of execution of the *FunctionPort* where it is linked. This means, once the software *DesignFunction* is executed the immediate out (or in for read) port value propagates to *FunctionPort* and the *HwAbstractionFunction* is executed as an immediate R/W operation of the *HardwarePin*.

9.2.1.6 Package _instanceRef

This package describes the *"instanceRef"* context for the dependency *"instanceRef"* used between modeling elements.

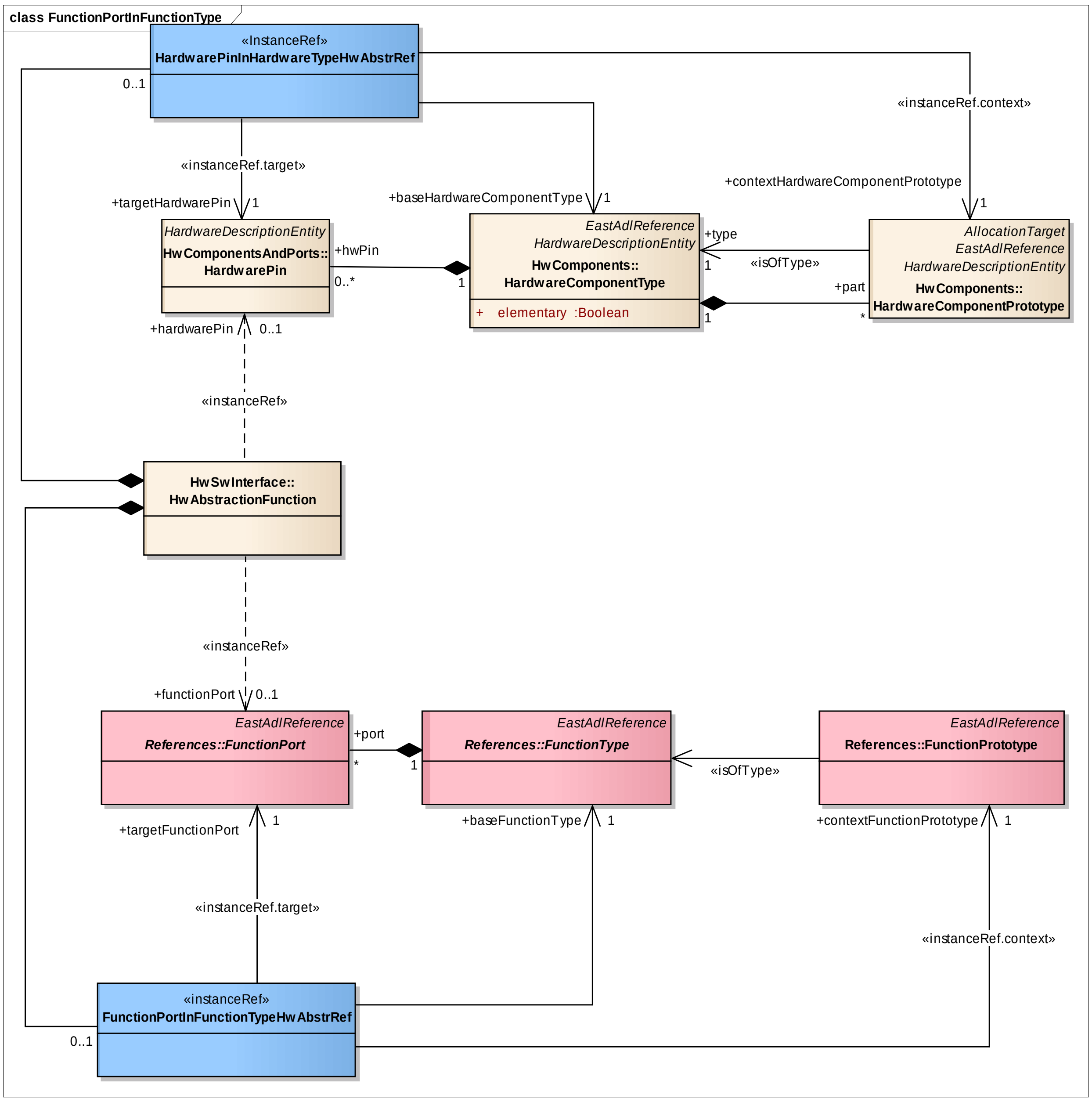


Figure 7: **FunctionPortInFunctionType** - *(Class diagram)*

This class diagram represents the definition of the *instanceRef* target, base and context for *FunctionPort* and *HardwarePin* in the use of *HwAbstractionFunction*. The *HardwarePinInHardwareTypeHwAbstrRef* *"instanceRef"* meta-class is the container for holding the relation of HardwarePin in context of *HardwareComponentPrototype* for the use of *HwAbstractionFunction* (from HwSwInterface). The *FunctionPortInFunctionHwAbstrRef "instanceRef"* meta-class is the container for holding the relation of FunctionPort in context of *Functionprotype* for the use of *HwAbstractionFunction* (from *HwSwInterface*).

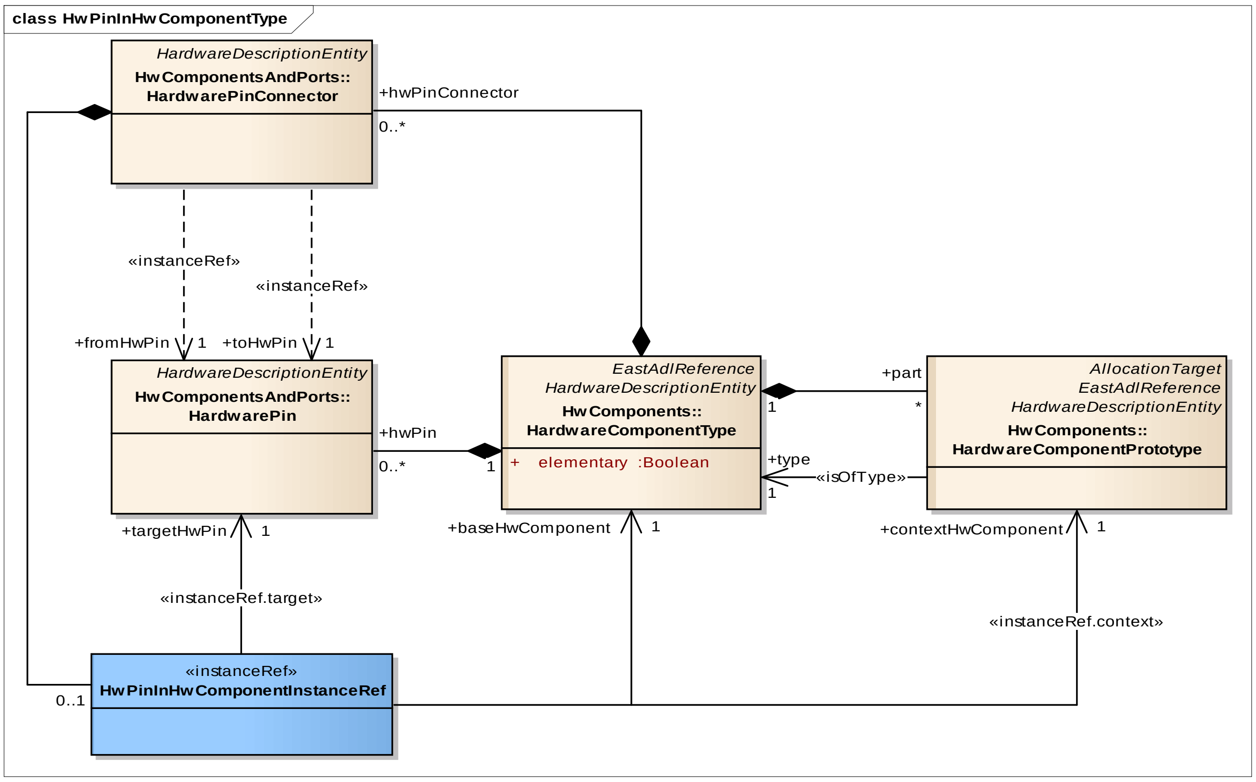

class HwPinInHwComponentType
HardwareDescriptionEntity
HwComponentsAndPorts::HardwarePinConnector
+hwPinConnector
0..*
«instanceRef»
«instanceRef»
+fromHwPin
1
+toHwPin
1
HardwareDescriptionEntity
HwComponentsAndPorts::HardwarePin
+hwPin
0..*
1
EastAdlReference
HardwareDescriptionEntity
HwComponents::HardwareComponentType
+ elementary :Boolean
+part
1
*
+type
«isOfType»
1
AllocationTarget
EastAdlReference
HardwareDescriptionEntity
HwComponents::HardwareComponentPrototype
+targetHwPin
1
+baseHwComponent
1
+contextHwComponent
1
«instanceRef.target»
«instanceRef.context»
«instanceRef»
HwPinInHwComponentInstanceRef
0..1

Figure 8: **HwPinInHwComponentType** - *(Class diagram)*

This class diagram represents the definition of the *instanceRef* target, base and context for *HardwarePin* in the use of *HardwarePinConnector. The HwPinInHwComponentInstanceRef "instanceRef"* meta-class is the container for holding the *relation of HardwarePin* in context of *HardwarePrototype* for the use of *HardwarePinConnector.*

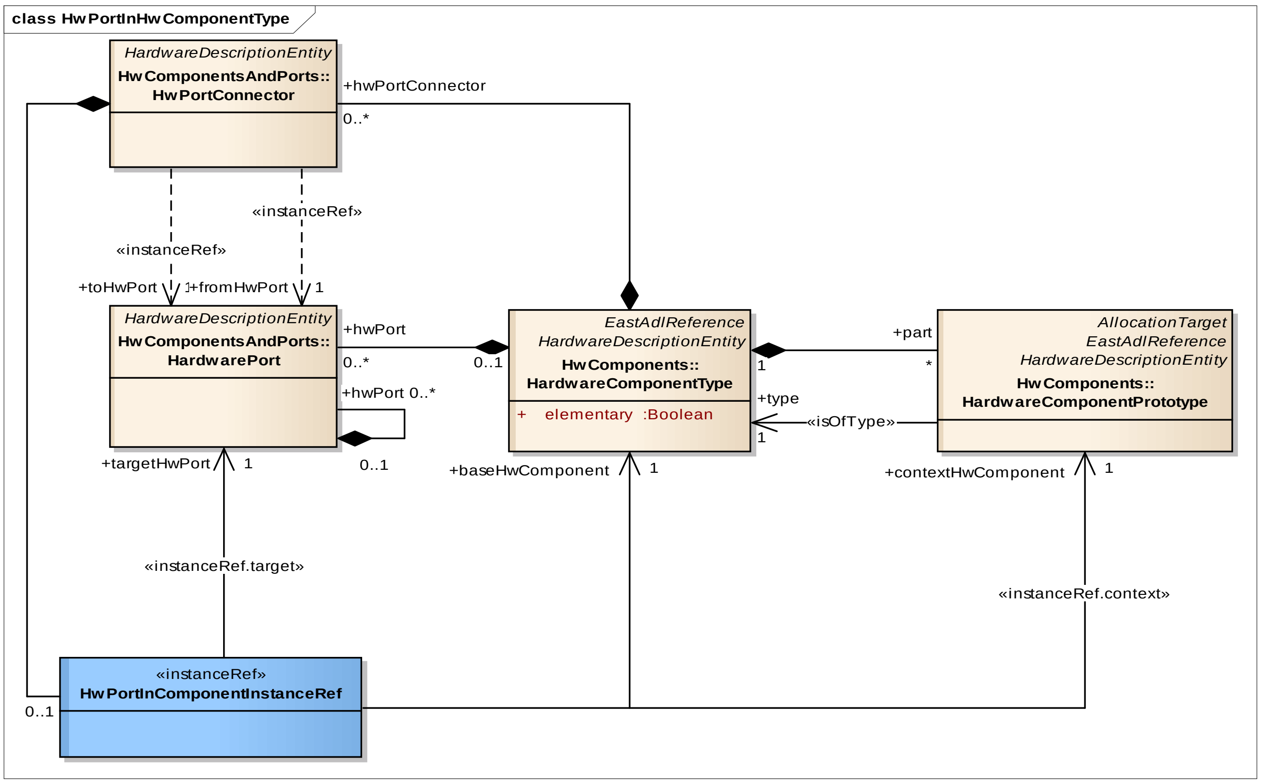

class HwPortInHwComponentType
HardwareDescriptionEntity
HwComponentsAndPorts::HwPortConnector
+hwPortConnector
0..*
«instanceRef»
«instanceRef»
+toHwPort
1
+fromHwPort
1
HardwareDescriptionEntity
HwComponentsAndPorts::HardwarePort
+hwPort
0..*
0..1
+hwPort 0..*
0..1
+targetHwPort
1
EastAdlReference
HardwareDescriptionEntity
HwComponents::HardwareComponentType
+ elementary :Boolean
1
+type
1
«isOfType»
+part
*
AllocationTarget
EastAdlReference
HardwareDescriptionEntity
HwComponents::HardwareComponentPrototype
+baseHwComponent
1
+contextHwComponent
1
«instanceRef.target»
«instanceRef.context»
«instanceRef»
HwPortInComponentInstanceRef
0..1

Figure 9: **HwPortInHwComponentType** - *(Class diagram)*

This class diagram represents the definition of *the instanceRef* target, base and context for *HardwarePort* in the use of *HardwarePortConnector*. This *HwPortInComponentInstanceRef "instanceRef"* meta-class reference is the container for holding the relation of *HardwarePort* in context of *HardwarePrototype* for the use of *HardwarePortConnector*.

9.3 Detailed Description of Classes and Links of Package Hardware

In the following subsections, a detailed description of the classes and links of the WT 3.2.2 - contribution to the SAFE meta-model is given. Name of the top-level package is "Hardware". This on the other hand contains 6 sub-packages, as following

- FailureFormula
- Failure
- FailurePart
- HWQuantitativeMeasure
- HWArchitecturalMetrics
- ProbabilisticMethods

The structural meta model as part of the proposal for adaption of EAST-ADL was described in Section 9.1.

9.3.1 Package FailureFormula

This sub-package contains all equations necessary for the evaluation of the hardware architecture.

