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 D3.2.2

Proposal for extension of Meta model for hardware modeling

Due date of deliverable: 28/02/2013

Actual submission date: 28/02/2013

Start date of the task: 13/12/2011 **Duration:** 15 months

Task coordinator name: Philippe Cuenot

Organization name of lead contractor for this deliverable: Continental France

Editor: Philippe Cuenot

Contributors: Philippe Cuenot, Nico Adler, Stefan Otten

Reviewers: Florent Meurville, Martin Hillenbrand

© 2012 The SAFE Consortium

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

1 Table of contents

© 2012 The SAFE Consortium

2 List of figures

3 Executive Summary

The work task WT3.2.2 targets the topics of definition of hardware modeling. 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 modeling constructs enabling the calculation of hardware quantitative measures required by the ISO 26262 [1] for hardware architectural metrics and the safety goal evaluation 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 initial methodology for hardware technical safety concept representation, for hardware component failure mode and 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 consumer electronic interchange format IP-XACT [4] from Accelera Organization is discussed. A first overview of proposed links is given.

© 2012 The SAFE Consortium

4 Introduction and overview of document

This document at hand 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 equipment and electronics components. The proposed method will rely on existing automotive standards AUTOSAR and EAST-ADL, and will forecast to elaborate of first approach to connect it to consumer electronics standard IP-XCAT.

4.1 Scope of WT 3.2.2

Work task WT3.2.2 deals with hardware description.

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 representation and decomposition into component part description, with respect to safety evaluation and quantitative measurement related to random hardware failure. The existing current meta-model of EAST-ADL and AUTOSAR will be analyzed to provide proposals for improvement to basic standards via change request and to define appropriate safety-related extensions in terms of the described topics of WT3.5

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

In order to be able to do so, 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 elements with regards to software elements for the software architecture. The hardware architecture level represents the set of hardware components for the intended

© 2012 The SAFE Consortium

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 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 above 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 parts 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 above 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 join calculations between hardware random failure from different hardware abstraction level (hardware architecture and hardware electronic design).

Fault and contribution to Safety Goal/Malfunction

The contribution of the hardware component to the violation of the safety goal shall allow tagging safety-related component. Although the item identified during hazard analysis can be decomposed according to sub-system development scenario. The hardware sub-system can only exhibits a local malfunction, and its contribution to the top level system malfunction linked to the violation of the safety goal. The relation to the top level malfunction, linked to the safety goal, of the local malfunction attached to a sub-part overall architecture, shall be incorporated in the meta model. The basic fault event of the occurrence of the top level malfunction, as hardware component fault, should be characterized by the type of fault (e.g. single point fault, latent fault, multiple or residual fault).

Hardware Metrics and Probabilistic value

Based on the hardware component faults, theirs relations for safety mechanism and associated coverage rate, the hardware architecture metrics (Single Point Fault and Latent Fault metric) need to be allocated first and subsequent 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 measures with their respective parameters or assumption. It shall also be able to express relation over the assumption of the logical hardware component and physical component part, to offer basic repository for the complete failure analysis methodology defined in WT3.3.1.

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 standard 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 measures 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.

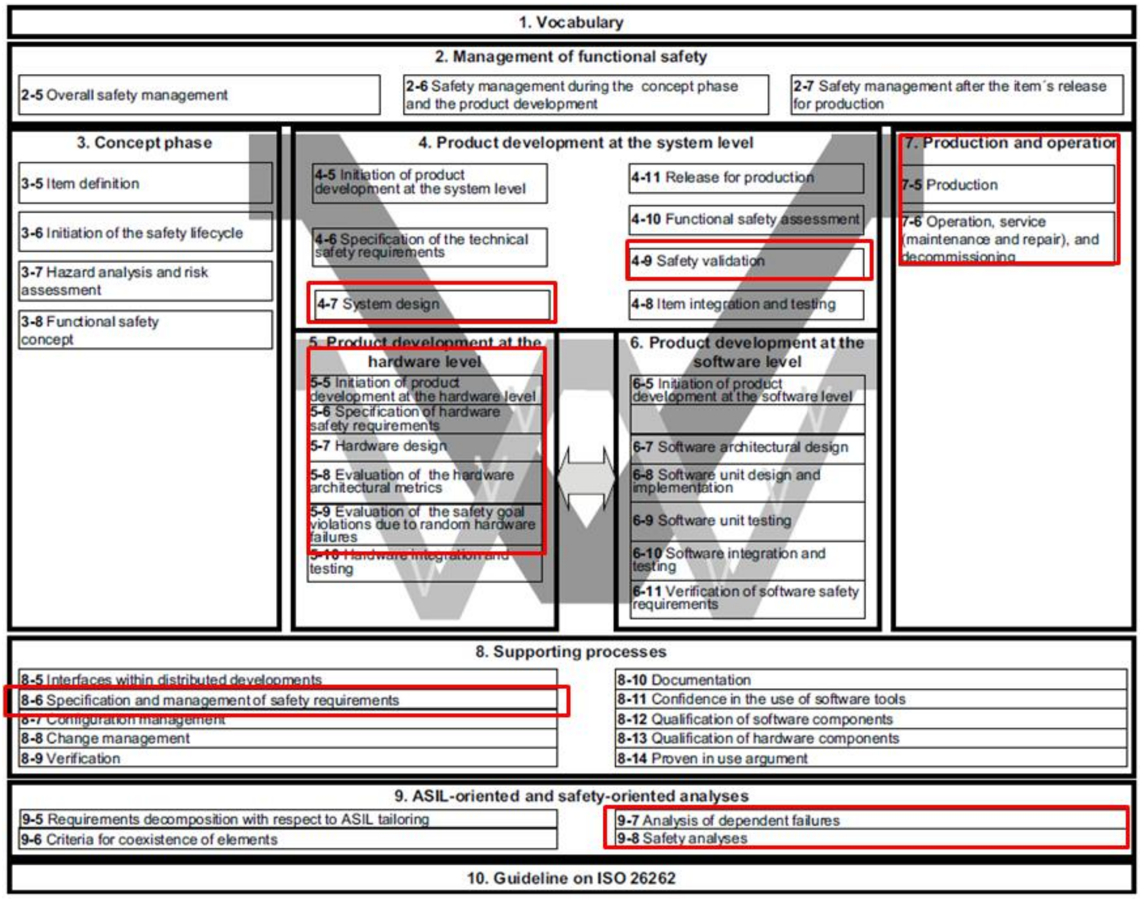

Figure 1: Overview on ISO 26262 (Relevant parts highlighted)

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 mechanisms to define measure 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 tolerant fault 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 component.

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 for the hardware architecture shall be defined according to architecture single point fault and latent-multiple fault, and for quantification of avoidance of the violation of safety goal due to random hardware failure.

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 for random hardware failure 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 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 parts interconnected. The traceability between hardware and safety requirement shall be traceable down to the lowest level. The environmental condition and potential cause of failure of hardware component shall be considered during design of hardware component.

Safety Analysis

The safety analysis on hardware design identifies the causes of failure and effect of faults in the overall system failure. The effectiveness of safety mechanism shall demonstrate to avoid single-point fault, to maintain the system in safe sate and to validate coverage with respect 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 design 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.

Evaluation of safety goal violation due to random hardware failure

The evaluation of the residual risk of violation of safety goal due to random hardware failure due to single-point fault, 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 used either Probabilistic Metrics for random Hardware Failure (PMHF) which is build by a quantification of a fault tree analysis, or Failure Rate Class (FRC) method which basically evaluates each fault (single-point, latent …) for each individual hardware component. 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.

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 "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 coexistence 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 above sections an overview is given.

Analysis of Dependent Failure

The analysis of depended failures on the architecture induces to introduce specific measure to be applied to architecture element (e.g. such as redundancy, dissimilar development, safety mechanism, physical barrier, etc). A common cause failure and cascading analysis failures analysis shall

© 2012 The SAFE Consortium

be performed for the architecture considering operational life of the product. This evaluation shall be performed on systematic fault, random hardware failure 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 element 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.

© 2012 The SAFE Consortium

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.

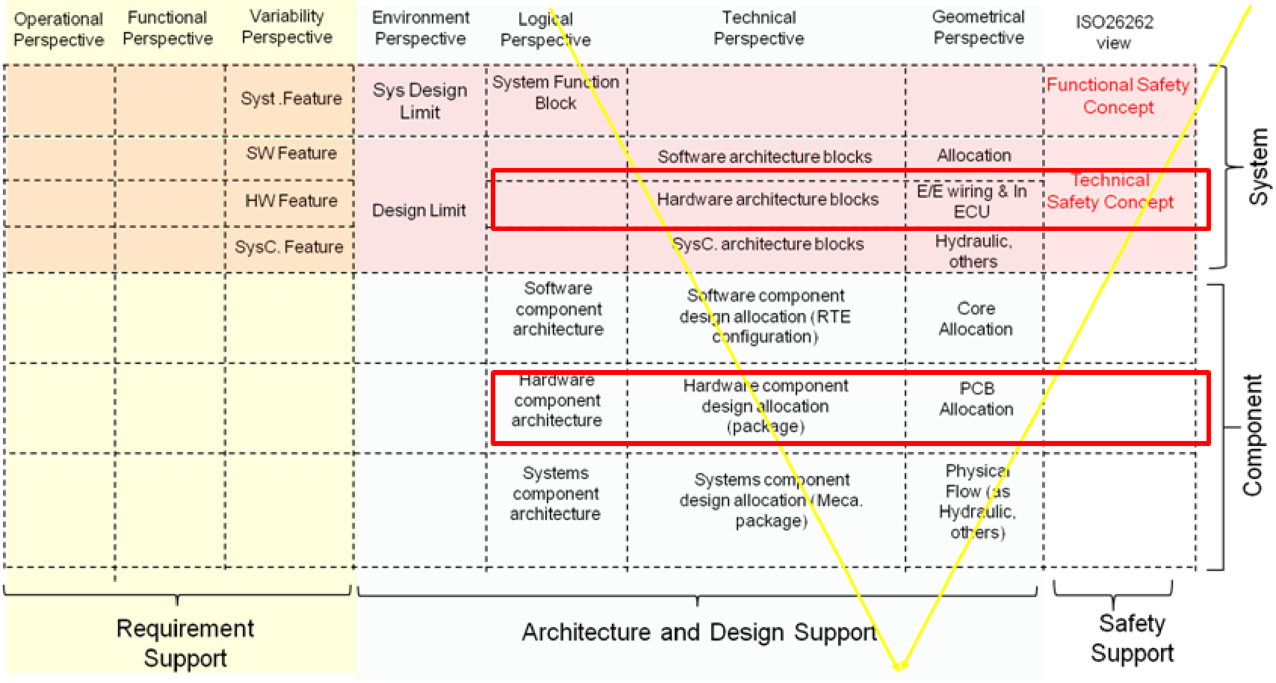

Figure 2: Overview on structure of architecture (Relevant parts highlighted)

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 characterization, 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.

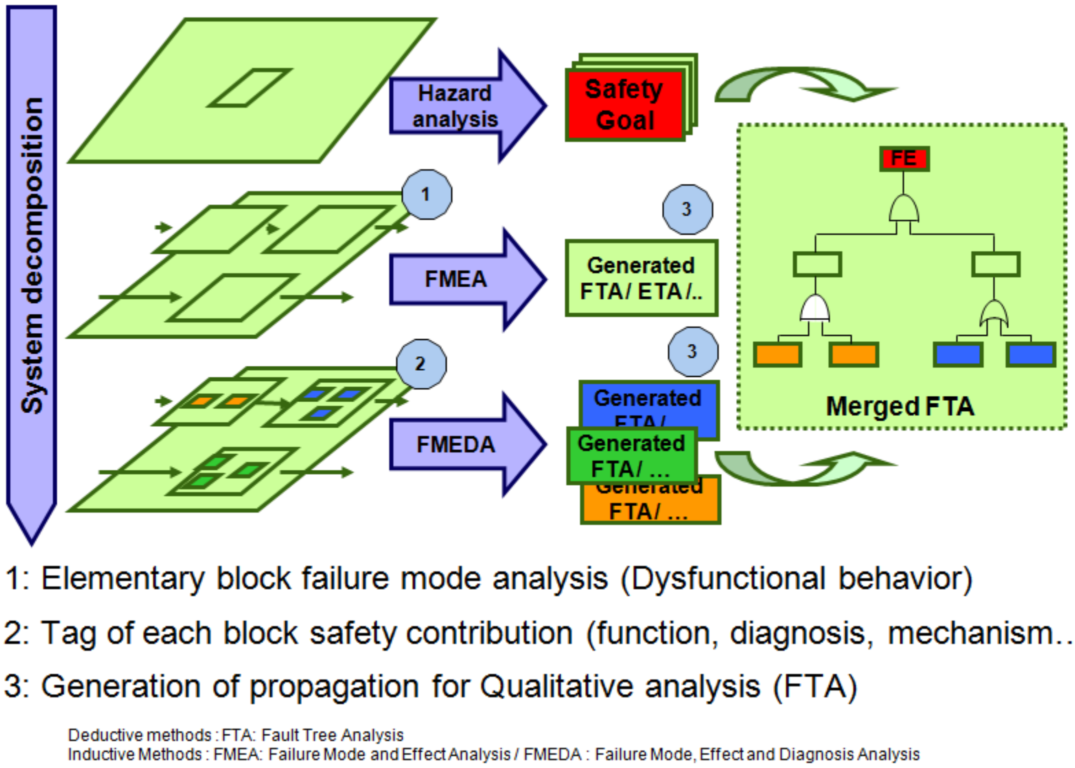

Figure 3: Overview on iterative safety analysis methods

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.

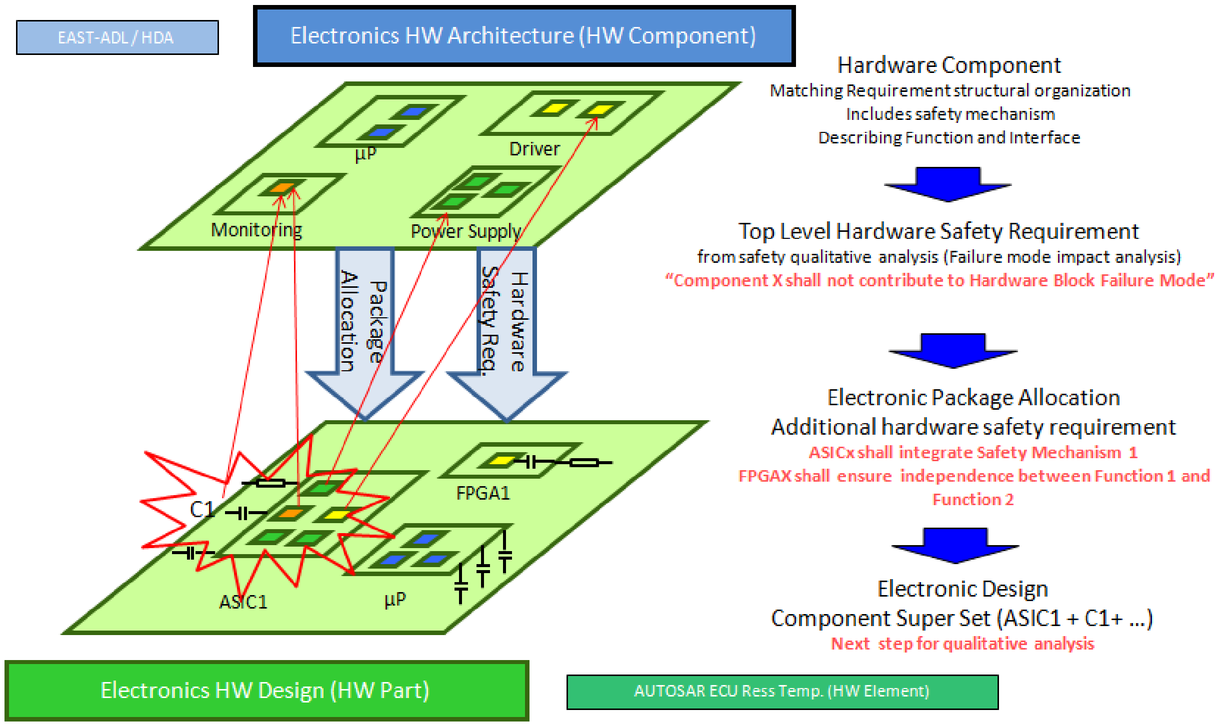

Figure 4: Hardware allocation and quantitative analysis

© 2012 The SAFE Consortium

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 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.

© 2012 The SAFE Consortium

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 synthesize in 7, is initiate 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 characterization 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.

© 2012 The SAFE Consortium

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.

© 2012 The SAFE Consortium

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 characterization. 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 are 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.

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
- Sub-Package *HWQuantitativeMeasure* for the classification of the assessments to the architectural metrics or probabilistic methods for hardware safety evaluation. Additionally, the quantitative assessment for the calculation of single contribution for each failure mode is included.
- Sub-Package *HWArchitecturalMetrics*, according to the requirements category Hardware Architectural Metrics
- Sub-Package *ProbabilisticMethods*, according to the requirements category Safety Goal Violation
- Sub-Package *Traceability*, according to the requirements category Traceability mainly reusing existing artifacts from EAST-ADL

© 2012 The SAFE Consortium

- 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.

© 2012 The SAFE Consortium

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.

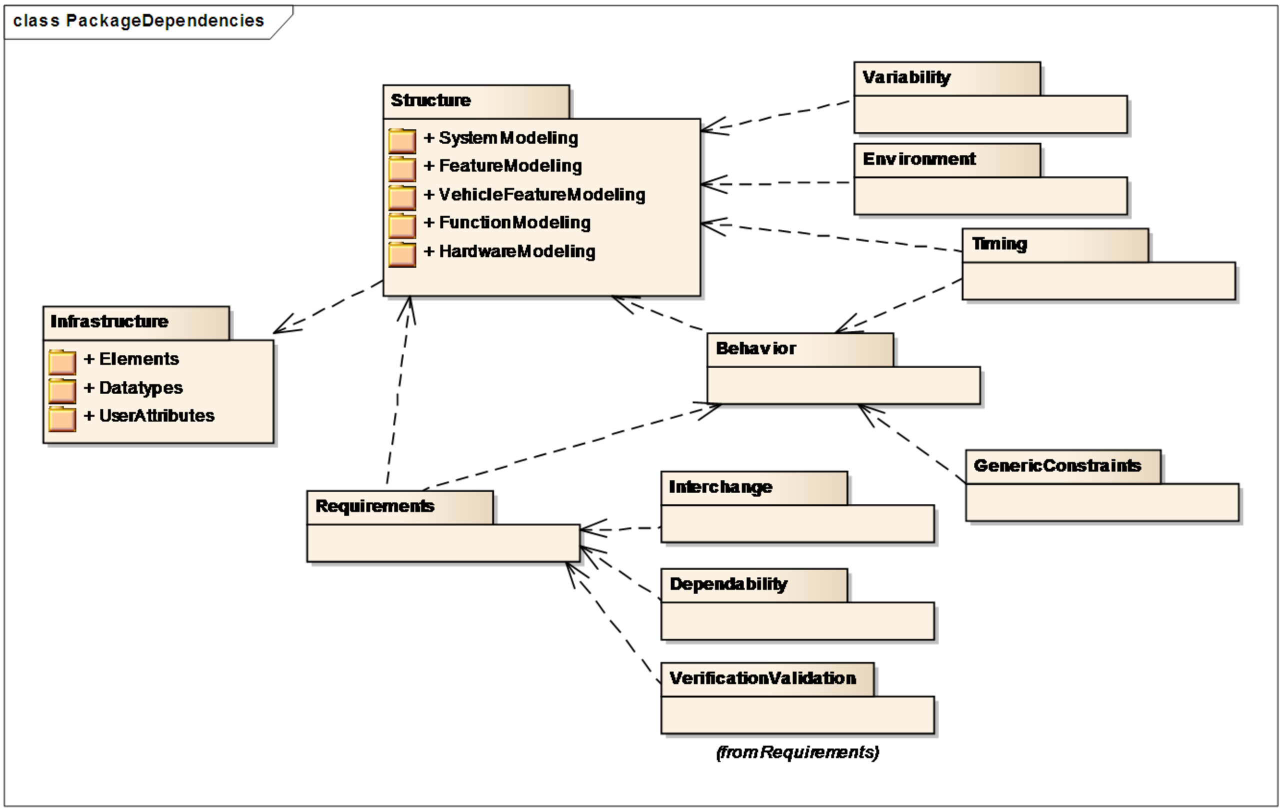

Figure 5: Class diagram for Package Dependencies

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.