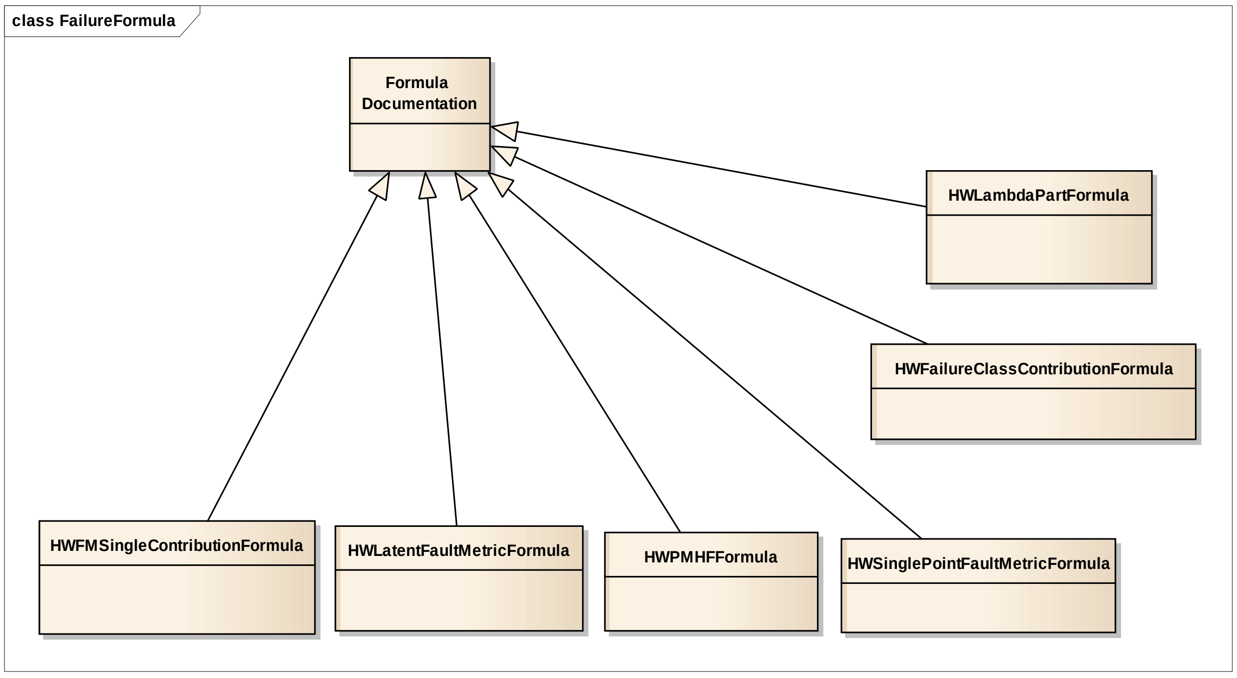

class FailureFormula
Formula Documentation
HWLambdaPartFormula
HWFailureClassContributionFormula
HWFMSingleContributionFormula
HWLatentFaultMetricFormula
HWPMHFFormula
HWSinglePointFaultMetricFormula

Figure 1: **FailureFormula** - *(Class diagram)*

This diagram shows all formula expressions required for the evaluation of the hardware architecture. The *FormulaDocumentat*ion class indicates for class specialization that documentation on formula expressions is documented and can be used for editor implementation (e.g. Java implementation). The contents of the documentation for each formula attached to a specialization are described below in the respective use in the package.

9.3.2 Package Failure

This sub-package describes the failure model of the hardware as derived from the requirements of the ISO 26262.

9.3.2.1 Root Package

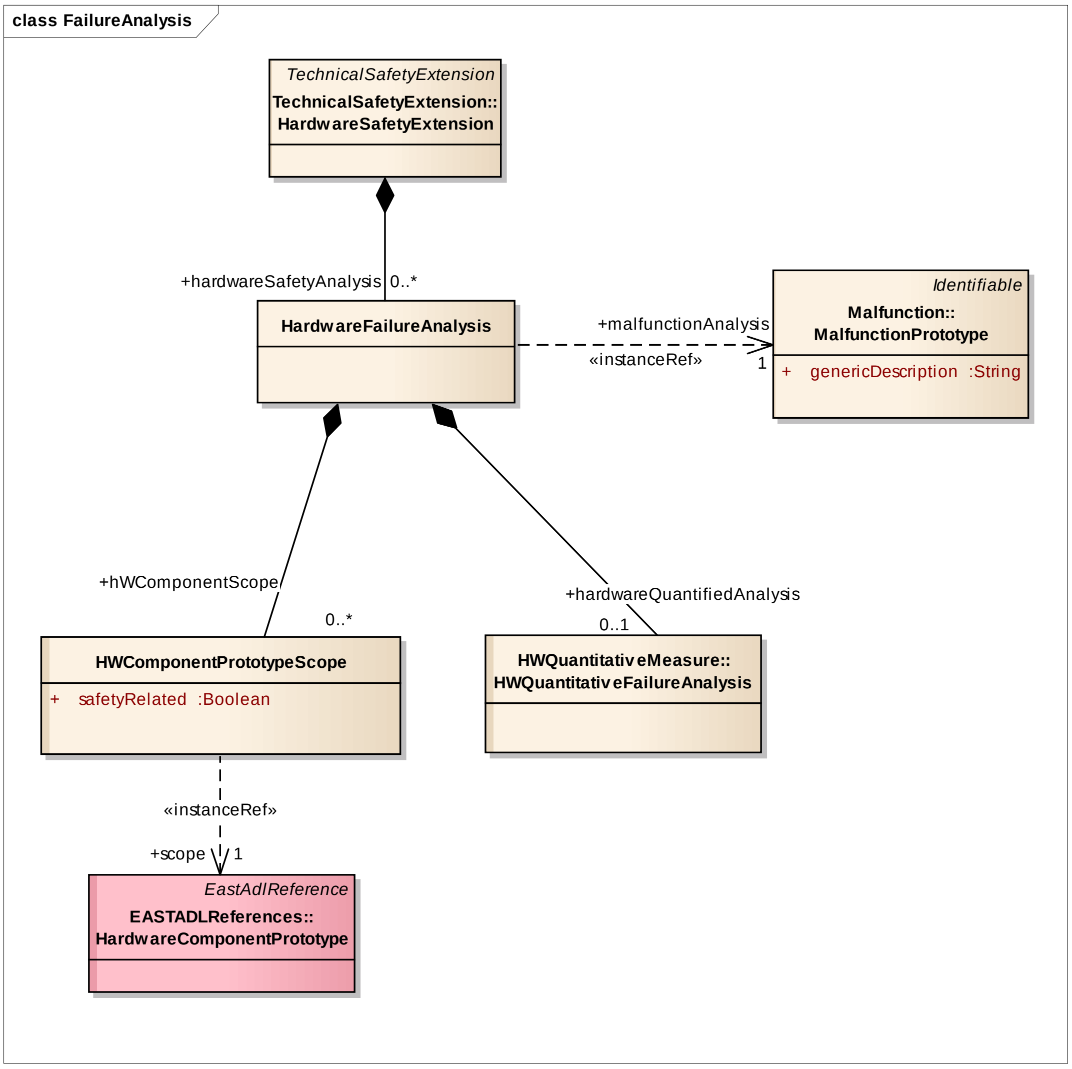


Figure 2: **FailureAnalysis** - *(Class diagram)*

This diagram shows an overview of the hardware component failure extension root information where hardware related failure data and analysis shall be performed.

The *HardwareFailureAnalysis* class represents the container for all Hardware Failure Analysis. One or several *HardwareFailureAnalysis* are aggregated to the *HardwareSafetyExtension*. Each safety goal (as Malfunction), must lead to a safety analysis, so this class contains all the information related to the analysis as: the relation to the malfunction as the *MalfunctionPrototype* for each analysis, the *HwComponentPrototypeScope* to identify all hardware component specific to the context as *HWComponentPrototype* inside a type composition, the *HWQuantifiedFailureAnalysis* to store the results of quantitative analysis performed on the level of the composition.

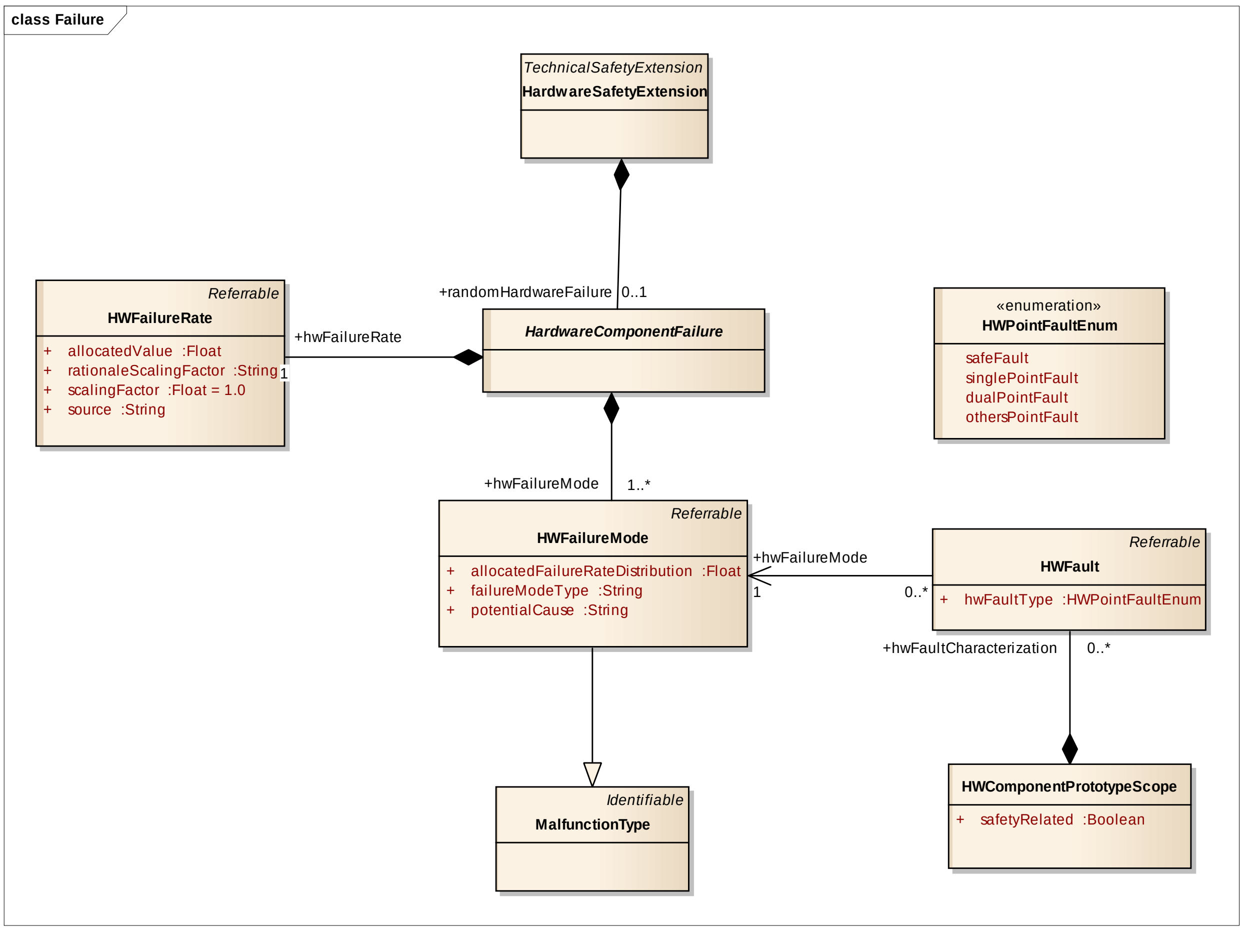


Figure 3: **<u>Failure</u>** - *(Class diagram)*

This diagram shows an overview of the hardware component failure model.

This *HardwareComponentFailure* class describes the *randomHardwareFailure* role for the failure data extension for all *HWComponents* accessible via the generic safety extension mechanism referencing *HardwareSafetyExtension*. The aggregation relation of *HardwareComponentFailure* allows defining failure rate and failure mode at the component level. The class *HWFailureMode* describes the failure mode of a *HWComponent*. The *HWFailureMode* is a specialization of a *MalfunctionType*. It can be traced according Requirement tracing relation from a *TechnicalSafetyRequirement* composed with *QuantifiedDiagnosticCoverageProperty* class identifying the Diagnostic Coverage value for Latent and/or Residual Fault to be able then to compute HW metrics. The attribute *allocatedFailureRateDistribution* of *HWFailureMode* describes the allocated distribution of the failure rate of the specific failure mode (in percentage) of an *HWComponent*. The sum of all failure rate distributions of all failure modes for a single hardware component must lead to the value 100% (may check for consistency). The *failureModeType* attribute describes the type of a failure mode of an *HWComponent* (e.g. "No value" for a sensor). The *potentialCause* attribute allows the documentation of the potential cause of the *HWComponent* failure mode (e.g. high temperature). The *HWFailureMode* can be derived from e.g. Industry Source (see ISO Part 5 8.4.3). The *HWFailureRate* captures the failure rate of an *HWComponent*. Its attribute *allocatedValue* express the FIT rate allocated to this *HWComponent* out of statistics for architectural evaluation and calculation of metrics and probabilistic methods (it shall be expressed in FIT). The *rationaleScalingFactor* attribute provides a rationale, if a scaling factor different to 1.0 is applied. The *scalingFactor* attribute allows potential scaling between different sources of failure rates as described in ISO Part 5 Annex F. *The* source attributes FIT rate source shall documented according to possible source as described in ISO 26262 Part 5 8.4.3 as a) failure rate from industry source (IEC/TR 62380, IEC 61709 ...) or b) statistic based on return field or test, or c) Expert judgment. The appropriate HWFailureRate can be derived from e.g. Industry Source (see ISO Part 5 8.4.3) as an allocated value or calculated via analysis.

The class *HWFault* represents the classification of an *HWComponen*t fault defined as result analysis as *SafeFault*, *SinglePointFault or MultiplePointFault* for a specific *FailureMode* in a context of an *HardwareFailureAnalysis* for an *HardwareArchitecture*. *HWFault* can only exist for *HardwareComponentPrototype* when *HWComponent* are used given by its aggregation to *HWComponentPrototypeScope*. The attribute *hwFaultType* stores the classification of the *FailureMode* for related *MalfunctionPrototype* (linked to violation of a Safety Goal). It can be *SafeFault* (no violation of Safety Goal), *SinglePointFault* (as direct violation of the safety goal), *DualPoint-Fault* (violation of Safety Goal in conjunction with another fault as for example a safety mechanism), and *OthersPointFault* (remark: Multiple-point fault for n>2 are considered as safe faults unless shown to be relevant in the technical safety concept (see ISO Part 5 7.4.3.2 Note 1). The *HWFault* holds the association to *HWFailureMode* to allow reference for the classification of the *HWFailureMode*.