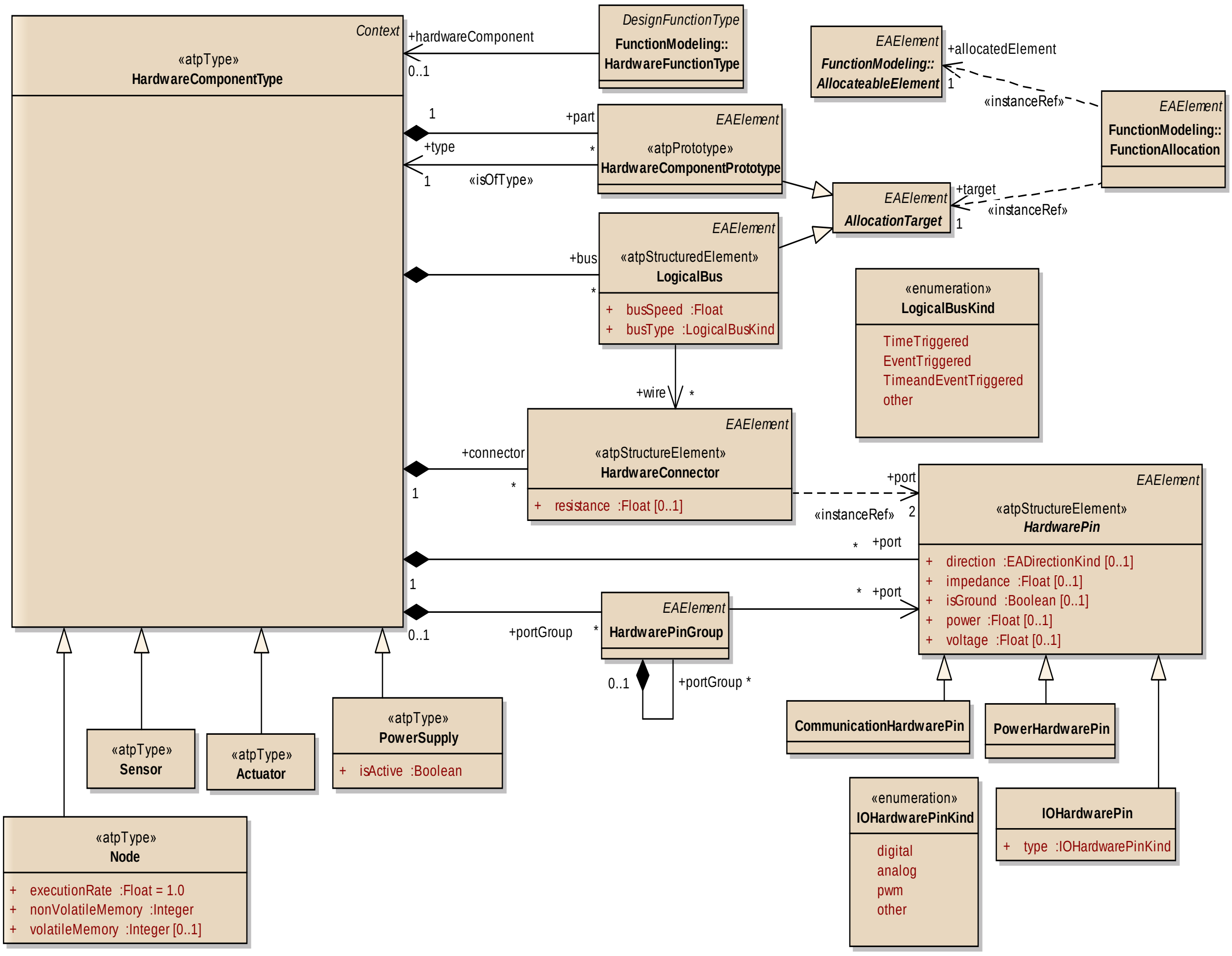

Figure 6: Class diagram for Hardware Modeling in the EAST-ADL2

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 behavior 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.

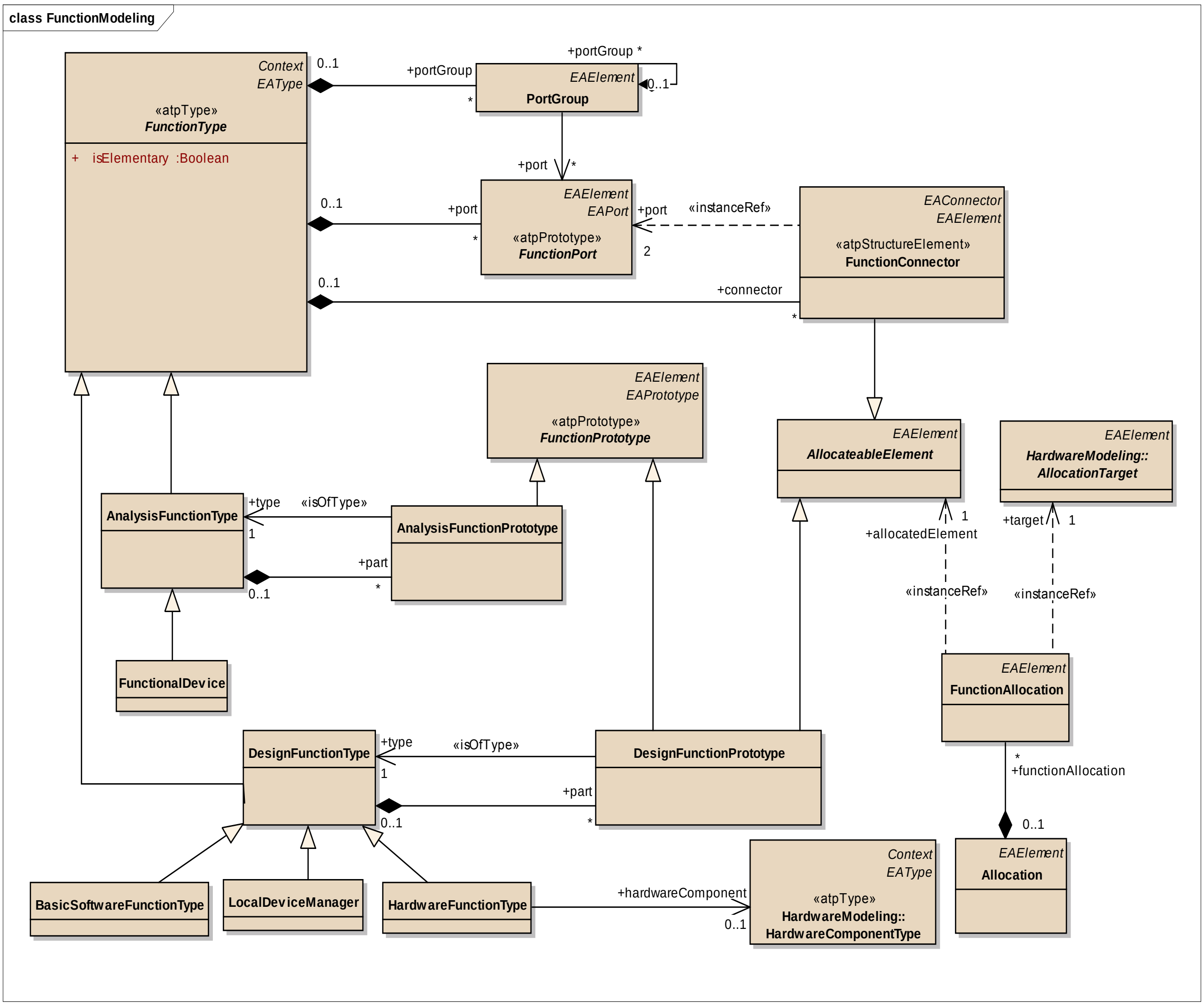

Figure 7: Class diagram for Function Modeling in the EAST-ADL2

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.

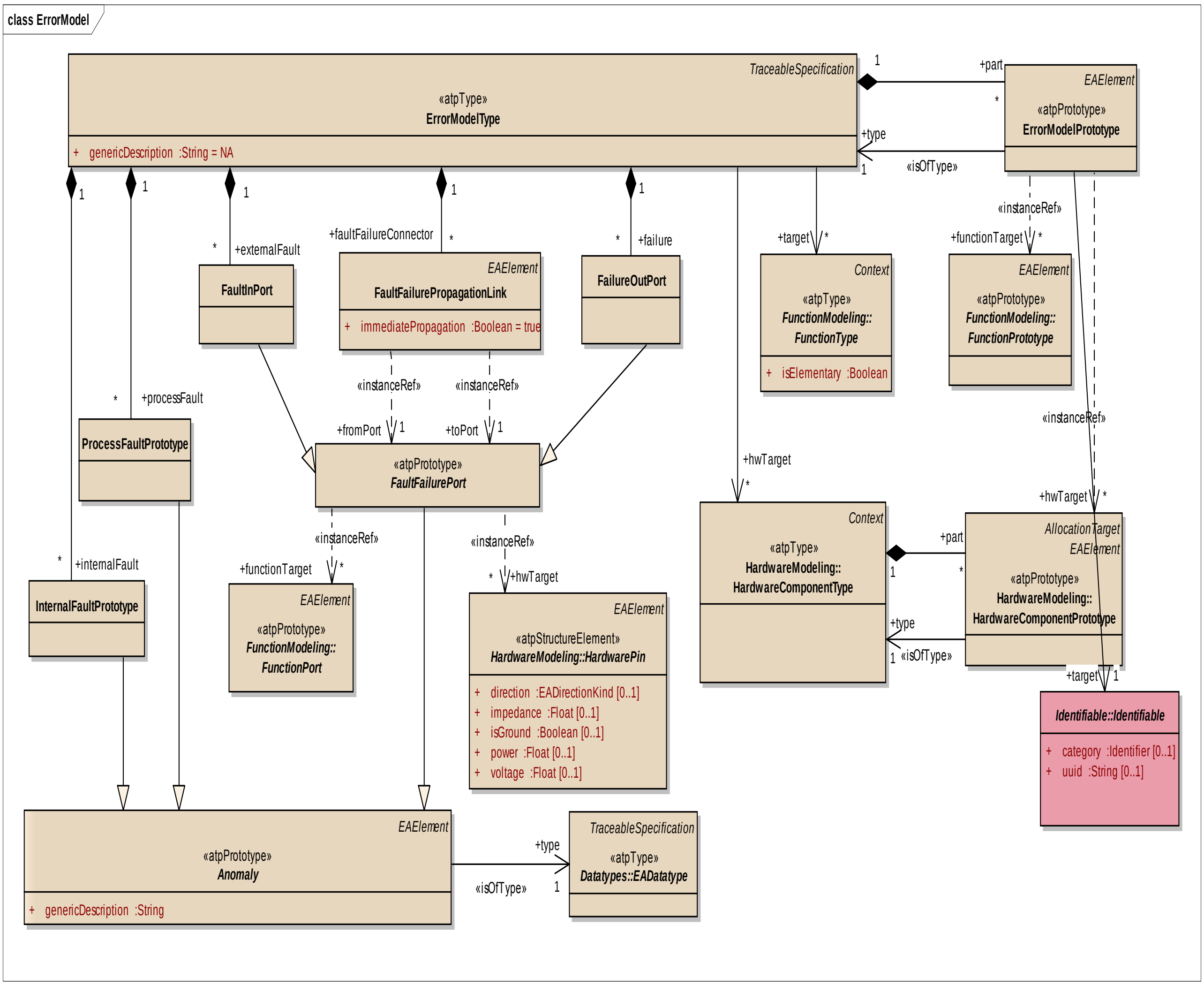

Figure 8: Class diagram for Error Modeling in the EAST-ADL2 Dependability

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 *ErrorModel* as typed *FailureOutPort*.

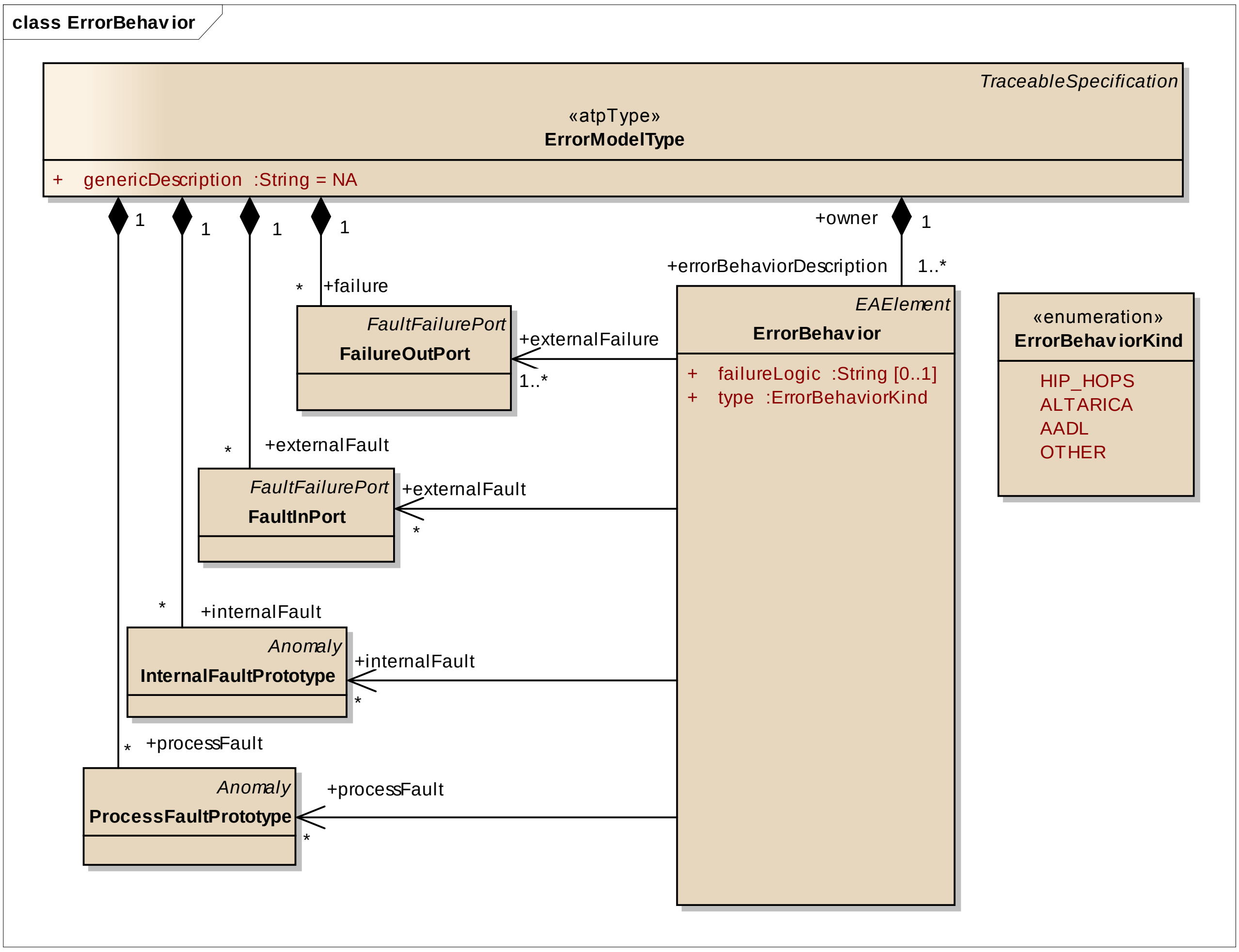

Figure 9: Class diagram for Error Behavior in the EAST-ADL2 Dependability

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

© 2012 The SAFE Consortium

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.

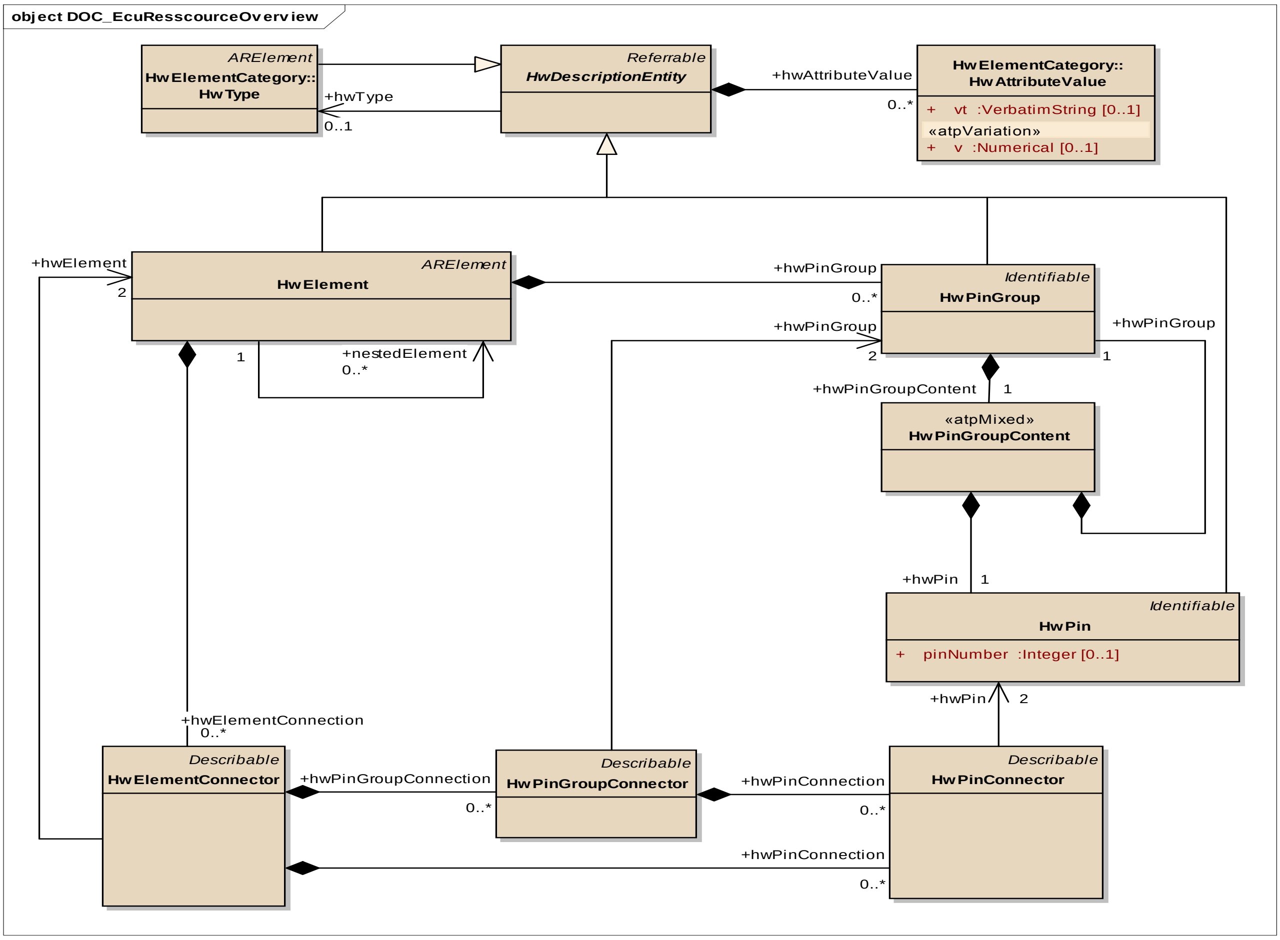

Figure 10: AUTOSAR ECU Resource overview

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.

.

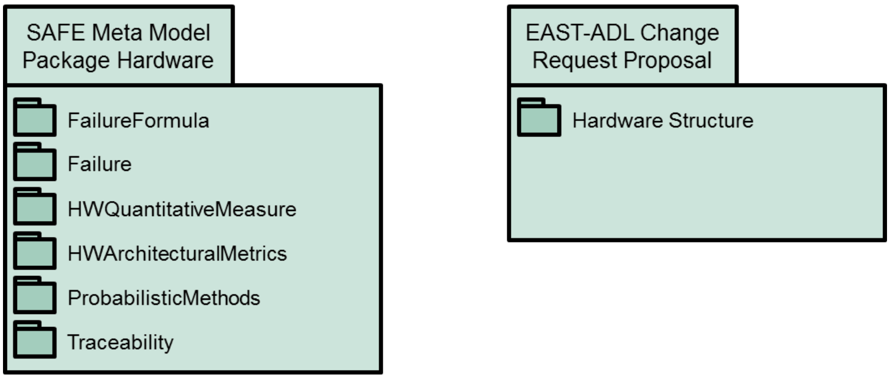

Figure 11: Overview on WT 3.2.2-contribution to SAFE meta-model

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 *HarwdareStructure.*

The package *Hardware* with its sub-packages FailureFormula, Failure, HWQuantitativeMeasure, HWArchitecturelMetrics, ProbabilisticMethods and Traceability 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.

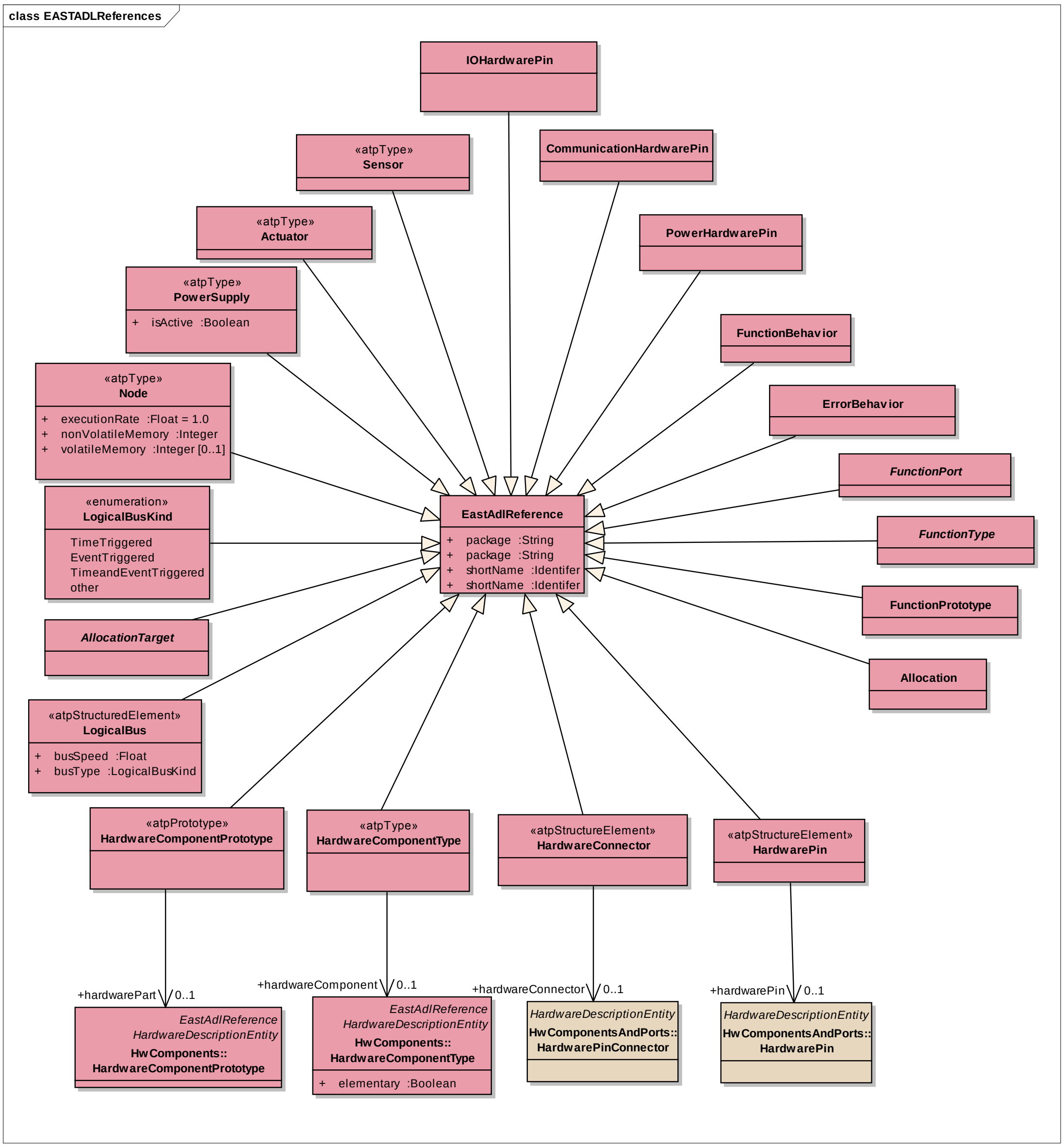

Figure 12: References of package Hardware to EAST-ADL

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

Package Notes:

This package describes the Change Request proposal for the original EAST-ADL package HardwareModeling

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.