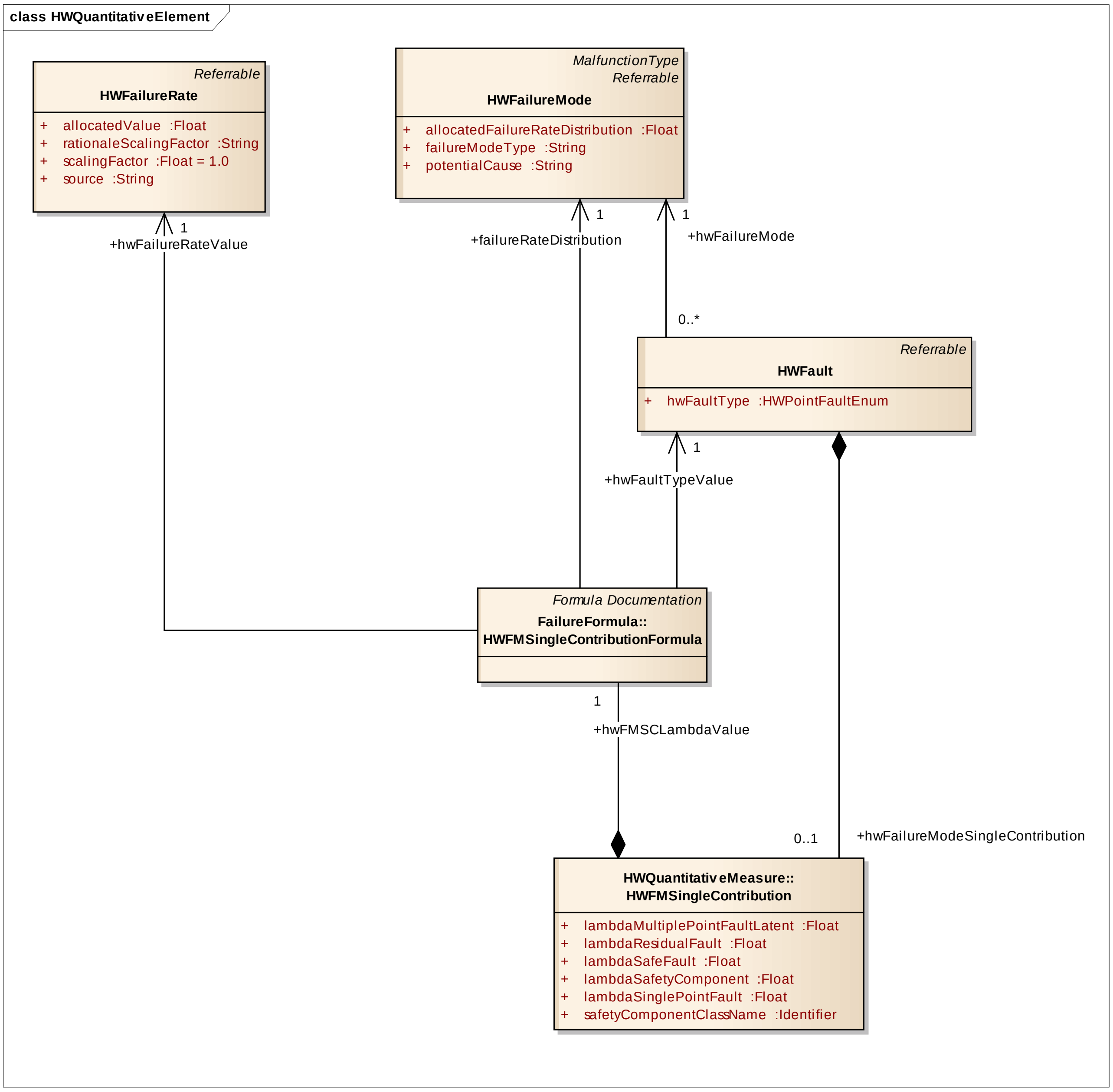

class HWQuantitativeElement
Referrable
HWFailureRate
+ allocatedValue :Float
+ rationaleScalingFactor :String
+ scalingFactor :Float = 1.0
+ source :String
MalfunctionType
Referrable
HWFailureMode
+ allocatedFailureRateDistribution :Float
+ failureModeType :String
+ potentialCause :String
1
+hwFailureRateValue
1
+failureRateDistribution
1
+hwFailureMode
0..*
Referrable
HWFault
+ hwFaultType :HWPointFaultEnum
1
+hwFaultTypeValue
Formula Documentation
FailureFormula::
HWFMSingleContributionFormula
1
+hwFMSCLambdaValue
0..1
+hwFailureModeSingleContribution
HWQuantitativeMeasure::
HWFMSingleContribution
+ lambdaMultiplePointFaultLatent :Float
+ lambdaResidualFault :Float
+ lambdaSafeFault :Float
+ lambdaSafetyComponent :Float
+ lambdaSinglePointFault :Float
+ safetyComponentClassName :Identifier

Figure 4: **HWQuantifiedElement** - *(Class diagram)*

This diagram contains the calculation of the single failure mode contribution of *HWComponent* as preliminary step for the safety evaluation.

The class *HWFMSingleContribution* describes the single contribution in term of failure rate (lambda) to the elementary metrics of the *HWFault* for each failure mode of an *HWComponent*, thanks to the aggegartion relation to *HWFault*. This entity is used to store preliminary calculation for the element used in the context of architectural metrics and probabilistic measurement. This intermediate calculus are stored in its attributes and defined by a *FormulaDocumentation* in the *HWFMSingleContributionFormula* class. The attribute lambdaMultiplePointFaultLatent stores the specific failure rate for single failure mode contribution as multiple-point latent e.g. lambda(MPF,L), the *lambdaSinglePointFault* stores the specific failure rate for single failure mode contribution as single-point fault e.g. lambda(SPF), the *lambdaResidualFault* stores the specific failure rate for single failure mode contribution as residual fault e.g. lambda(RF) and *lambdaSafeFault* stores the specific failure rate for single failure mode contribution as safe fault e.g. lambda(SF). The attribute *lambdaSafetyComponent* attribute stores the sum of specific failure rates for the hardware component for verification and *safetyComponentClassName* the name of the hardware component class for facilitating further consolidation of calculation. The *HWFMSingleContributionFormula* class aggregated to *HWFMSingleContribution* permits its attributes calculation, and holds association to all elements embedded in the formula calculation via respective role *hwFaultTypeValue*, *failureRateDsitribution* and *hwFailureRatevalue*. The formula expression shall be for each *FailureMode* of a safety-related *HwComponent* (part of the item). The formulation is defined as following:

```
lambdaSafetyComponent = Value(HWFailureRate)
SafetyComponentName = HardwareComponent Class name // to allow detect multiple counting of lambdaSafetyComponent
If (HWFault == SafeFault)
        lambdaSafeFault(HWFMSingleContribution) = [Value(HWFailureRate)*failureRateDistribution(HWFailureMode) ]
Else
        lambdaSafeFault(HWFMSingleContribution) = 0
Endif

If (HWFault == SinglePointFault)
        lambdaSinglePointFault(HWFMSingleContribution) = [Value(HWFailureRate)*failureRateDistribution(HWFailureMode) ]
Else
        lambdaSinglePointFault(HWFMSingleContribution) = 0
Endif

If (HWFault == DualPointFault)
   If   (HWSafetyMechanism covers the FailureMode) //residual Fault as HWFailureMode.HWSafetyMechanism != null
        lambdaResidualFault(HWFMSingleContribution) = [Value(HWFailureRate)*failureRateDistribution(HWFailureMode)]* [ 1 - hwDiagnosticCoverageRF(HWSafetyMechanism)/100 ]
        lambdaMultiplePointFaultLatentM(HWFMSingleContribution) = [ Value(HWFailureRate) * failureRateDistribution(HWFailureMode) * hwDiagnosticCoverageRF(HWSafetyMechanism) ] * [ ( 1 - hwDiagnosticCoverageLF(HWSafetyMechanism)/100) ]
   Else  // assume 0% of efficiency for MPL,L metrics (from order 2)
        lambdaResidualFault(HWFMSingleContribution) = 0
        lambdaMultiplePointFaultLatent(HWFMSingleContribution) = [Value(HWFailureRate)*failureRateDistribution(HWFailureMode) ]
  Endif
Endif
```

Notes that *Value(HWFailureRate)* and *failureRateDistribution(HWFailureMode)* are applied on the calculated value extracted from electronic design level to perform the final calculation and verification of the architectural hardware metrics and probabilistic evaluation of violation of the safety goal. The selection between an allocated and calculated value is a tool feature. It allows first a calculation for estimation based on allocation field of failure rate and distribution, and then verification based on *HWComponentQuantifiedFMFromPart* as extract from Failure Part Analysis (see below).

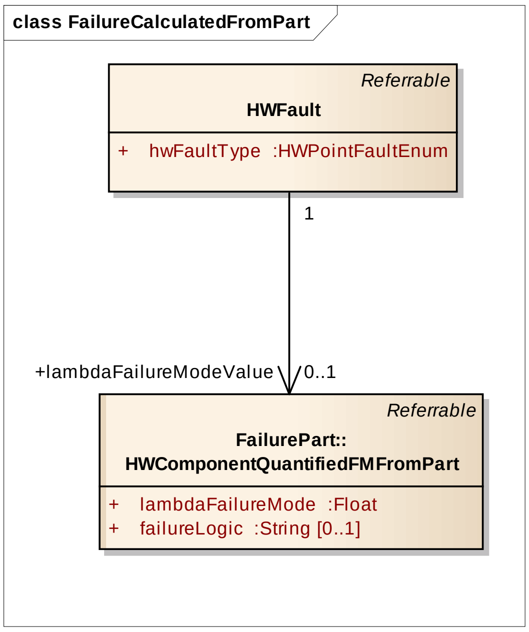

class FailureCalculatedFromPart
Referrable
HWFault
+ hwFaultType :HWPointFaultEnum
1
+lambdaFailureModeValue
0..1
Referrable
FailurePart::
HWComponentQuantifiedFMFromPart
+ lambdaFailureMode :Float
+ failureLogic :String [0..1]

Figure 5: **FailureCalculatedFromPart**- *(Class diagram)*

This diagram shows the association of an *HWFault* of *HWFailureMode* of a hardware component on higher level and its interference with hardware element part and the associated calculations.

This *HWComponentQuantifiedFMFromPart* class describes the quantified failure rate of a *HWFailureMode* of an *HWComponent* based on the contribution of each *HWPartFailureMode* of the related *HWPart* as *AUTOSAR HW Element*. The quantified value is based on the *failureLogic* attribute expressing relationship from *HWPart* to *HWComponent* using a logical expression and calculated by a formula stored in the attribute *lambdaFailureMode*. See in package FailurePart for detailed on calculation.

9.3.2.2 Package _instanceRef

This package describes the *"instanceRef"* context for the dependency *"instanceRef"* used between modeling elements.

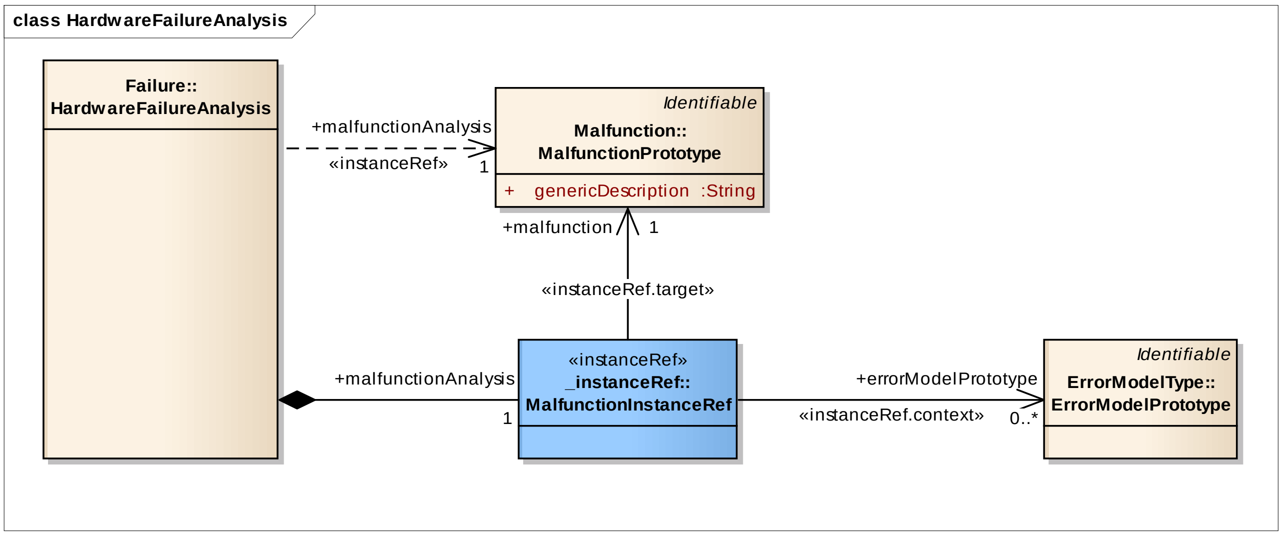


Figure 6: **HardwareFailureAnalysis** - *(Class diagram)*

This class diagram represents the definition of the *instanceRef* target, base and context for a *MalfunctionPrototype* in the use of *HardwareFailureAnalysis. The MalfunctionInstanceRef "instanceRef"* meta-class is the container for holding *the relation to malfunctionAnalysis* of an *HardwareFailureAnalysis* class *for a MalfunctionPrototype* used in context of *ErrorModelPrototype.*

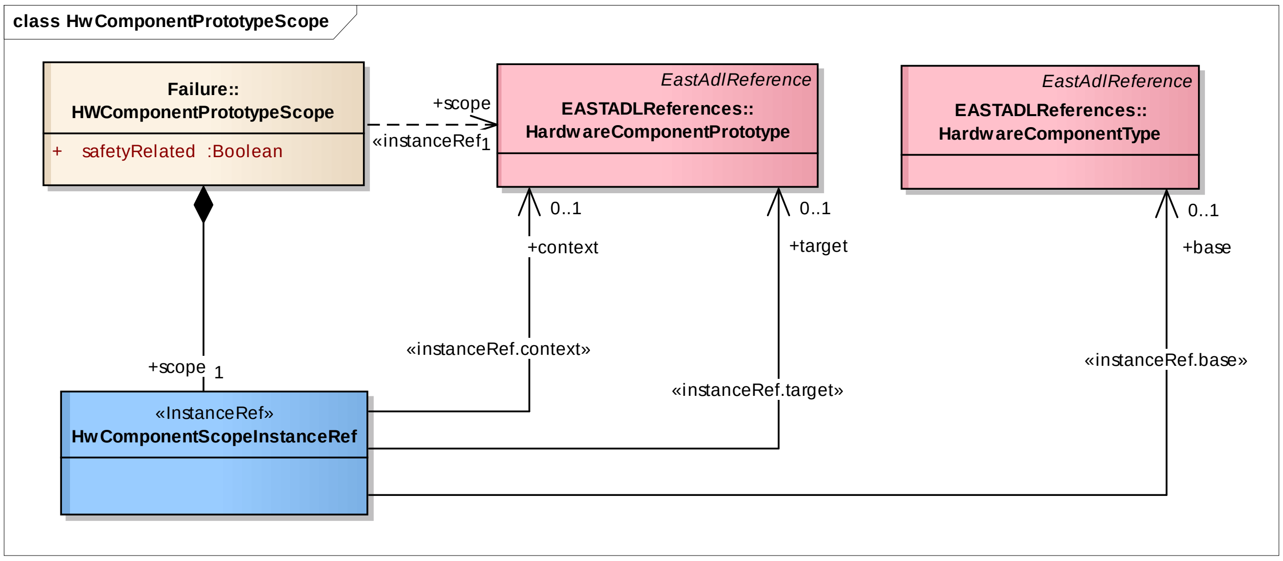


Figure 7: **HwElementprototypeScope** - *(Class diagram)*

This class diagram represents the definition of the *instanceRef* target, base and context for *HWComponentPrototypeScope* in the use of *HwComponent*. The *HwComponentInstanceRef*s *"instanceRef"* meta-class is the container for holding the relation to *HwComponentType* in context of *HwComponentPrototype*.