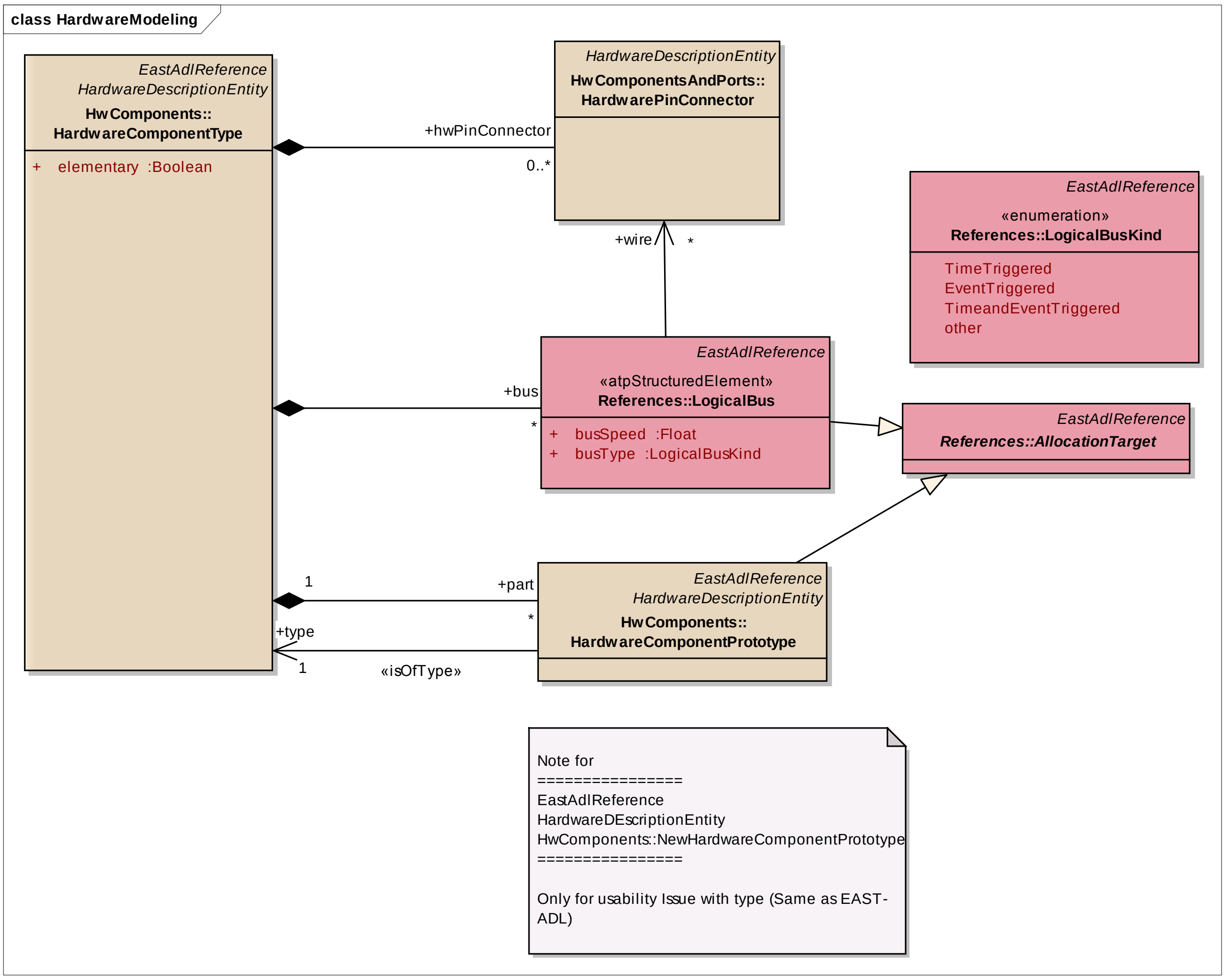

Figure 1: **HardwareModeling** - *(Class diagram)*

Diagram Notes:

This diagram shows an overview of the basic element of HardwareModeling as HardwareComponentType and HardwareComponentPrototype.

It also depicts the conservation of LogicalBus for backward compatibility. It is now proposed to be replaced by a more flexible concept the HardwarePort.

9.2.1.1 Package HwCategory

Package Notes:

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.

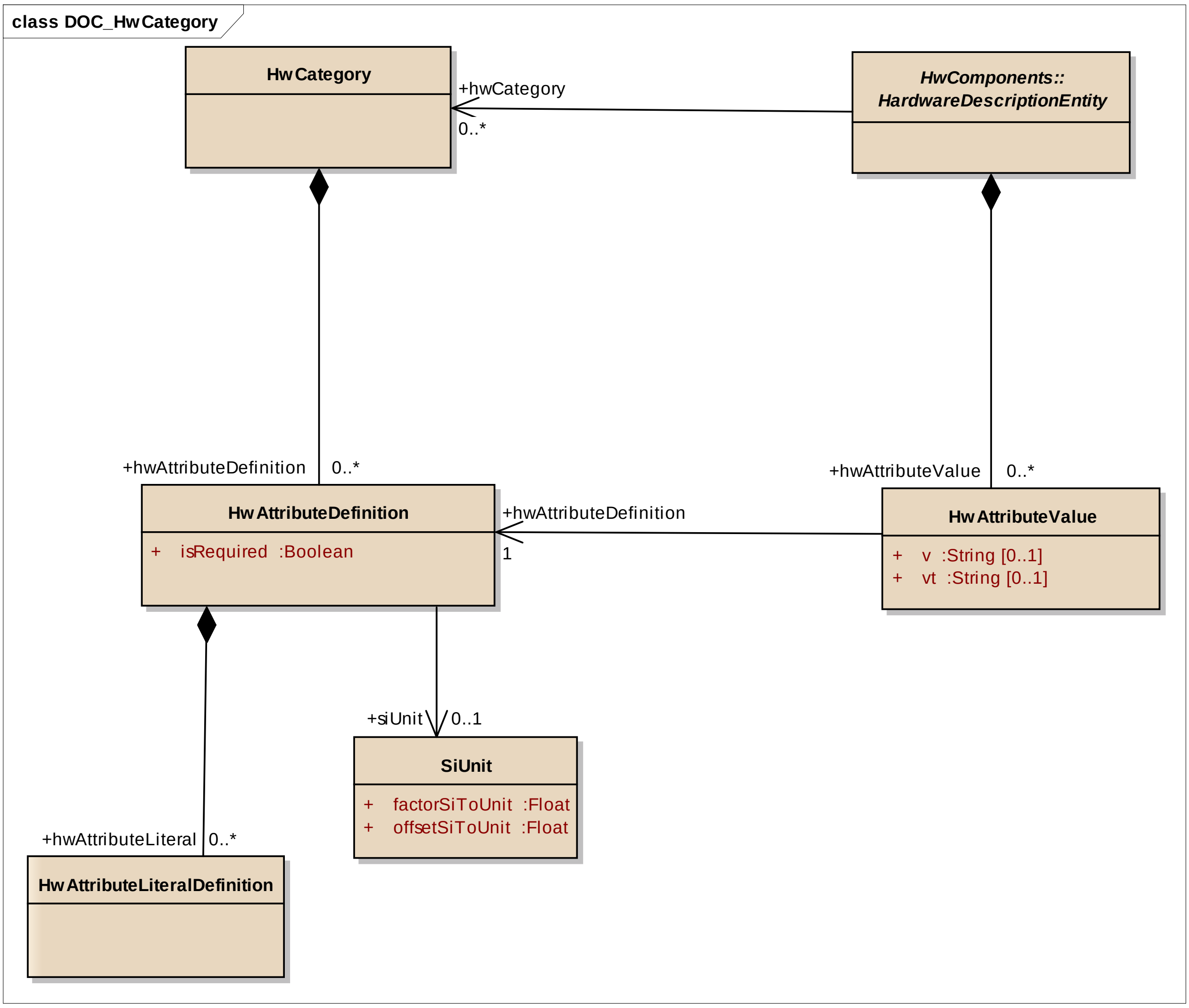

Figure 2: **DOC_HwCategory** - *(Class diagram)*

Diagram Notes:

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.

9.2.1.1.1 Class HwAttributeDefinition

Element Base Classes:

Element Notes:

This HwAttributeDefinition class represents the ability to define a particular hardware attribute.

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.

Semantic:

none

Connections

Connector	Source	Target
Association Source -> Destination	HwAttributeDefinition	SiUnit
Aggregation Source -> Destination	HwAttributeDefinition	HwCategory
Association Source -> Destination	HwAttributeValue	HwAttributeDefinition
Aggregation Source -> Destination	HwAttributeLiteralDefinition	HwAttributeDefinition

Attributes

Attribute	Notes	Default
isRequired Boolean	This attribute specifies if the defined attribute value is required to be provided.	

9.2.1.1.2 Class HwAttributeLiteralDefinition

Element Base Classes:

Element Notes:

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.

Semantic:

None

Connections

Connector	Source	Target
Aggregation	HwAttributeLiteralDefinition	HwAttributeDefinition

Connector	Source	Target
Source -> Destination		

9.2.1.1.3 Class HwAttributeValue

Element Base Classes:

Element Notes:

This HwAttributeValue class represents the ability to assign a hardware attribute value. Note that v and vt are mutually exclusive.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HwAttributeValue	HardwarePort
Aggregation Source -> Destination	HwAttributeValue	HardwarePin
Aggregation Source -> Destination	HwAttributeValue	HardwareDescriptionEntity
Aggregation Source -> Destination	HwAttributeValue	HardwareComponentPrototype
Association Source -> Destination	HwAttributeValue	HwAttributeDefinition

Attributes

Attribute	Notes	Default
v String	This represents a textual hardware attribute value.	
vt String	This represents a numerical hardware attribute value.	