9.3.3 Package FailurePart

This sub-package describes the failure model of the hardware as derived from the requirements of the ISO 26262.

9.3.3.1 Root Package

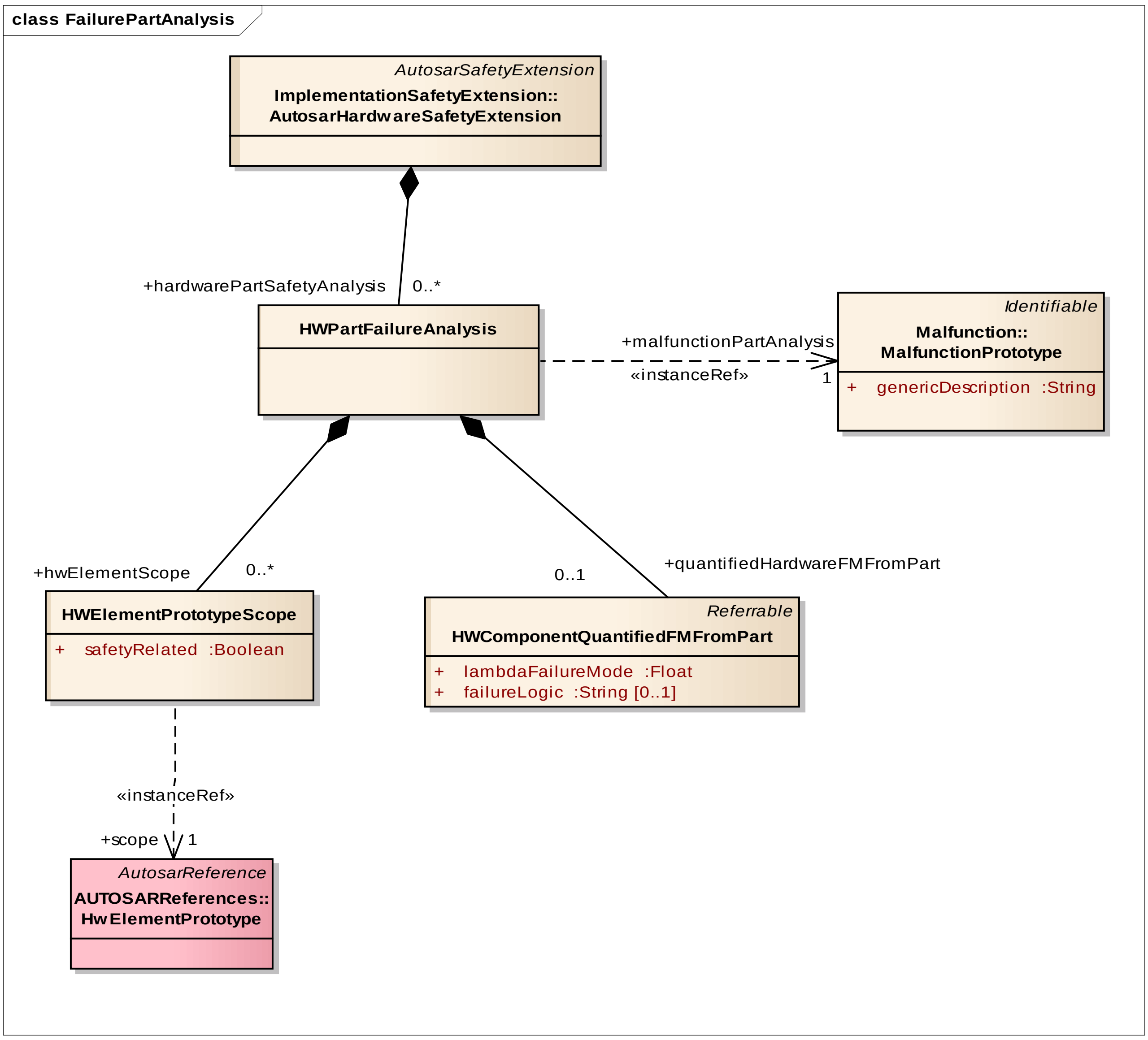


Figure 8: **FailurePartAnalysis** - *(Class diagram)*

This diagram shows an overview of the hardware part failure extension root information where hardware part related failure data and analysis shall be performed.

The *HWPartFailureAnalysis* class represents the container for all Hardware Part Failure Analysis. One or several *HWPartFailureAnalysis* are aggregated to the *AutosarHardwareSafetyExtension*. Each Malfunction must lead to a safety part analysis, so this class contains all the information related to the part analysis as: the relation to the malfunction as the *MalfunctionPrototype* for each part analysis, the

HwElementPrototypeScope to identify all hardware part specific to the context as *HwElementPrototype* inside a type composition, the *HWComponentQuantifiedFMfromPart* to store the results of quantitative analysis performed on the part level and relation to the top level *HWFailureMode* (as malfunction).

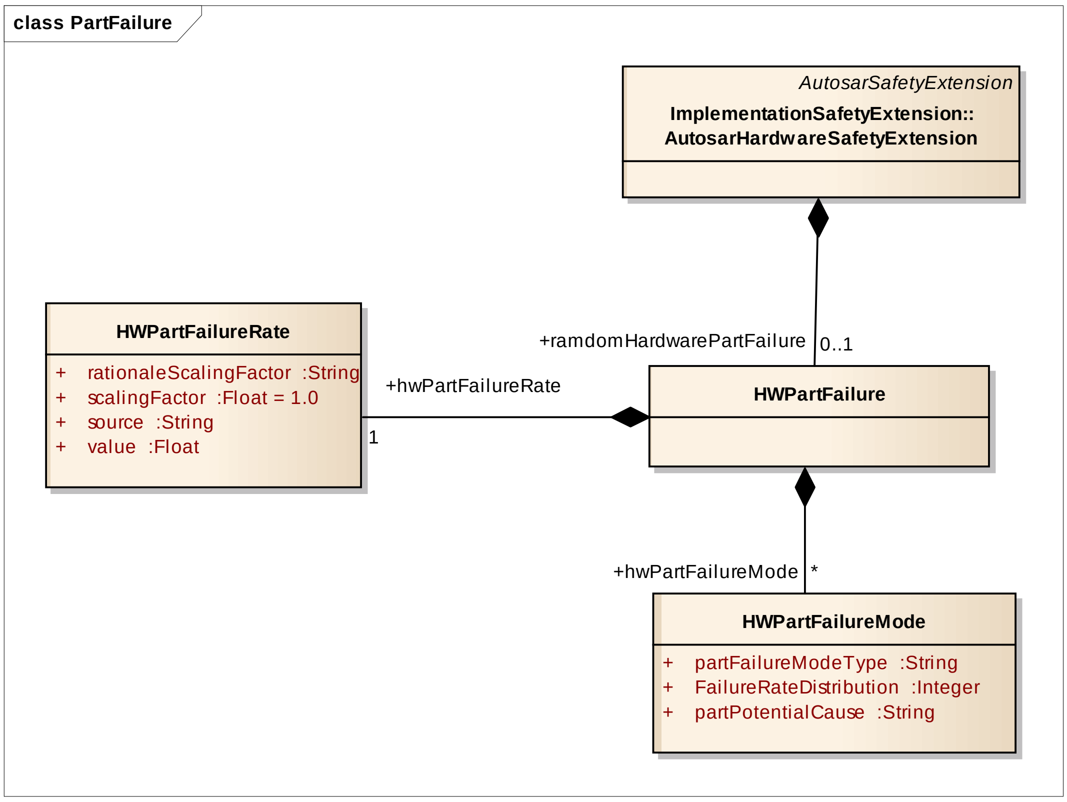

class PartFailure
AutosarSafetyExtension
ImplementationSafetyExtension::
AutosarHardwareSafetyExtension
HWPartFailureRate
+ rationaleScalingFactor :String
+ scalingFactor :Float = 1.0
+ source :String
+ value :Float
+hwPartFailureRate
1
+ramdomHardwarePartFailure
0..1
HWPartFailure
+hwPartFailureMode
*
HWPartFailureMode
+ partFailureModeType :String
+ FailureRateDistribution :Integer
+ partPotentialCause :String

Figure 9: **PartFailures** - *(Class diagram)*

This diagram shows the hardware part failures and its contribution to the hardware component failure on higher level.

The *HWPartFailure* class describes the *randomHardwarePartFailure* role for the failure data extension for all Autosar *HWElement* accessible via the generic safety extension mechanism referencing Autosar*HardwareSafetyExtension*. The aggregation relation of *HWPartFailure* allows defining failure rate and failure mode at the part level (hardware design level). The class *HWPartFailureMode* describes the failure mode of a *HWElement*. The attribute partF*ailureModeType* describes the type of a part failure mode of a *HWElement* (e.g. "Short Circuit to ground" for a resistance). The attribute *FailureRateDistribution* of *HWPartFailureMode* describes the distribution of the failure rate of the specific failure mode (in percentage) of a *HWElement.* The partP*otentialCause* attribute allows the documentation of the potential cause of the *HWElement* failure mode (e.g. high temperature). The *HWPartFailureMode* can be derived from e.g. Industry Source (see ISO Part 5 8.4.3). The *HWpartFailureRate* captures the failure rate of a *HWElement*. Its attribute *value* express the FIT rate allocated to this *HWElement*, it shall be expressed in FIT. The *rationaleScalingFactor* attribute provides a rationale, if a scaling factor different to 1.0 is applied. The *scalingFactor* attribute allows potential scaling between different sources of part failure rates as described in ISO Part 5 Annex F. The *source* attributes FIT rate source shall documented according to possible source as described in ISO 26262 Part 5 8.4.3 as a) failure rate from industry source (IEC/TR 62380, IEC 61709 ...) or b) statistic based on return field or test, or c) Expert judgment. The appropriate *HWPartFailureRate* can be derived from e.g. Industry Source (see ISO Part 5 8.4.3).

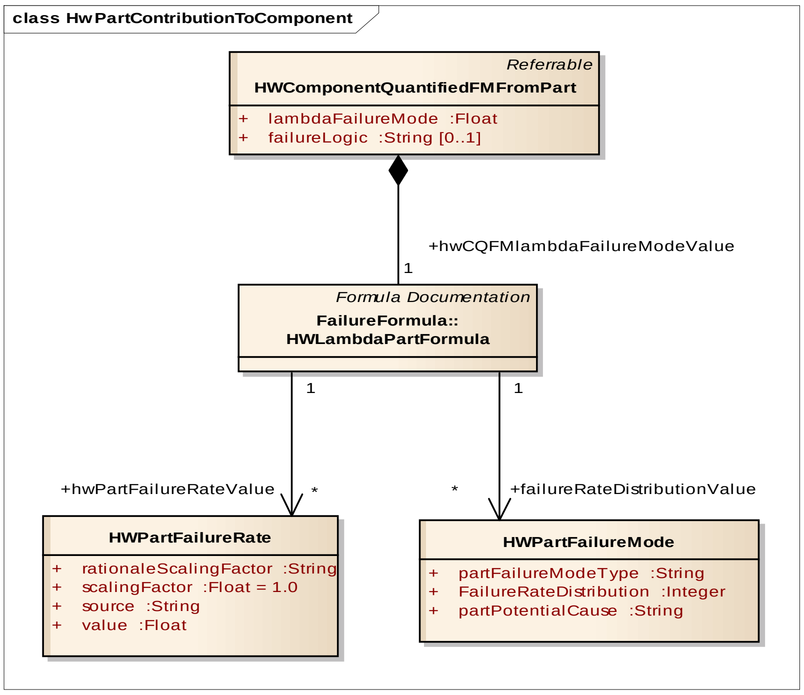

class HwPartContributionToComponent
Referrable
HWComponentQuantifiedFMFromPart
+ lambdaFailureMode :Float
+ failureLogic :String [0..1]
+hwCQFMlambdaFailureModeValue
1
Formula Documentation
FailureFormula::
HWLambdaPartFormula
1
1
+hwPartFailureRateValue
*
*
+failureRateDistributionValue
HWPartFailureRate
+ rationaleScalingFactor :String
+ scalingFactor :Float = 1.0
+ source :String
+ value :Float
HWPartFailureMode
+ partFailureModeType :String
+ FailureRateDistribution :Integer
+ partPotentialCause :String

Figure 9: **HwPartFContributionToComponent** - *(Class diagram)*

This diagram shows the hardware part failures and its contribution to the hardware component failure on higher level.