9.2.1.1.4 Class HwCategory

Element Base Classes:

Element Notes:

This HwCategory class represents the ability to declare hardware category and its particular 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).

In addition, this construct can be attached to any HardwareComponent for further characteristic description (e.g. technology, etc...).

The decision for introduction of this element was to introduce a flexible definition of parameter for any hardware entity, and to move the parameter definition closer to AUTOSAR modeling style (to be reused or propagated between abstraction view).

Semantic:

none

Connections

Connector	**Source**	**Target**
Association Source -> Destination	HardwarePort	HwCategory
Aggregation Source -> Destination	HwAttributeDefinition	HwCategory
Association Source -> Destination	HardwareDescriptionEntity	HwCategory
Association Source -> Destination	HardwarePin	HwCategory
Association Source -> Destination	HardwareComponentType	HwCategory

9.2.1.1.5 Class SiUnit

Element Base Classes:

Element Notes:

This is SiUnit class represent 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

Semantic:

Defined by SiUnit.

Connections

Connector	**Source**	**Target**
Association Source -> Destination	HwAttributeDefinition	SiUnit

Attributes

Attribute	Notes	Default
factorSiToUnit Float	This is the factor for the conversion from and to siUnits.	
offsetSiToUnit Float	This is the offset for the conversion from and to siUnits.	

9.2.1.2 Package HwComponentBehavior

Package Notes:

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.

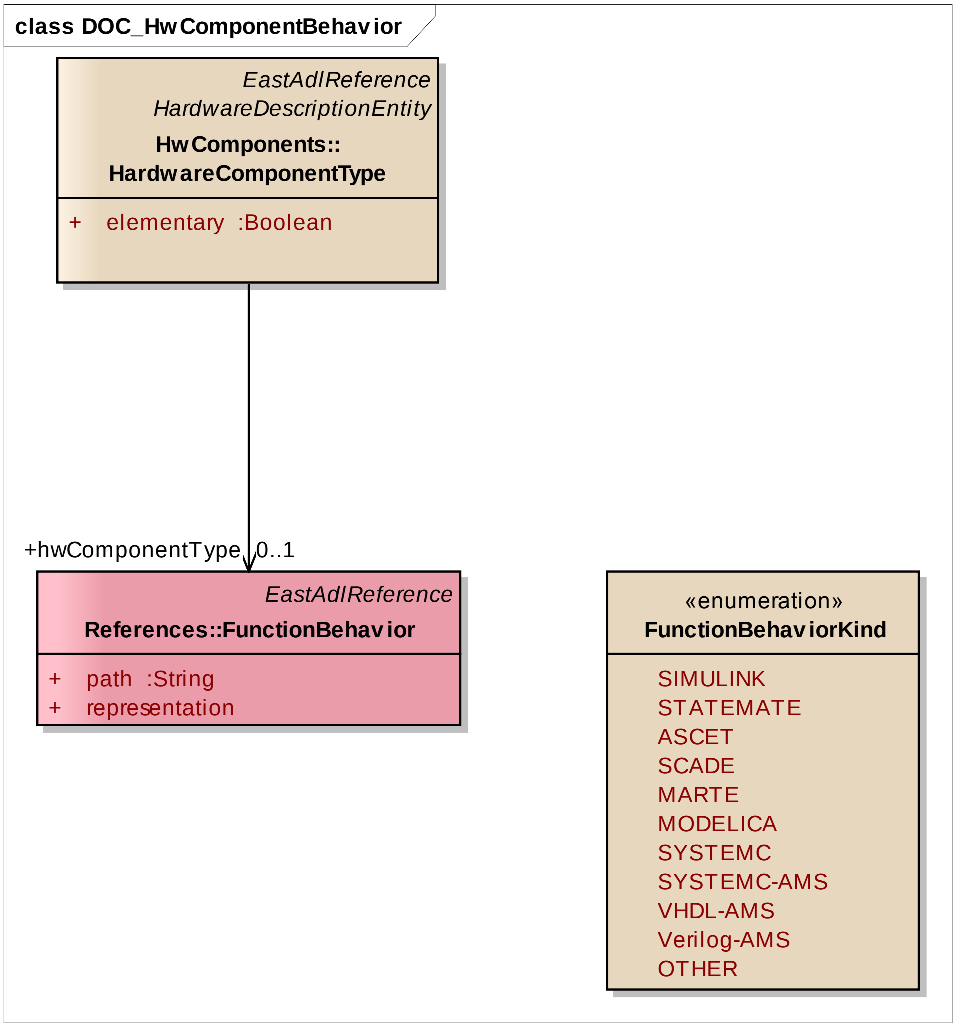

Figure 3: **DOC_HwComponentBehavior** - *(Class diagram)*

Diagram Notes:

This diagram shows the relation of HardwareComponentType with a FunctionBehavior to map the behavior of the hardware compo a function.

9.2.1.2.1 *Enumeration FunctionBehaviorKind*

Element Base Classes:

Element Notes:

FunctionBehaviorKind is an enumeration which lists the various representations used to describe a FunctionBehavior. It is used as a property of a FunctionBehavior. Hardware modeling language 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.

Semantics:

It should be noted that though one can use several languages to provide a representation of a FunctionBehavior, the semantics shall remain compliant with the overall EAST-ADL execution semantics (at least at the port a pin interface).

Extension:

Enumeration, no extension.

Attributes

Attribute	Notes	Default
SIMULINK		
STATEMATE		
ASCET		
SCADE		
MARTE		
MODELICA		
SYSTEMC		
SYSTEMC-AMS		
VHDL-AMS		

Attribute	Notes	Default
Verilog-AMS		
OTHER		

9.2.1.3 Package HwComponents

Package Notes:

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

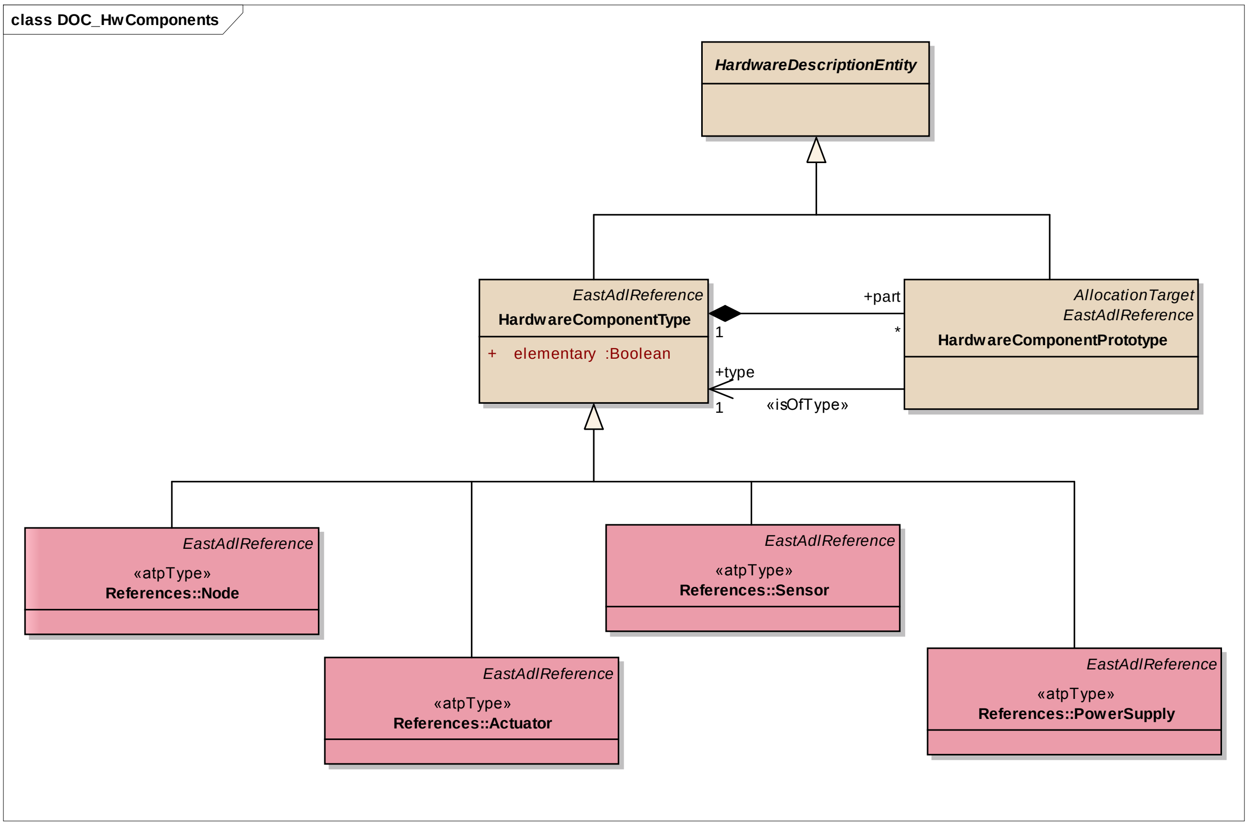

Figure 4: **DOC_HwComponents** - *(Class diagram)*

Diagram Notes:

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.

9.2.1.3.1 Class HardwareDescriptionEntity

Element Base Classes:

Element Notes:

This abstract class describes any hardware entity for further use.

Semantic:

none

Connections

Connector	Source	Target
Generalization Source -> Destination	HardwareComponentType	HardwareDescriptionEntity
Generalization Source -> Destination	HardwarePort	HardwareDescriptionEntity
Generalization Source -> Destination	HardwarePinConnector	HardwareDescriptionEntity
Generalization Source -> Destination	HwPortConnector	HardwareDescriptionEntity
Generalization Source -> Destination	HardwareComponentPrototype	HardwareDescriptionEntity
Generalization Source -> Destination	HardwarePin	HardwareDescriptionEntity
Association Source -> Destination	HardwareDescriptionEntity	HwCategory
Aggregation Source -> Destination	HwAttributeValue	HardwareDescriptionEntity

9.2.1.3.2 Class HardwareComponentPrototype

Element Base Classes: **AllocationTarget, EastAdlReference, HardwareDescriptionEntity**

Element Notes:

Appears as part of a HardwareComponentType and is itself typed by a 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.

Semantics:

The HardwareComponentPrototype represents an occurrence of a hardware element, according to the type of the HardwareComponentPrototype.

Notation:

It shall be shown in the same style as the class specified as type, however it shall be clear that this is a part.