The *HWComponentQuantifiedFMFromPart* class describes the quantified failure rate of a *HWFailureMode* of an *HWElement* based on the contribution of each *HWPartFailureMode* and *HWPartFailureRate* of the related *HWPart* as *AUTOSAR HW Element*. The *HWLambdaPartFormula* class aggregated to *HWComponentQuantifiedFMFromPart* permits its attributes calculation, and holds association to all elements embedded in the formula calculation via respective role *hwpartFailureRateValue* and *failureRateDistributionValue*. The formula expression shall be for each *FailureMode* of a safety-related *HwComponent* (part of the item). The formulation is defined as following:

// function all represent the failureLogic equation
lambdaFailureMode = function all HWPartFailureMode [Value(HWPartFailureRate) * FailureRateDistribution(HWPartFailureMode), AutosarHWelement)

9.3.3.2 Package _instanceRef

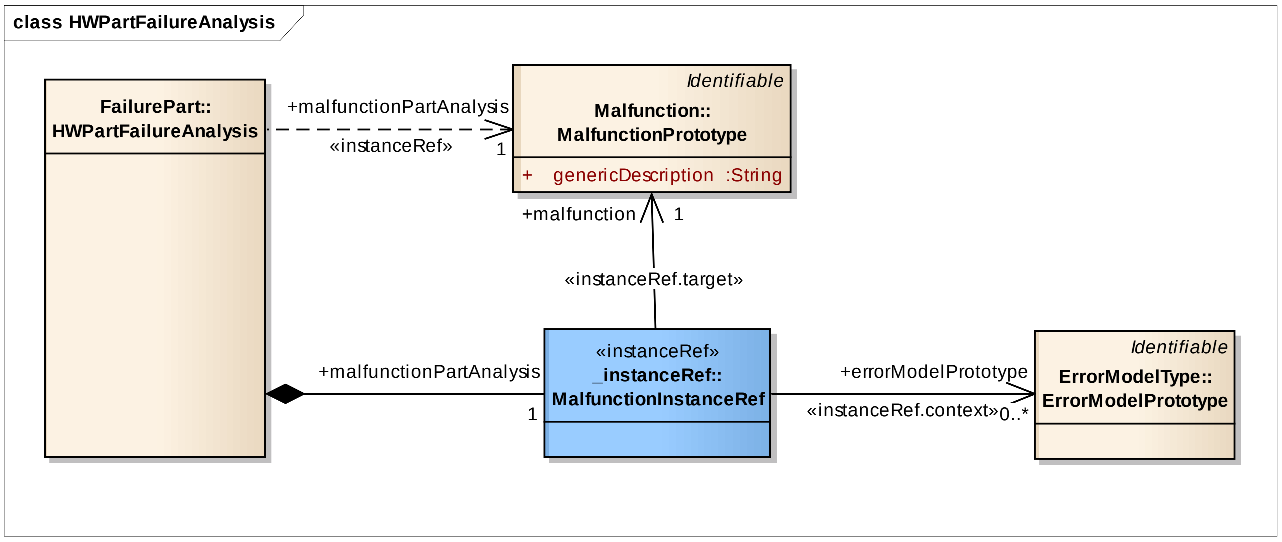


Figure 10: **HWPartFailureAnalysis** - *(Class diagram)*

This class diagram represents the definition of the *instanceRef* target, base and context for a *MalfunctionPrototype* in the use of *HWPartFailureAnalysis. The MalfunctionInstanceRef "instanceRef"* meta-class is the container for holding *the relation to malfunctionPartAnalysis* of an *HWPartFailureAnalysis* class *for a MalfunctionPrototype* used in context of *ErrorModelPrototype.*

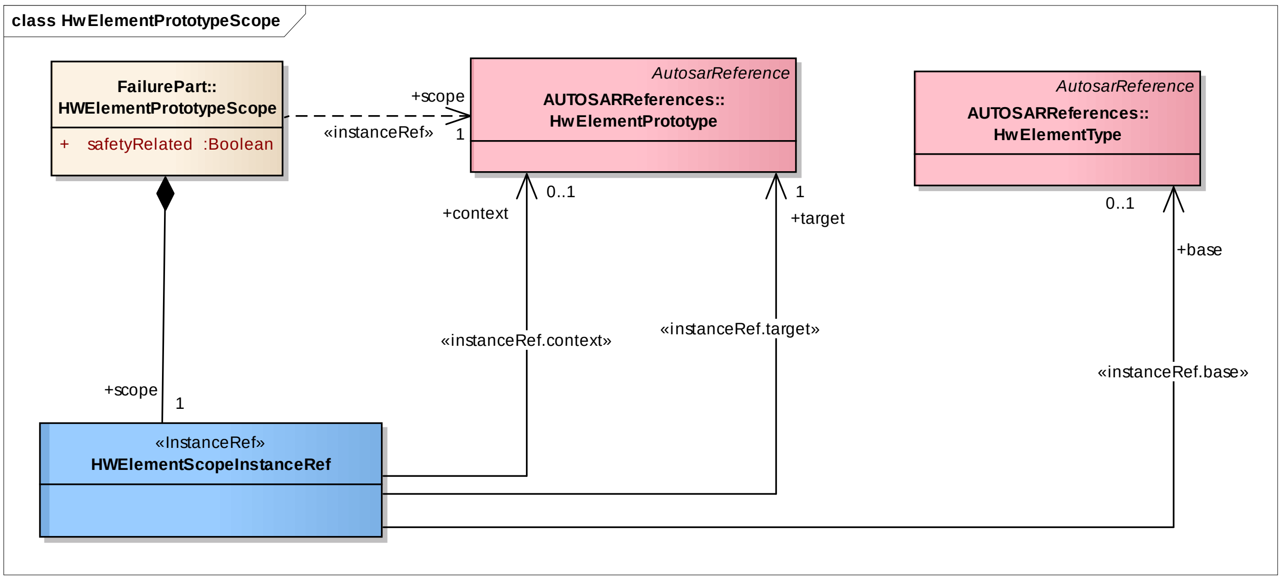


Figure 11: **HwElementprototypeScope** - *(Class diagram)*

This class diagram represents the definition of the *instanceRef* target, base and context for *HWElementPrototypeScope* in the use of *HwElement*. The *HwElementInstanceRef*s *"instanceRef"* meta-class is the container for holding the relation to *HwElementType* in context of *HwElementPrototype*.

9.3.4 Package HWQuantitativeMeasure

This sub-package contains the storage and classification of the safety evaluation. In addition it includes the single failure mode contribution as basis for the concrete evaluation.

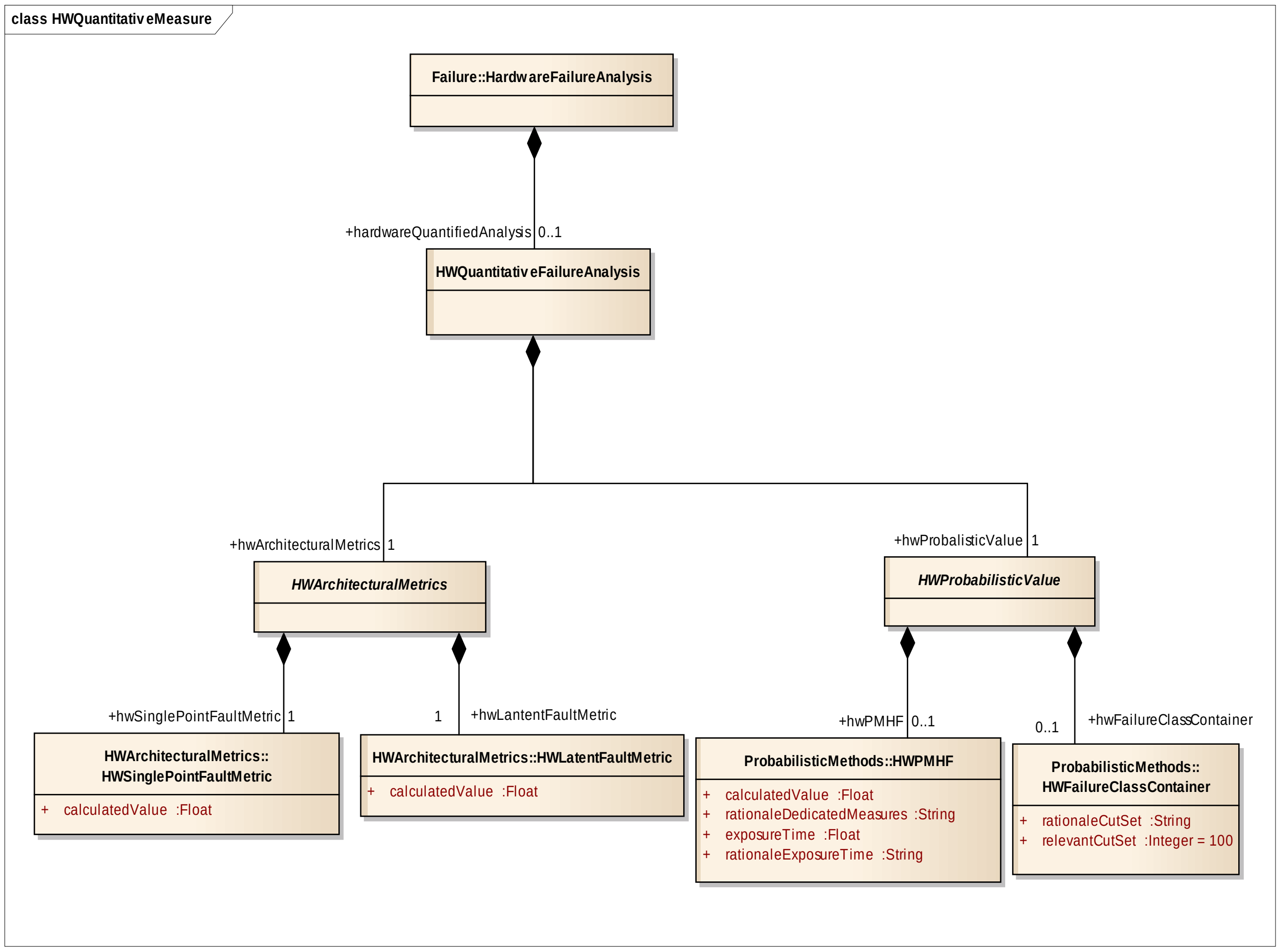

class HWQuantitativeMeasure
Failure::HardwareFailureAnalysis
+hardwareQuantifiedAnalysis 0..1
HWQuantitativeFailureAnalysis
+hwArchitecturalMetrics 1
HWArchitecturalMetrics
+hwProbalisticValue 1
HWProbabilisticValue
+hwSinglePointFaultMetric 1
1 +hwLantentFaultMetric
+hwPMHF 0..1
0..1 +hwFailureClassContainer
HWArchitecturalMetrics:: HWSinglePointFaultMetric
+ calculatedValue :Float
HWArchitecturalMetrics::HWLatentFaultMetric
+ calculatedValue :Float
ProbabilisticMethods::HWPMHF
+ calculatedValue :Float
+ rationaleDedicatedMeasures :String
+ exposureTime :Float
+ rationaleExposureTime :String
ProbabilisticMethods:: HWFailureClassContainer
+ rationaleCutSet :String
+ relevantCutSet :Integer = 100

Figure 12: **HWQuantitativeMeasure** - *(Class diagram)*

This diagram gives an overview about the quantitative analysis claimed by ISO 26262 Part 5 Clause 8 and Clause 9.

The class *HWQuantitativeFailureAnalysis* represents the container for all quantified failure analysis required by the ISO 26262 Part 5 for a dedicated SafetyGoal as specified by aggregation on a *HardwareFailureAnalysis* class. *HWQuantitativeFailureAnalysis* allows clustering all meta class for the hardware architectural metrics *in HWArchitecturalMetrics, as* described in the ISO Part 5 Clause 8 (Single-Point-Fault Metric, Latent-Fault Metric) and for storing in *HWProbabilisticValue the* probabilistic value for violation of safety goal (PMH) or Failure Class Method described in the ISO Part 5 Clause 9. Formally the *HWArchitecturalMetrics* is composed of *HWSinglePointFaultMetric* for the representation of the single-point fault metric, demanded by ISO Part 5 Clause 8. The single-point fault metric describes the robustness of the hardware architecture to cope with single-point and residual faults (also see ISO Part 5 Annex C). It value in % is stored in attribute *calculatedValue*. *HWArchitecturalMetrics* is also composed of *HWSinglelatentFaultMetric* for the representation of the latent fault metric, demanded by ISO Part 5 Clause 8. The latent fault metric describes the robustness of the hardware architecture to cope with multiple-point latent faults (also see ISO Part 5 Annex C). It value in % is stored in attribute *calculatedValue*. The *HWProbabilisticValue* class is aggregating the results of one of the two methods PMHF or Failure Rate

Class. The *HWPMHF* class describes the Probabilistic Metric for random Hardware Failures (PMHF) as in ISO Part 5 clause 9.4.2. The attribute *calculatedValue* is the result of the calculation of the PMHF (in FIT). The attribute *rationaleDedicatedMeasures* shall allow defining a rationale for applied dedicated measures in the design. The *exposurTtime* attribute is the duration of exposure used in the simplified computation of the PMH. It shall be expressed in h. The attribute *rationaleExposureTime* is for Documentation of rationale for Exposure Time. The HWfailureClassContainer is a container to store all HW element failure class results and associated assumptions taken for the saving of the cut-set cut context as recorded in its attributes, as it is defined in ISO Part 5 clause 9.4.3. The attribute *rationaleCutSet* provides a textual rationale for the number of relevant cut-sets and *relevantCutSet* stores the number of relevant cut-set.

9.3.5 Package HWArchitecturalMetrics

This sub-package describes the hardware architectural metrics as claimed by ISO 26262 Part 5 Clause 8. A detailed description of the architectural metrics can be found in ISO 26262 Part 5 Annex C.

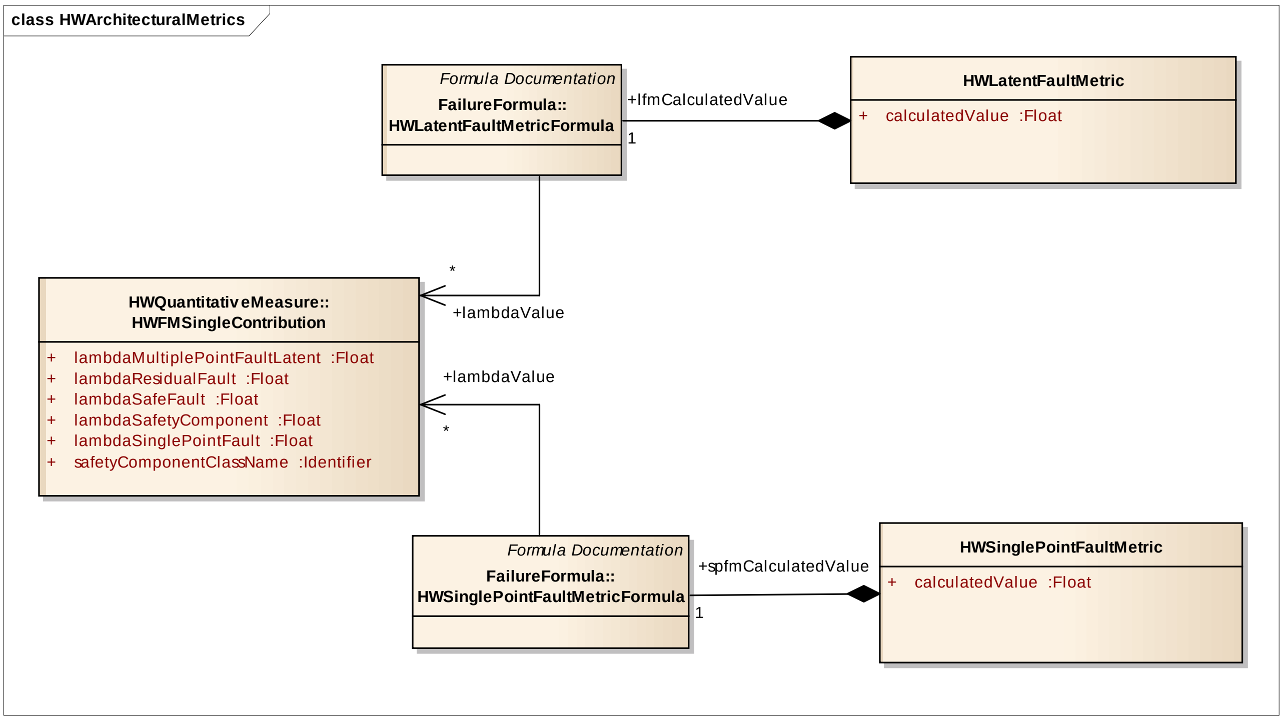


Figure 13: **HWArchitectureMetrics** - *(Class diagram)*

This diagram shows the calculation hardware architectural metrics as described in ISO Part 5-Clause 8 and Annex C.

The *HWSinglePointFaultMetric* class stores the results of the Hardware Single Point Fualt metric based on the contribution of each *HWFMSingleContibution* provided by the *HWSinglePointFaultMetricFormula* documentation class. The generic formula is defined by SPF metric = 100% - total (single point faults failure rate + residual faults failure rate) / total (safety related HWComponent failure rate). In the context of modeling the formula is defined as following:

Value(SinglePointFaultMetric) = { 1 - [(Sum (lambdaSinglePointFault(FMSingleContribution) + lambdaResidualFault(FMSingleContribution)) / Sum(LambdaSafetyComponent)] } * 100
// Sum(LambdaSafetyComponent) is only counted once for a HWElement (identical safetyComponentClassName).

Notes that V*alue(SinglePointFaultMetric*) is applied on estimated value from electronic design level for final calculation and verification of the final single-point fault metric. The selection between calculated and estimated value is a tool feature that allow first a calculation for estimation based on allocation field of failure rate and distribution. Only safety-related *HWComponent* are considered.

The *HWLatentPointFaultMetric* class stores the results of the Hardware Multiple Latent Fault metric based on the contribution of each *HWFMSingleContibution* provided by the *HWLatentFaultMetricFormula* documentation class. The generic formula is defined by MPF,Latent metric = 100% - total (multiple-point faults latent failure rate) /(total (safety-related HWComponent failure rate) - total (single-point faults failure rate + residual faults failure rate)). In the context of modeling the formula is defined as following:

Value(MultipleLatentFaultMetric) = { 1 - [Sum (lambdaMultipleFaultLatent(FMSingleContribution) / [Sum(LambdaSafetyComponent) - Sum (lambdaSinglePointFault(FMSingleContribution) + lambdaResidualFault(FMSingleContribution)]] } * 100
Sum(LambdaSafetyComponent) is only counted once for a HWElement (identical safetyComponentClassName).

Notes that *Value(MutiplePointFaultMteric)* is applied on estimated value from electronic design level for final calculation and verification of the final latent fault metric. The selection between calculated and estimated value is a tool feature that allow first a calculation for estimation based on allocation field of failure rate and distribution. Only safety-related *HWComponent* are considered.

9.3.6 Package ProbabilisticMethods

This sub-package describes the residual risk of safety goal violation due to random hardware failures as claimed by ISO 26262 Part 5 Clause 9. This contains the probabilistic metric for random hardware failures (PMHF) and as an alternative the failure rate class method (FRC).

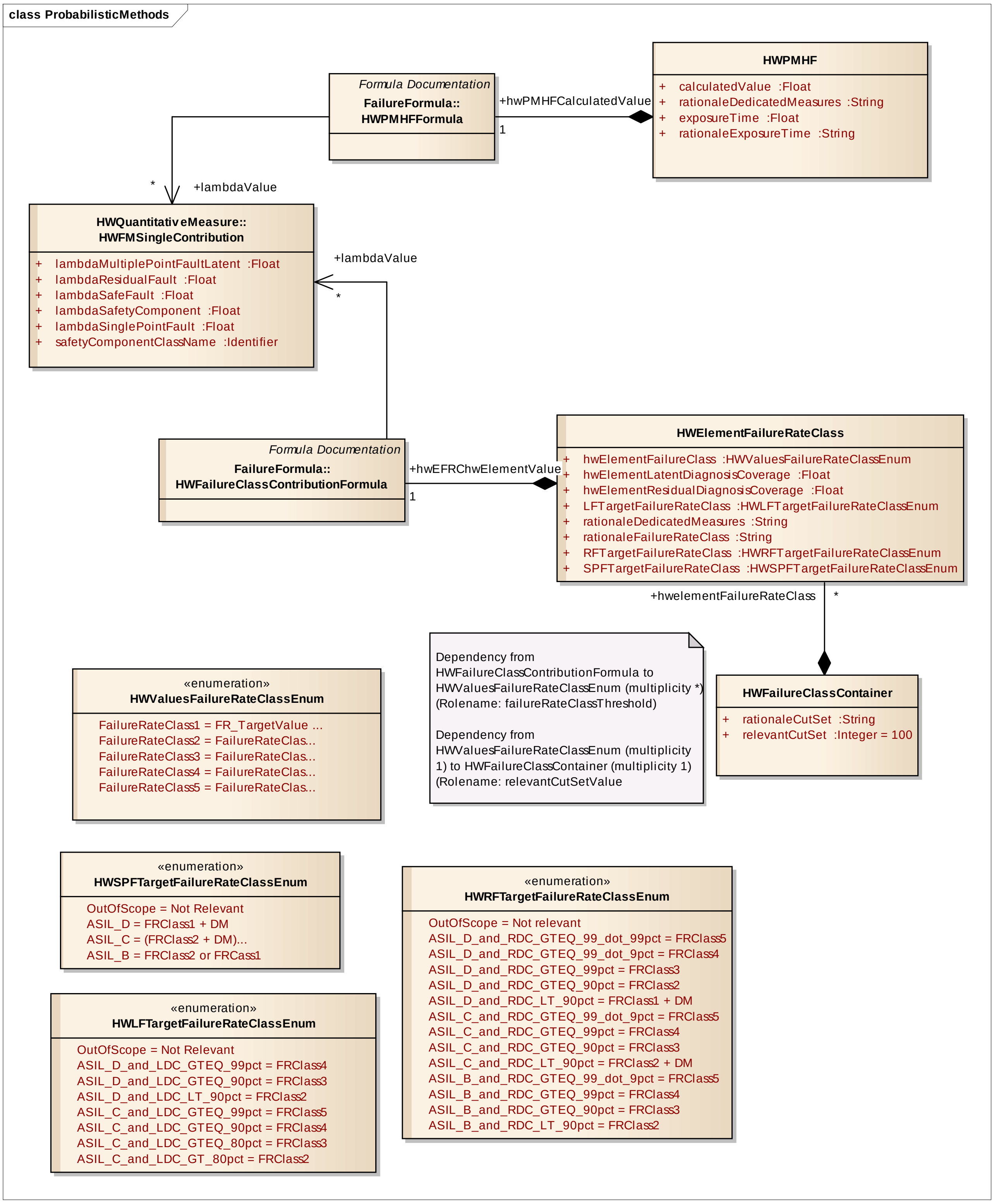

class ProbabilisticMethods
Formula Documentation
FailureFormula::
HWPMHFFormula
+hwPMHFCalculatedValue
1
HWPMHF
+ calculatedValue :Float
+ rationaleDedicatedMeasures :String
+ exposureTime :Float
+ rationaleExposureTime :String
*
+lambdaValue
HWQuantitativeMeasure::
HWFMSingleContribution
+ lambdaMultiplePointFaultLatent :Float
+ lambdaResidualFault :Float
+ lambdaSafeFault :Float
+ lambdaSafetyComponent :Float
+ lambdaSinglePointFault :Float
+ safetyComponentClassName :Identifier
+lambdaValue
*
Formula Documentation
FailureFormula::
HWFailureClassContributionFormula
+hwEFRChwElementValue
1
HWElementFailureRateClass
+ hwElementFailureClass :HWValuesFailureRateClassEnum
+ hwElementLatentDiagnosisCoverage :Float
+ hwElementResidualDiagnosisCoverage :Float
+ LFTargetFailureRateClass :HWLFTargetFailureRateClassEnum
+ rationaleDedicatedMeasures :String
+ rationaleFailureRateClass :String
+ RFTargetFailureRateClass :HWRFTargetFailureRateClassEnum
+ SPFTargetFailureRateClass :HWSPFTargetFailureRateClassEnum
+hwelementFailureRateClass
*
Dependency from HWFailureClassContributionFormula to HWValuesFailureRateClassEnum (multiplicity *) (Rolename: failureRateClassThreshold)
Dependency from HWValuesFailureRateClassEnum (multiplicity 1) to HWFailureClassContainer (multiplicity 1) (Rolename: relevantCutSetValue
HWFailureClassContainer
+ rationaleCutSet :String
+ relevantCutSet :Integer = 100
«enumeration»
HWValuesFailureRateClassEnum
FailureRateClass1 = FR_TargetValue ...
FailureRateClass2 = FailureRateClas...
FailureRateClass3 = FailureRateClas...
FailureRateClass4 = FailureRateClas...
FailureRateClass5 = FailureRateClas...
«enumeration»
HWSPFTargetFailureRateClassEnum
OutOfScope = Not Relevant
ASIL_D = FRClass1 + DM
ASIL_C = (FRClass2 + DM)...
ASIL_B = FRClass2 or FRCass1
«enumeration»
HWRFTargetFailureRateClassEnum
OutOfScope = Not relevant
ASIL_D_and_RDC_GTEQ_99_dot_99pct = FRClass5
ASIL_D_and_RDC_GTEQ_99_dot_9pct = FRClass4
ASIL_D_and_RDC_GTEQ_99pct = FRClass3
ASIL_D_and_RDC_GTEQ_90pct = FRClass2
ASIL_D_and_RDC_LT_90pct = FRClass1 + DM
ASIL_C_and_RDC_GTEQ_99_dot_9pct = FRClass5
ASIL_C_and_RDC_GTEQ_99pct = FRClass4
ASIL_C_and_RDC_GTEQ_90pct = FRClass3
ASIL_C_and_RDC_LT_90pct = FRClass2 + DM
ASIL_B_and_RDC_GTEQ_99_dot_9pct = FRClass5
ASIL_B_and_RDC_GTEQ_99pct = FRClass4
ASIL_B_and_RDC_GTEQ_90pct = FRClass3
ASIL_B_and_RDC_LT_90pct = FRClass2
«enumeration»
HWLFTargetFailureRateClassEnum
OutOfScope = Not Relevant
ASIL_D_and_LDC_GTEQ_99pct = FRClass4
ASIL_D_and_LDC_GTEQ_90pct = FRClass3
ASIL_D_and_LDC_LT_90pct = FRClass2
ASIL_C_and_LDC_GTEQ_99pct = FRClass5
ASIL_C_and_LDC_GTEQ_90pct = FRClass4
ASIL_C_and_LDC_GTEQ_80pct = FRClass3
ASIL_C_and_LDC_GT_80pct = FRClass2

Figure 14: **HWArchitecturalMetrics** - *(Class diagram)*

This diagram contains the evaluation of safety goal violation according to ISO 26262 Part 5 Clause 9. This contains the PMHF and the FRC.

The *HWPMF* class stores the results of Probabilistic Metric for random Hardware Failures (PMHF).based on the contribution of each *HWFMSingleContibution* and using a simplified methods calculation as stored in the *HWPMHFFormula* documentation class. The simplified formula is defined by PMHF = single point faults failure rate + residual faults failure rate + (total safety related faults failure rate / 10-9 * delta) * latent multiple point faults failure rate. In the context of modeling the formula is defined as following:

Value(HWPMHF) = [Sum (lambdaSinglePointFault(HWFMSingleContribution) + lambdaResidualFault(HWFMSingleContribution))] + [Sum(LambdaSafetyComponent) * 1.10-9 * exposureTime(HWPMHF) * lambdaMultiplePointLatent(HWFMSingleContribution)]

// Sum(xxxxValue(xxxxLambdaSafetyComponent) is applied for estimated and calculated, and only counted once (identical safetyComponentClassName).

Notes that *Value(HWPMHF)* is applied on calculated Value extracted from electronic design level for final calculation and verification of the final PMHF probability. The selection between calculated and estimated value is a tool feature that allow first a calculation for estimation based on allocation field of failure rate and distribution. Only Component safety relevant is considered.

The *HWFailureClassContainer* class all individual component evaluation results defined in the set of *HWElementFailureRateClass* using simplified methods for HWComponent FIT rate calculation. *HWElementFailureRateClass* class describes for an *HWComponent*, the Failure Rate Class element to evaluate measure for a malfunction (link to violation of a safety goal) for a single element. This violation is based on failure rate class according to context of evaluation such as ASIL level, list of *HWFault* and diagnostic coverage of the *HWComponent* as HW Element. It allows also storing the target for failure rate class, relevant or not depending of the possible *HWFault* of the failure mode of the *HWComponent* as hardware Element. Furthermore if dedicated measures (DM) are required due to failure class target matching and the necessary information are captured as a textual description. The calculation of the attribute *HWElementFailureClass* and *HWElementDiagnosticCoverage* is derived from the Formula Expression *FMSingleContributionFormula*. The *hwElementFailureRateClass* attributes from *HWElementfailureRateClass* is the failure Rate Class taken from *HWValuesRateClassEnum* based on the failure rate of the hardware component. *FailureRateClass* value corresponds to the maximum value applied in the Failure Rate Class X considering that lower value is Class X-1 (and 0 for class 1). The failure rate class values are determined according to ISO 26262 Part 5 9.4.3.3. Failure Class is based on the number of relevant cut-set. The float *hwElementLatentDiagnosticCoverage* attribute as the diagnostic coverage value with respect to latent faults on hardware element level, calculated with the specific failure rate of all latent multiple-point faults and the overall failure rate of the hardware part element. The float *hwElementResidualDiagnosticCoverage* attribute as the diagnostic coverage value with respect to latent faults on hardware element level, calculated with the specific failure rate of all latent multiple-point faults and the overall failure rate of the hardware part element. The *LFTargetFailureRateClass* attribute as the Target Failure Rate Class for multiple-point latent faults, taken from *HWLFTargetFailureRateClassEnum*. The values of *HWLFTargetFailureRateClassEnum* are taken from ISO 26262 Part 5 9.4.3.11 -Table 9 (Targets of failure rate class and coverage of hardware part regarding dual-point faults). The string *rationaleFailureRateClass* attribute as the rationale for matching criteria on Failure Rate Class. The string *rationaleDedicatedMeasures* attribute providing rationale for dedicated measures, if required. According to ISO 26262 Part 5 9.4.2.4, examples for dedicated measures are a) design features such as hardware part over design (e.g. electrical or thermal stress rating) or physical separation (e.g. spacing of contacts on a printed circuit board); b) a special sample test of incoming material to reduce the risk of occurrence of this failure mode; c) a burn-in test; d) a dedicated control set as part of the control plan; and e) assignment of safety-related special characteristics. The *RFTargetFailureRateClass* attribute as Target Failure Rate Class for residual faults, taken from *HWRFTargetFailureRateClassEnum*. The values of Target Failure Rate Class for residual faults, taken from *HWRFTargetFailureRateClassEnum* are taken from ISO 26262 Part 5 9.4.3.6 -Table 8 (Maximum failure rate classes for a given diagnostic coverage of the hardware part - residual faults). It describes the threshold for Residual Failure according to ASIL level and identifying Failure Class Rate limit (FRClassx) and Dedicated Measure (DM) if necessary. Notice that RDC is addressing the *hwElementResidualDiagnosticCoverage* parameter of the *HWElementFailureRateClass*. The

SPFTargetFailureRateClass attribute as Target Failure Rate Class for single-point faults, taken from *HWSPFTargetFailureRateClassEnum*. The value of *HWSPFTargetFailureRateClassEnum* are taken of ISO 26262 Part 5 9.4.3.5 -Table 7 (Targets of failure rate classes of hardware parts regarding single-point faults).