Extension: Property

Connections

Connector	Source	Target
Association Source -> Destination	HWFailureModeInstanceRef	HardwareComponentPrototype
Association Source -> Destination	HardwareComponentPrototype	HardwareComponentPrototype
Association Source -> Destination	HardwareComponentPrototypeInstanceRef	HardwareComponentPrototype
Association Source -> Destination	HWFailureAnalysis	HardwareComponentPrototype
Association Source -> Destination	HardwareComponentPrototypeInstanceRef	HardwareComponentPrototype
Association Source -> Destination	HardwarePinInHardwareTypeHwAbstrRef	HardwareComponentPrototype
Association Source -> Destination	HwComponentInstanceRef	HardwareComponentPrototype
Association Source -> Destination	HardwareComponentPrototype	HardwareComponentType
Association Source -> Destination	HwPortInComponentInstanceRef	HardwareComponentPrototype
Association Source -> Destination	HwPinInHwComponentInstanceRef	HardwareComponentPrototype
Association Source -> Destination	HwComponentInstanceRef	HardwareComponentPrototype
Aggregation Source -> Destination	HardwareComponentType	HardwareComponentPrototype
Generalization Source -> Destination	HardwareComponentPrototype	HardwareDescriptionEntity
Aggregation Source -> Destination	HwAttributeValue	HardwareComponentPrototype

© 2012 The SAFE Consortium

Connector	Source	Target
Generalization Source -> Destination	HardwareComponentPrototype	AllocationTarget
Dependency Source -> Destination	Installation	HardwareComponentPrototype
Generalization Source -> Destination	HardwareComponentPrototype	EastAdlReference
Association Source -> Destination	HardwareComponentPrototype	HWSafetyGoalRelated

9.2.1.3.3 Class HardwareComponentType

Element Base Classes: **EastAdlReference, HardwareDescriptionEntity**

Element Notes:

The HardwareComponentType represents hardware element on an abstract level, allowing preliminary engineering activities related to hardware.

Once hardware and software architecture split is decided, it allows representing hardware element including behavior. This is the starting point for hardware architecture element for exploration/optimization and then restarts the electronic design.

Semantics:

The HardwareComponentType is a structural entity that defines a part of an electrical architecture. Through its ports or pins it can be connected to electrical sources and sinks. It is logical behavior, the transfer function, may be defined in a HardwareFunctionType referencing the HardwareComponentType. This is typically connected through its ports to the environment model to participate in the end-to-end behavioral definition of a function.

Extension:

Class

Connections

Connector	Source	Target
Association Source -> Destination	HWFailureModeInstanceRef	HardwareComponentType
NoteLink Source -> Destination	<anonymous>	HardwareComponentType
Generalization Source -> Destination	HardwareComponentType	EastAdlReference
Generalization Source -> Destination	Cable	HardwareComponentType
NoteLink	<anonymous>	HardwareComponentType

Connector	**Source**	**Target**
Source -> Destination		
Association Source -> Destination	HwPinInHwComponentInstanceRef	HardwareComponentType
Aggregation Source -> Destination	HwPortConnector	HardwareComponentType
Aggregation Source -> Destination	HardwarePinConnector	HardwareComponentType
Association Source -> Destination	HwComponentInstanceRef	HardwareComponentType
Aggregation Source -> Destination	HardwareComponentType	HardwarePort
Association Source -> Destination	HwPortInComponentInstanceRef	HardwareComponentType
Aggregation Source -> Destination	HardwareComponentType	HardwarePin
Generalization Source -> Destination	Actuator	HardwareComponentType
Generalization Source -> Destination	PowerSupply	HardwareComponentType
Generalization Source -> Destination	Node	HardwareComponentType
Association Source -> Destination	HardwareComponentPrototype	HardwareComponentType
Aggregation Source -> Destination	HardwareComponentType	HardwareComponentPrototype
Generalization Source -> Destination	Sensor	HardwareComponentType
Generalization Source -> Destination	HardwareComponentType	HardwareDescriptionEntity
Association Source -> Destination	HardwareComponentType	HwCategory
Association Destination -> Source	FunctionBehavior	HardwareComponentType
Association Source -> Destination	HardwareComponentType	HardwareComponentType

© 2012 The SAFE Consortium

Connector	Source	Target
Aggregation Source -> Destination	LogicalBus	HardwareComponentType
Association Source -> Destination	HardwareComponentFailureExtension	HardwareComponentType
Association Source -> Destination	HardwarePinInHardwareTypeHwAbstrRef	HardwareComponentType
Generalization Source -> Destination	MechanicalComponent	HardwareComponentType

Attributes

Attribute	Notes	Default
elementary Boolean	This parameter is used to define if a hardware component is further decomposed with parts.	

9.2.1.4 Package HwComponentsAndPorts

Package Notes:

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.

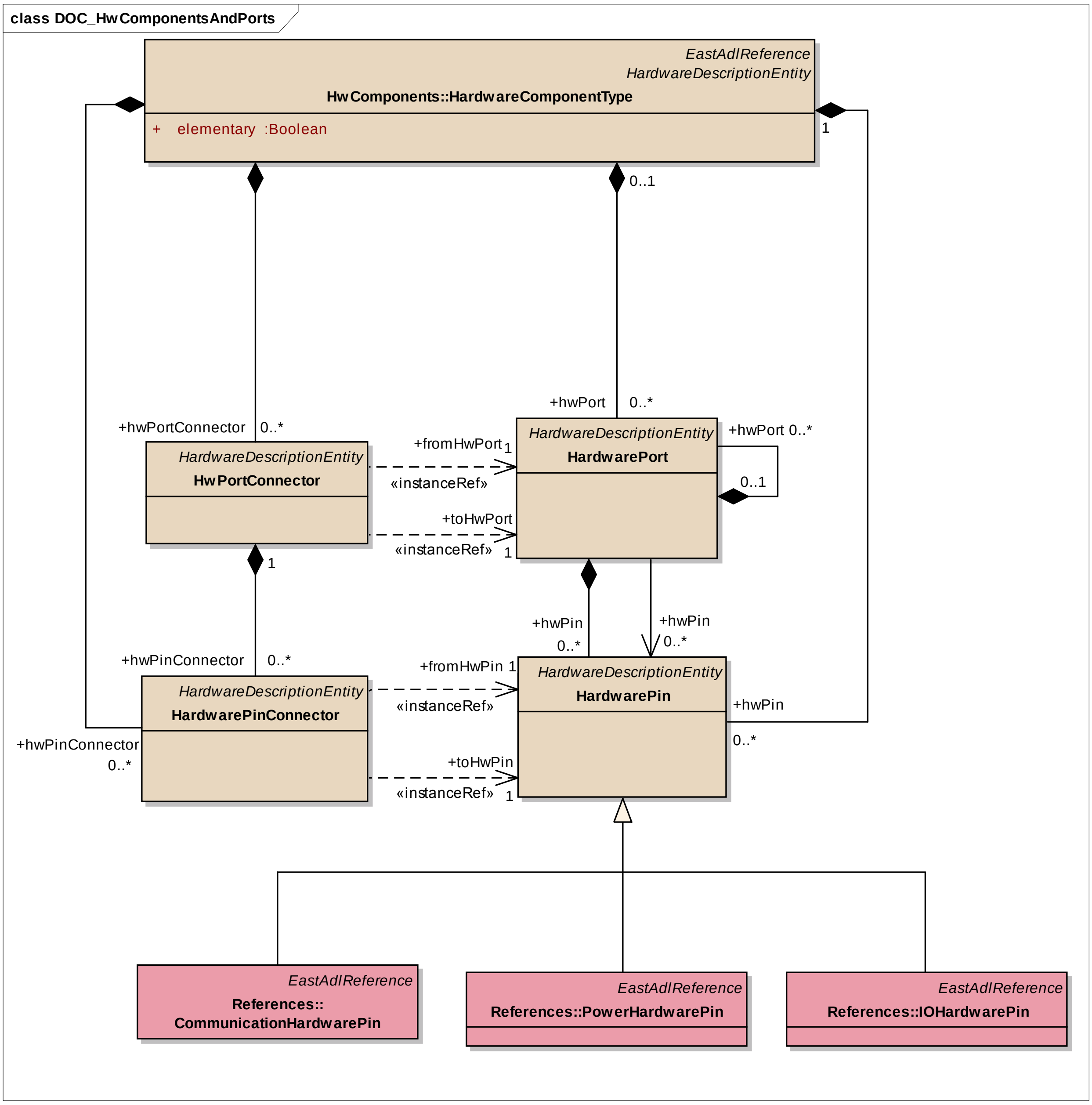

Figure 5: **DOC_HwComponentsAndPorts** - *(Class diagram)*

Diagram Notes:

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.

9.2.1.4.1 Class HwPortConnector

Element Base Classes: **HardwareDescriptionEntity**

Element Notes:

Hardware Port Connector connectors represent port wires that electrically connects the hardware components through its ports.

Semantics:

The connector joins the two referenced ports electrically.

Extension:

Connector

Connections

Connector	Source	Target
Dependency Source -> Destination	HwPortConnector	HardwarePort
Aggregation Source -> Destination	HwPortInComponentInstanceRef	HwPortConnector
Dependency Source -> Destination	HwPortConnector	HardwarePort
Aggregation Source -> Destination	HwPortConnector	HardwarePinConnector
Aggregation Source -> Destination	HwPortConnector	HardwareComponentType
Generalization Source -> Destination	HwPortConnector	HardwareDescriptionEntity

9.2.1.4.2 Class HardwarePinConnector

Element Base Classes: **HardwareDescriptionEntity**

Element Notes:

Hardware Pin Connector connectors represent wires that electrically connect the hardware components through its pins.

Semantics:

The connector joins the two referenced pins electrically.

Extension:

Connector

Connections

Connector	Source	Target
Association Source -> Destination	HardwareConnector	HardwarePinConnector
Aggregation Source -> Destination	HwPortConnector	HardwarePinConnector

Connector	Source	Target
Aggregation Source -> Destination	HwPinInHwComponentInstanceRef	HardwarePinConnector
Dependency Source -> Destination	HardwarePinConnector	HardwarePin
Aggregation Source -> Destination	HardwarePinConnector	HardwareComponentType
Generalization Source -> Destination	HardwarePinConnector	HardwareDescriptionEntity
Association Source -> Destination	LogicalBus	HardwarePinConnector
Dependency Source -> Destination	HardwarePinConnector	HardwarePin

9.2.1.4.3 *Class HardwarePin*

Element Base Classes: **HardwareDescriptionEntity**

Element Notes:

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.

Semantics:

Hardware pin represents an electrical connection point.

Extension:

Port

Connections

Connector	Source	Target
Association Source -> Destination	HardwarePin	HardwarePin
Association Source -> Destination	HardwarePinInHardwareTypeHwAbstrRef	HardwarePin
Generalization Source -> Destination	PowerHardwarePin	HardwarePin
Generalization Source -> Destination	IOHardwarePin	HardwarePin

© 2012 The SAFE Consortium

Connector	Source	Target
Association Source -> Destination	HwPinInHwComponentInstanceRef	HardwarePin
Dependency Source -> Destination	HwAbstractionFunction	HardwarePin
Aggregation Source -> Destination	HardwarePin	HardwarePort
Association Source -> Destination	HardwarePort	HardwarePin
Generalization Source -> Destination	CommunicationHardwarePin	HardwarePin
Dependency Source -> Destination	HardwarePinConnector	HardwarePin
Aggregation Source -> Destination	HardwareComponentType	HardwarePin
Generalization Source -> Destination	HardwarePin	HardwareDescriptionEntity
Association Source -> Destination	HardwarePin	HwCategory
Aggregation Source -> Destination	HwAttributeValue	HardwarePin
Dependency Source -> Destination	HardwarePinConnector	HardwarePin

9.2.1.4.4 Class HardwarePort

Element Base Classes: **HardwareDescriptionEntity**

Element Notes:

The HardwarePort provides means to organize hardware pins by composing HwPin. HardwarePort can be connected by HwPortConnector. It can be used to define external/internal communication bus down to the level of communication transactor for hardware bus.