The *HWElementfailureRateClass* uses a simplified method for *HWComponent* FIT rate calculation based on relation to *HWFMSingleContribution* documented in the *HWFailureClassContributionFormula*. The simplified formula shall be calculated for each *FailureMode* of a safety-related *HwComponent* as
HW Element Failure Rate Class = Failure Class (safety-related failure rate component)
HW Element Residual Diagnostic Coverage = 100% - total (single point faults failure rate + residual faults failure rate) /safety related failure rate component
HW Element Latent Diagnostic Coverage = 100% - total(multiple fault latent) / ((safety related failure rate component) - total (single point faults failure rate + residual faults failure rate))
In the context of modeling the formula is defined as following:

hwElementResidualDiagnosticCoverage
HWElementFailureRateClass(hwElementFailureClass) = HWValuesFailureRateClassEnum(LambdaSafetyComponent)
HWElementFailureRateClass(hwElementResidualDiagnosticCoverage) = { 1 - (Sum (lambdaSinglePointFault(HWFMSingleContribution) + lambdaResidualFault(HWFMSingleContribution)) / LambdaSafetyComponent } * 100
HWElementFailureRateClass(hwElementLatentDiagnosticCoverage) = { 1 - Sum (lambdaMultipleFaultLatent(HWFMSingleContribution) / LambdaSafetyComponent } * 100

Note that *Value(hwElementDiagnosticCoverage)* is applied on estimated Value from electronic design level to perform the final calculation and verification of the individual *HWElement* FailureRateClass and *ElementDiagnosticCoverage.* The selection between calculated and estimated value is a tool feature that allow first a calculation for estimation based on allocation field of failure rate. Only safety-related component are considered and *LambdaSafetyComponent* is only counted once for a *HWElement* (identical safetyComponentClassName).

10 Description Based on an Example

Within this section the hardware modeling concept is described based on an example. ISO26262 Part 5 Annex E [1] describes an example for a valve control. This includes sensors, a microprocessor as control unit, valves as actuators and their interconnection with other elementary hardware components. Two safety goals with their ASIL, safety mechanisms and different hardware components fulfilling functions are described. Figure 13 is given as an electronic schematic used in Annex E.1 to present the example of metrics calculation.

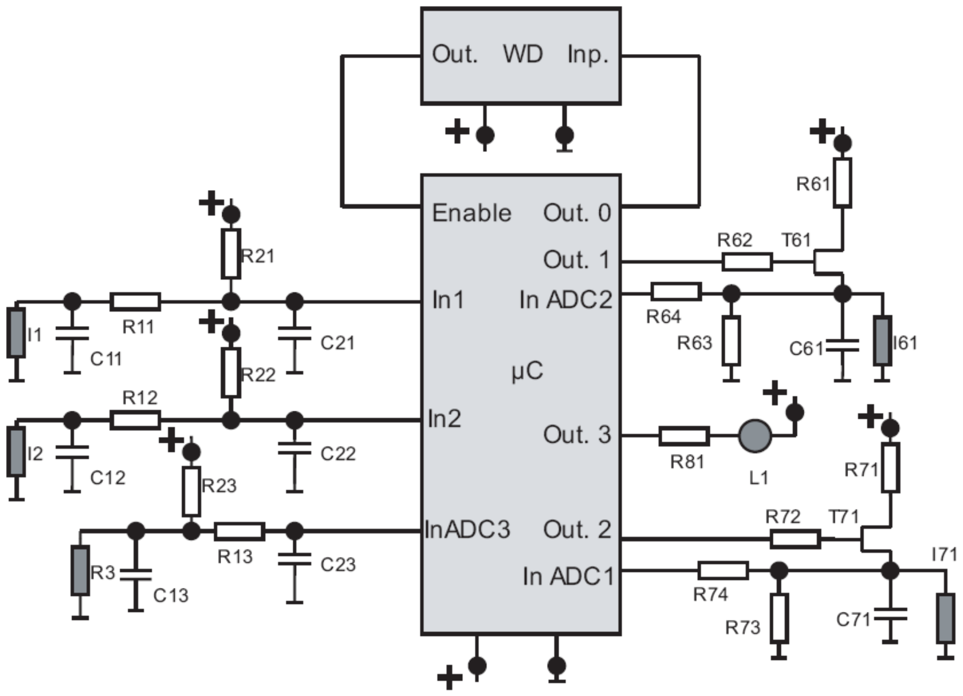


Figure 13: Electronic Schematic diagram ISO26262-Part5 Figure E.1

The representation of the technical safety concept (TSC) of the ISO26262 example shall be extended from the given electronic schematic of Figure 13 in order to apply the proposed modeling methods. The hardware architecture shall be defined by logical component as functional blocks from a top-down development approach. So, the hardware architectural design has been re-engineered to represent HWComponent as represented in Figure 14 . For information, the software elements of the architecture, and in particular software safety mechanism SM2, have been added in red on the microprocessor. Notice that the Hardware Software Interface (HSI) required by a standard TSC has not been added, due to graphical representation.

In this following section, the hardware modeling methods, with dependency to failure propagation from WT 3.3.1 contribution and model-based safety evaluation from WT3.3.3, is described based on this example. Only a brief description is presented, as this example will be studied later in the project thanks to tool and method environments for demonstrating meta models results and methods. In addition, the described engineering steps for the example are reduced to one considered safety goal and limited to calculation of Hardware Architectural Metrics.

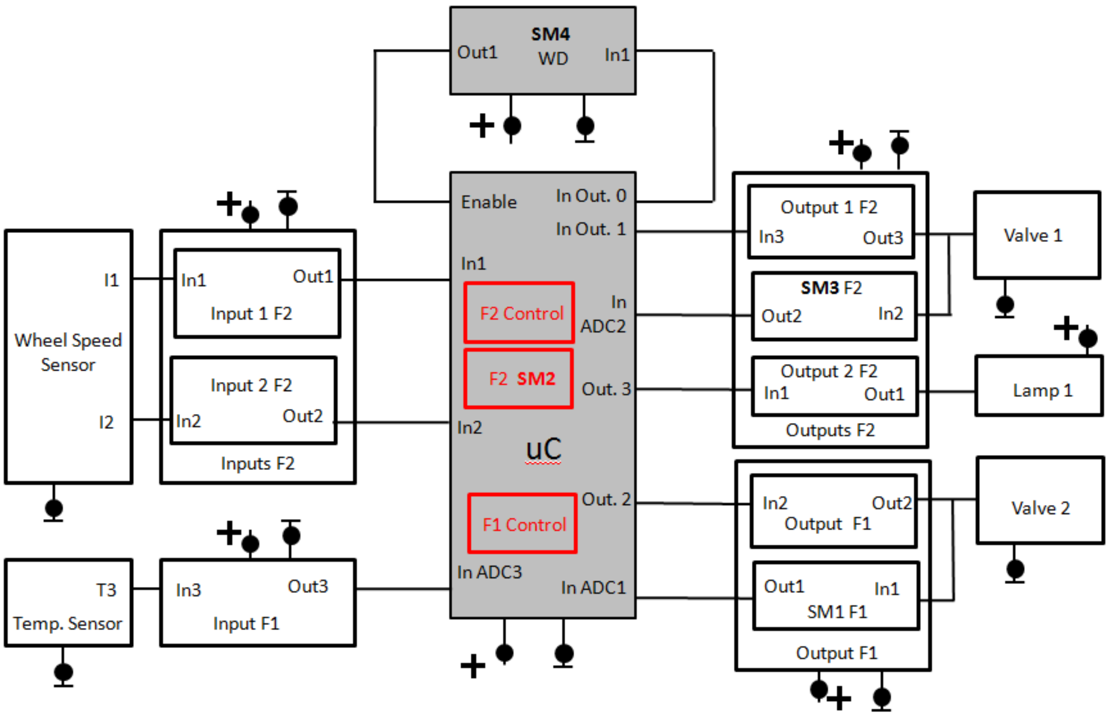


Figure 14: Technical Safety Concept for ISO26262-Part5 Figure E.1

10.1 Step 1: Capture Hardware Technical Safety Concept

- Define HW components
- Clarify HSI
- Define malfunction of each HW component (internalFault as Failure mode, externalFault as input fault and externalFailure as output failure propagation)
- Information: Hardware architecture of the Technical Safety Concept in this context is an assembly of hardware component, as shown as black boxes in Figure 14.

10.2 Step 2: Complete HW Component Failure Propagation on Hardware Architecture

- Propagate fault failure link between all hardware components from WT3.3.1
- Identify contribution to top level malfunction of the Hardware architecture
- Information: Safety mechanism are already in architecture model (loop to Step1 can be added as a result of safety analysis)
- Complete the qualitative safety analysis (from WT3.3.1)
- Classify failure character and contribution for each fault thanks to cut-set order and coverage by a safety requirement with specification of diagnostic coverage of the safety mechanism; tag failure (Single Point, Residual, Multiple Point Latent)
- Identify primary Hardware Safety Requirements based on the top-level malfunction of the HW Architecture. The primary Hardware Safety Requirements shall prevent the occurrence of the malfunctions of the Hardware Components.

10.3 Step 3: Define target values for HW Components and calculate metrics

- Estimate (or use existing) values for Failure Rate and distribution as target value of the HW Component
- Estimate (or use existing) values for Diagnostic Coverage (Latent and Residual) of the Safety Mechanism as target value for the HW Component
- Compute metrics: Evaluate the hardware architectural metric results (see WT3.3.3 for details) and define additional measures (or revise assumption target values)
- Validate the preliminary hardware architectural metric results with the target value of the ASIL

10.4 Step 4: Define Hardware Part Allocation and Malfunction

- Design decision for merging HW Components to build a HW Part (exclusively for complex hardware like ASICs or microcontroller)
- Information: Estimated failure rates and diagnostic coverage of HW Components are used for HW Part analysis (and further composition of HW Part)
- Information: The merged HW Components contains the primary Hardware Safety Requirement (malfunction) which is used for Hardware Part Analysis

Remarks: It shall be noticed that HW Component shall be not be decomposed into several HW Parts (ASIC for example). This may influence badly the quantitative measurement on hardware architecture. If such request is necessary, the complete architecture shall be redesign with the respect of HW component is indivisible component.

10.5 Step 5: Develop Electronics Schematic

- Capture all electronic Hardware Parts as Hardware Elements in AUTOSAR (as from Figure 13) (complex Hardware and resistors, capacitor, etc.)
- Identify the concrete industry references for all HW Parts regarding technology, etc (Bill of material (BOM) as a result)

10.6 Step 6: Perform Electronic FMEA and contribution to HW Component malfunction

- Perform Electronic FMEA (based on electronic schematic) in order to identify HW Part Failure contribution to HW Component malfunction (as Failure Mode)
- Define logical behavioral relation (using AND and OR formula) between Failure Mode of HW Part and malfunction of the HW component (as Failure mode)
- Allocate failure rate and distribution from industrial data base to HW Parts from the BOM
- Allocate real value for Safety Mechanism diagnostic coverage (Latent and Residual) for all relevant HW Parts from the BOM
- Compute Failure Rate of the malfunction of the HW Component (from the behavioral relation)

10.7 Step 7: Verify Component Metrics and Probabilistic value

- Reintroduce at the Hardware Architecture level the computed Failure rate for HW Component to verify the hardware architectural metrics

11 IP-XACT interchange

The IP-XACT format is a well define XML schema for meta data that documents the characteristics of hardware Intellectual Property (IP) for the automation of the configuration and the integration of IP blocks. This is an IEEE standard by the ACCELERA Systems Initiative IP-XACT technical committee (see [4]]) that allow to exchange hardware digital IP elements, to manage them in libraries, to configure them and to automate their integration into a hardware design.

Three main element of IP-XACT can be introduced as component, bus interface and design, represented by the picture below (from document [9]). The component, from depicted Figure 15, describes all internal characteristics and external interfaces as for example bus interface. Then components are gathered in a so called design, as visible in Figure 16.

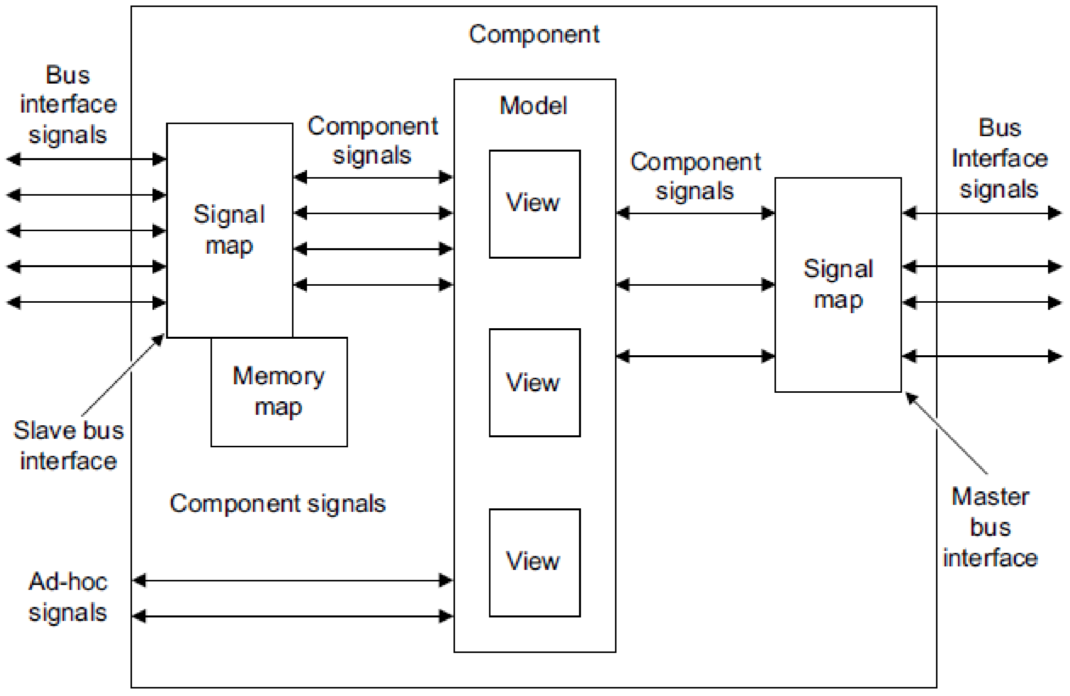


Figure 15: Structure of a component IP-XACT

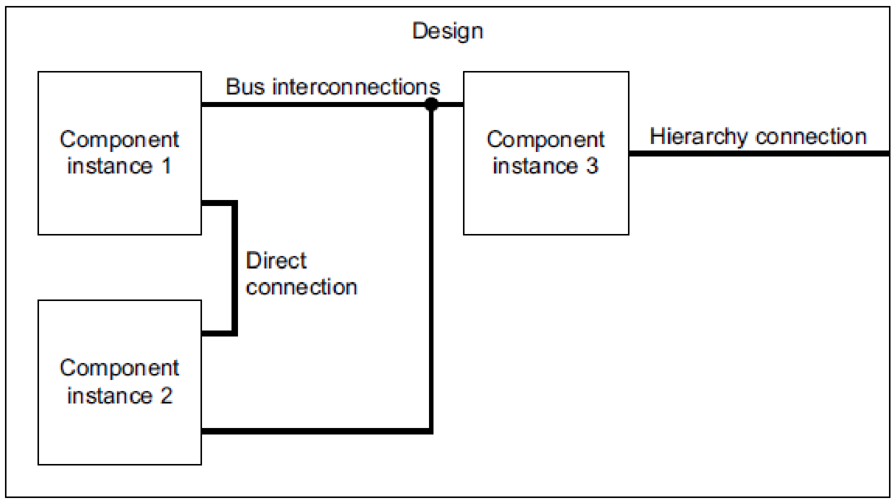


Figure 16: Design representation in IP-XACT

Due to actual limitation of digital element and interface, an extension for addressing analogue mixed signal domain is under discussion in the technical committee, expecting a potential candidate date for mid 2013. By default, the actual extension point "vendor extension" allow to define this extension, but it usage is not standardized.

In comparison, the actual AUTOSAR R4.0 meta model allows definition of digital and analog component as HWComponent using mechanism of HWCategory for specialization of hardware type as proposed in 8.3 and 8.4. Moreover, as show from Figure 17, a HWElement as a specialization of an HWDescriptionEntity is referencing an HWCategory composed by HW attributes definition. Such mechanism allows defining electrical characteristics associated to each HWElement and in particular HWPin. For more details on use of HWCategory please refers to AUTOSAR documented.

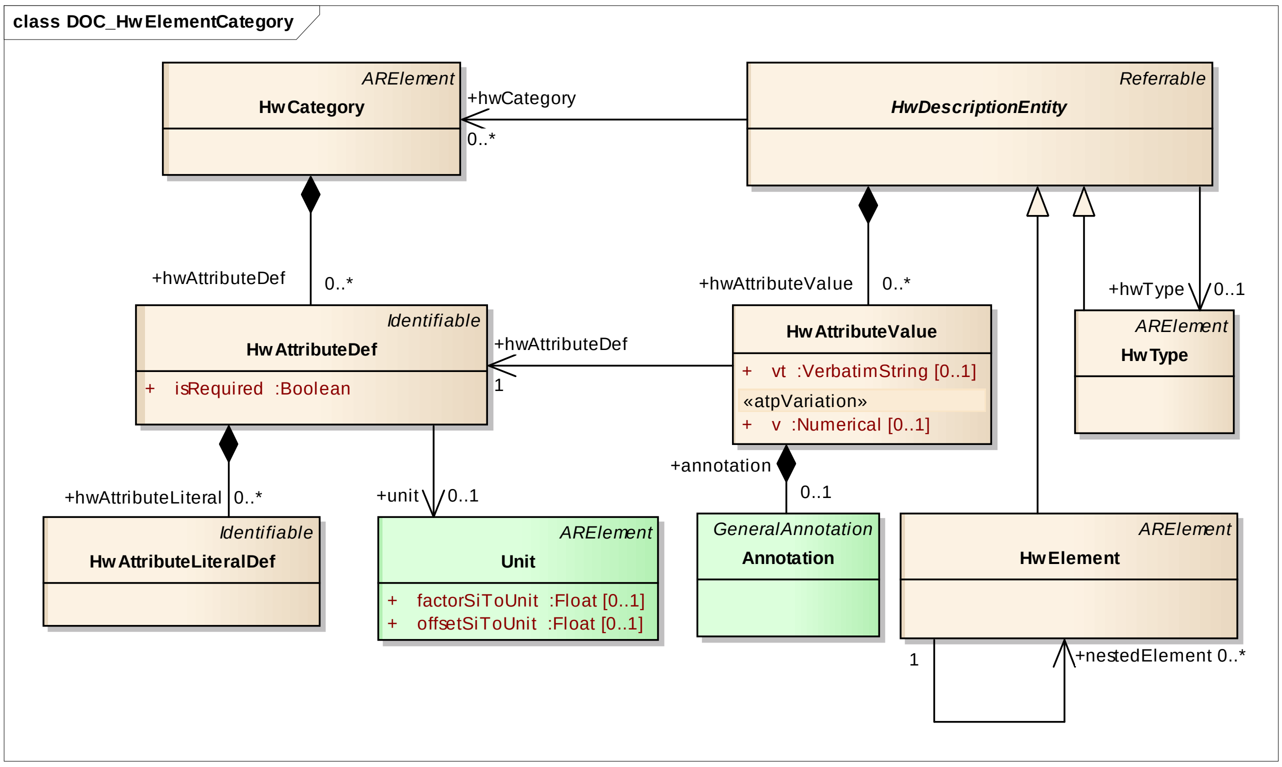


Figure 17: AUTOSAR HWcomponent and HWCategory

In order to be able to support hardware exchange element via IP-XACT interchange the existing classes from the IP-XACT XML definition to AUTOSAR ECU Resource Template selected meta Class has to be mapped. This preliminary mapping of respective IP-XACT classes versus AUTOSAR hardware elements will be specified in the next section. Thanks to the vendor extension we may propose an extension to support IP-XACT failure information modeling. Such mapping will then allow the writing of a model to model transformation to improve data exchange between silicon semi-conductor suppliers and automotive product suppliers.

11.1 Mapping rules

This section will represent a first draft for mapping the classes of the hardware parts from AUTOSAR R4.0 HWElement (with consideration of Type and Prototype) versus components of the IP-XACT IEEE1685-2009 standard.

The use of the character "~" in the table below, identify that construct of AUTOSAR can be applied but requires restriction and limitation in the use, as design pattern for specific usage (e.g. IP-XACT semantic definition a BusInterface compare to simple composition of HwPin (and HwPinGroup) in HwPingroup.

AUTOSAR R4.0	IP-XACT	Remarks for IP-XACT
HwElementType	Component Vendor/Library/Name	In addition IP-XACT identifies a Version attribute for an element (information from change management). It is so called VLN/V.
~ HwPinGroup	BusInterface Vendor/Library/Name	For bus interface definition, the parameter AbstractionType and BusType are managed under VLNV control.
~ HwElementType	Designs Vendor/Library/Name	VLNV control. Represent a Composition of ComponentInstances A design is always embedded in a component that defines top level interface.
IP-XACT Component description		
HwElementType	Design/Library/Name	Basic entry for Component description. A component can include a Design
	Model	Intermediate level to represent the element respective to the model as Ports, View and ModelParameter. A View represents an abstract level defining mapping to FileSets. A Port defines individual signal wire or transactional interface A ModelParameter defines configurations
	FileSets	Intermediate level for behavioral definition of the VLNV for definition of code execution source
	MemoryMap	Represent the information about the internal register. Is is not defined in AUTOSAR as MCAL implementation linked
MemoryMapped Assembly HWConnection	AdressesSpaces	Defines the memory mapping of the IP inside the CPU space address
~ HwPinGroup	BusInterface/BusInterfaces Salve/master BusType AbstractionType PortMap	Define a Bus Interface of the component Slave/Master defines access mode for direction and a logical name (or Monitor/System with Mirrored option for checking interface connection) BusType defines the Bus and high level attributes as compatibility rules. AbstractionType defines low levels signal implementation of a given BusType by logical name (wire or transactional). Several abstractions can be defined for a same BusType. PortMap define the mapping of logical port (wire or transactional) to a logical port physical mapping to the signal.

HwPin	Ports/Port/Wire/ WireTypeDefs/WireTypeDef	Digital port direction as single signal or vector of signal or a TLM port for transaction. Reference to FileSets behavior parameter.
HwPin	Ports/Port/Wire/ SignalTypeDefs/SignalTypeDef DomainTypeDefs/DomainTypeDef	Analogue domain definition. Reference to SignalType (discrete or continuous for AMS simulator) or DomainType (continuous analog or others domain for multi-domain simulator) with typeDefinition (reference to domain definition) or signalType (AMS model definition) and with viewNameRef as FileSets for code behavior parameter.
~ HwPinGroup	Ports/Port/Transactional/Service/ ServiceTypDef TransTypeDef	Digital transaction direction definition. ServiceTypDef definition the type of TLM transaction (digital simulator) and parameter. TransType as reference to FileSets for code behavior parameter.
HwCategory	View	Allow the definition two additional parameter “Language and Model Name” and “File Set Ref .List” the typing of the IP-XACT Port for the model of execution (digital, TimeDataFlow, Electrical Network).
IP-XACT Design description		
HwElementPrototype	ComponentInstances	Component instance name of of ComponentRef referencing the VLNV component inside the library. Port interface, as wire or transactional, is defined by portConnectors referencing physical Port of the component. Bus interface is defined by busConnectors reference Bus interfaces name of the component.
HwPinConnector	adHocConnections	Connecting two Ports with wire or transactional interface without using bus interface. The ports can be an internal port of the instance component as internalPortReference referencing componentRef as component instance name and portRef as Port name of the Component. Or it can be an external port of the design component as ExternalPortReference referenced by portRef as port name of the deign component.
~HwPinGroupConnector	hierConnections	Hierarchical connection of bus interface, identified by interfaceRef from the design component bus interface, and connected to a bus interface of a component instance referenced by componentRef as component instance name and busRef as Component Bus Interface name.
~HwPinGroupConnector	interConnections	Connection between two bus interfaces of component instance referenced by componentRef as component instance name and busRef as component bus Interface name of the component.

Figure 18: Class mapping between AUTOSAR and IP-XACT

11.2 Extension for failure information

The objective of this section is to find a solution on how to define that the failure information of the hardware part such mainly as failure mode, failure rate and distribution is attached to hardware element of IP-XACT. The objective is to ensure that failure relevant information can be transmitted with hardware component information and package. The selected data are the attributes of the classes depicted in Figure 19 as part the hardware meta model from section 9.

The IP-XACT vendor extensions concept allows registering definition of extra elements thanks the VPN(V) component. The selected data can be defined as parameter of the vendor extension in a new field failureDefs attached to components.

Then, the new field shall allow defining multiple *failureModeDefs/failureModeDef* for definition of component failure mode data and a single *failureRateDefs/failureRateDef* for definition of failure rate in time data of a component.

The proposal is to decompose the *failureModeDef* field in three parameters as the attributes of the ***HWPartFailureMode*** class from Figure 19 (*failureModeTypeDef* for the definition of failure type, *failureModeDef* for the definition of failure mode, *failureRateDistributionDef* for of the failure distribution, *failureModePotentialCauseDef* for textual definition of potential failure cause if relevant). The *failureRateDef* field is decomposed in two parameters as the attributes of the ***HWPartFailureRate*** class from Figure 19 (*failureRateValueDef* for definition of failure rate, *failureRateSourceDef* for textual definition of industry source of failure rate). In ***failureRateDef*** field, two addition parameters as *failureScalingFactor* and *failureRationaleScalingFactor* shall be optional as they depend of the use of the component in an IP-XACT Design.

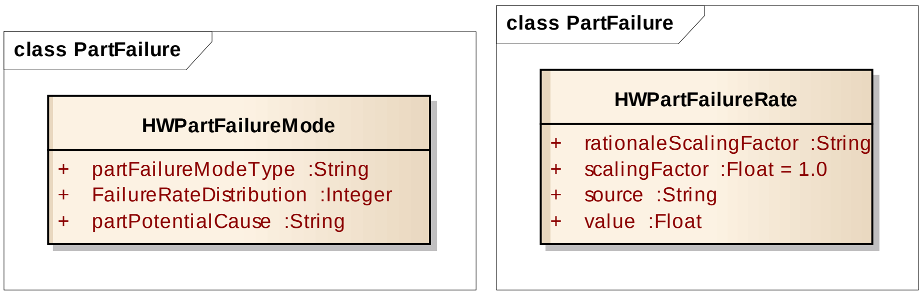

class PartFailure
HWPartFailureMode
+ partFailureModeType :String
+ FailureRateDistribution :Integer
+ partPotentialCause :String
class PartFailure
HWPartFailureRate
+ rationaleScalingFactor :String
+ scalingFactor :Float = 1.0
+ source :String
+ value :Float

Figure 19: Hardware Part Failure information for IP-XACT

As, this chapter is only an initial proposal it can only be discuss with Accelera member and align with ongoing activities defined in the Accelera IP-XACT work group.

12 Conclusions and Discussion

This document provides a proposal for adaption and extension of hardware structural and failure modeling. Additionally, constructs for quantitative safety evaluation of hardware in terms of hardware architectural metrics and evaluation of residual risk of safety goal violation conform to requirements addressed by ISO26262.

Since it was an objective to reuse EAST-ADL as much as possible, the current version of EAST-ADLV2.1 and AUTOSAR R4.0 were analyzed. Concrete proposal for future change request in these architecture description languages (expressed as meta model solution for EAST-ADL 2.1) is provided.

This document has been produced to support the WT3.3.1 overall safety evaluation methodology, and to provide meta model constructs to WT3.3.3 for the model-based safety evaluation of hardware. WT4.2.6 describes a research prototype implementation.

13 References

[1] ISO 26262 Road vehicles - Functional safety. (2011), Part 5 : Product development at the hardware level, http://www.iso.org/

[2] AUTOSAR Automotive Open System Architecture R4.0, AUTOSAR_RessourceTemplate_ECU.pdf, http://www.autosar.org/

[3] EAST-ADL language and Association, http://www.east-adl.info/

[4] IP-XACT Accelera System Initiative, http://www.accellera.org/activities/committees/ip-xact

[5] EAST-ADL language and Association, Deliverable D4.1.1: EAST-ADL Domain Model Specification, 2010

[6] Chen, D., Johansson, R., Lönn, H., Papadopoulos, Y., Sandberg, A., Törner, F., Törngren, M.: Modelling Support for Design of Safety-Critical Automotive Embedded Systems. In: Proceedings of SAFECOMP (2008)

[7] Peikenkamp, T., Cavallo, A., Valacca, L., Böde, E., Pretzer, M., Hahn, E.M.: Towards a Unified Model-Based Safety Assessment. In: Proceedings of SAFECOMP. (2006) 275–288

[8] SAFE_D3.3.3a.pdf (Specification for comparison of architecture)

[9] ARM reference manual for IP-XACT Component version 1.0 from 2007, http://infocenter.arm.com/help/topic/com.arm.doc.ddi0429a/DDI0429A_ip_exact_components_v1_0_ref_manual.pdf

[10] SAFE_D3.1.1c.pdf (Specification for requirements definition and traceability purpose)

[11] SAFE_D3.3.3b.pdf (Final specification for comparison of architecture)

[12] SAFE_D3.3.1b.pdf (Methodology and Tool specification for analysis of qualitative and quantitative cut-sets issued from error failure propagation analyses)

14 Acknowledgments

This document is based on the SAFE project in the framework of the ITEA2, EUREKA cluster programme 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). The responsibility for the content rests with the authors.