Notice that a HardwarePort can be also compose HardwarePort for larger representation or abstraction (e.g. address/data/control by a simple transaction).

There are two objectives

1) Abstraction of hardware pin(s), and definition of internal/external communication bus

2) Visualization: schematic entry tools - busses, like address, data, control bus

Semantics:

A HardwarePort is a composition HwPin. 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.

Extension:

Class

Connections

Connector	Source	Target
Dependency Source -> Destination	HwPortConnector	HardwarePort
Aggregation Source -> Destination	HardwarePin	HardwarePort
Association Source -> Destination	HwPortInComponentInstanceRef	HardwarePort
Association Source -> Destination	HardwarePort	HardwarePin
Aggregation Source -> Destination	HardwarePort	HardwarePort
Dependency Source -> Destination	HwPortConnector	HardwarePort
Aggregation Source -> Destination	HardwareComponentType	HardwarePort
Generalization Source -> Destination	HardwarePort	HardwareDescriptionEntity
Association Source -> Destination	HardwarePort	HwCategory
Aggregation Source -> Destination	HwAttributeValue	HardwarePort

9.2.1.5 Package HwSwInterface

Package Notes:

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.

© 2012 The SAFE Consortium

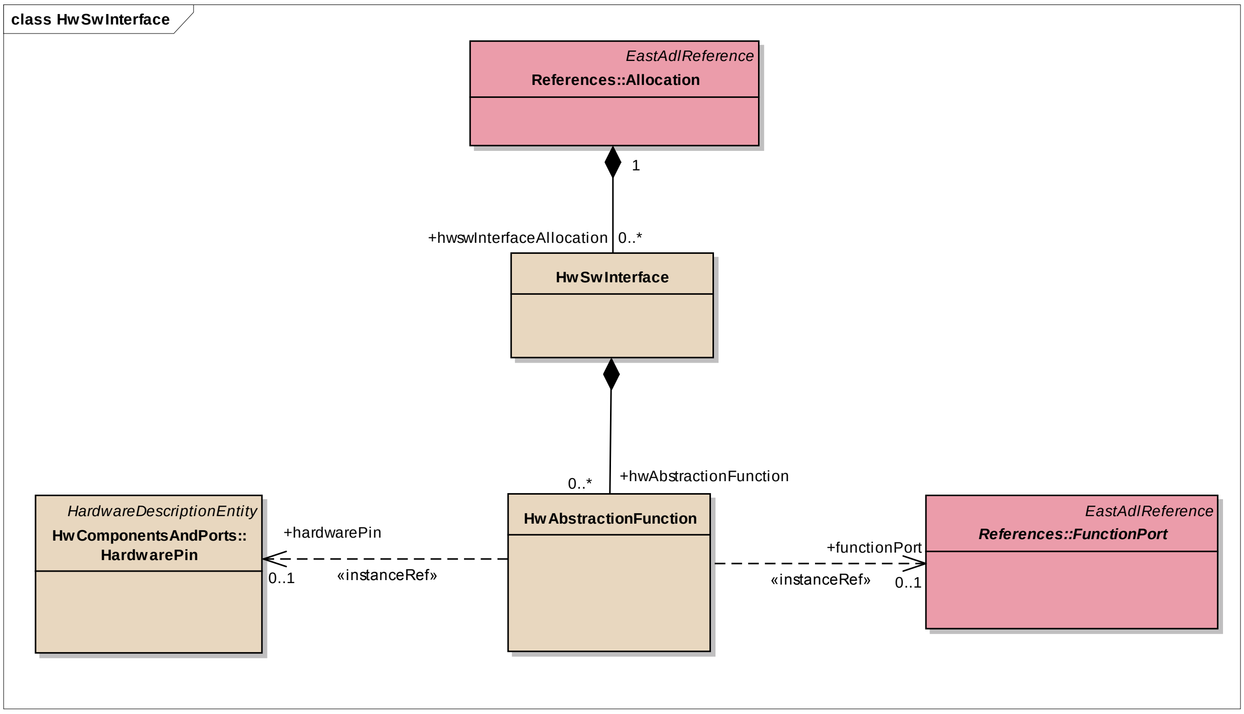

Figure 6: **HwSwInterface** - *(Class diagram)*

Diagram Notes:

This class diagram represents the definition of the HwSwInterface. A software element is represented by a DesignFunction and a hardware element by a HardwareComponent.

9.2.1.5.1 Class HwAbstractionFunction

Element Base Classes:

Element Notes:

The HwAbstractionFunction relates one HardwarePin with one FunctionPort.

This class represents the precise interface between a FunctionPort of DesignFunctionType defined as software element and a HardwarePin of a HardwareComponentType of a hardware. The two interfaces are from heterogeneous domain, so HwAbstraction is a construct that allows making this relation. This class defines 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).

Semantic:

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.

© 2012 The SAFE Consortium

Connections

Connector	Source	Target
Aggregation Source -> Destination	HardwarePinInHardwareTypeHwAbstrRef	HwAbstractionFunction
Aggregation Source -> Destination	FunctionPortInFunctionTypeHwAbstrRef	HwAbstractionFunction
Aggregation Source -> Destination	HwAbstractionFunction	HwSwInterface
Dependency Source -> Destination	HwAbstractionFunction	HardwarePin
Dependency Source -> Destination	HwAbstractionFunction	FunctionType
Dependency Source -> Destination	HwAbstractionFunction	FunctionPort

9.2.1.5.2 Class HwSwInterface

Element Base Classes:

Element Notes:

This class represents the HW-SW interface on the EAST-ADL abstraction Level "Design Level". This element is composed by a HwAbstractionFunction that allow defining precise interface between hardware and software element of the architecture. The hardware architecture is represented by HardwareComponentType and software architecture by DesignFunctionType. As these two elements have heterogeneous interface, as FunctionPort and HardwarePin as dedicated construct was necessary to represent this inter-relation.

The HwSwInterface element is contained into Allocation elements that originally bundles all functionAllocations, and now bundle the Hw-SwInterface elements. HwSwInterface is capable to independent of implementation but allocated into a dedicated hardware element for application purpose (build from HwSwInterface abstraction principle)

Semantic:

By itself, the HwSwInterface has no specific semantic. The semantic is hold by the HwAbstractionFunction.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HwAbstractionFunction	HwSwInterface
Aggregation Source -> Destination	HwSwInterface	Allocation

9.2.1.6 Package _instanceRef

Package Notes:

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

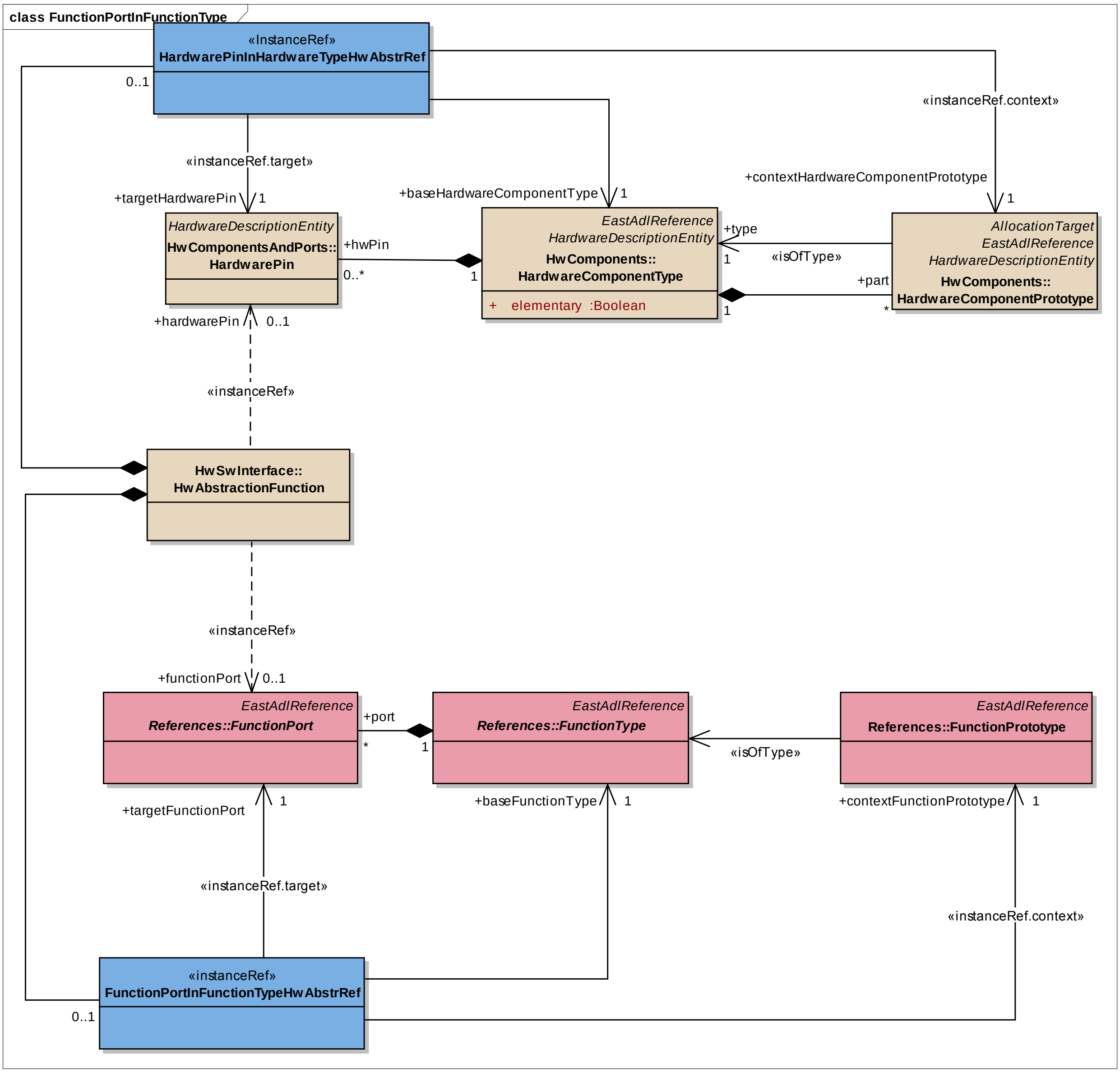

Figure 7: **FunctionPortInFunctionType** - *(Class diagram)*

Diagram Notes:

This class diagram represents the definition of the instanceRef target, base and context for FunctionPort and HardwarePin in the use of HwAbstractionFunction.

© 2012 The SAFE Consortium

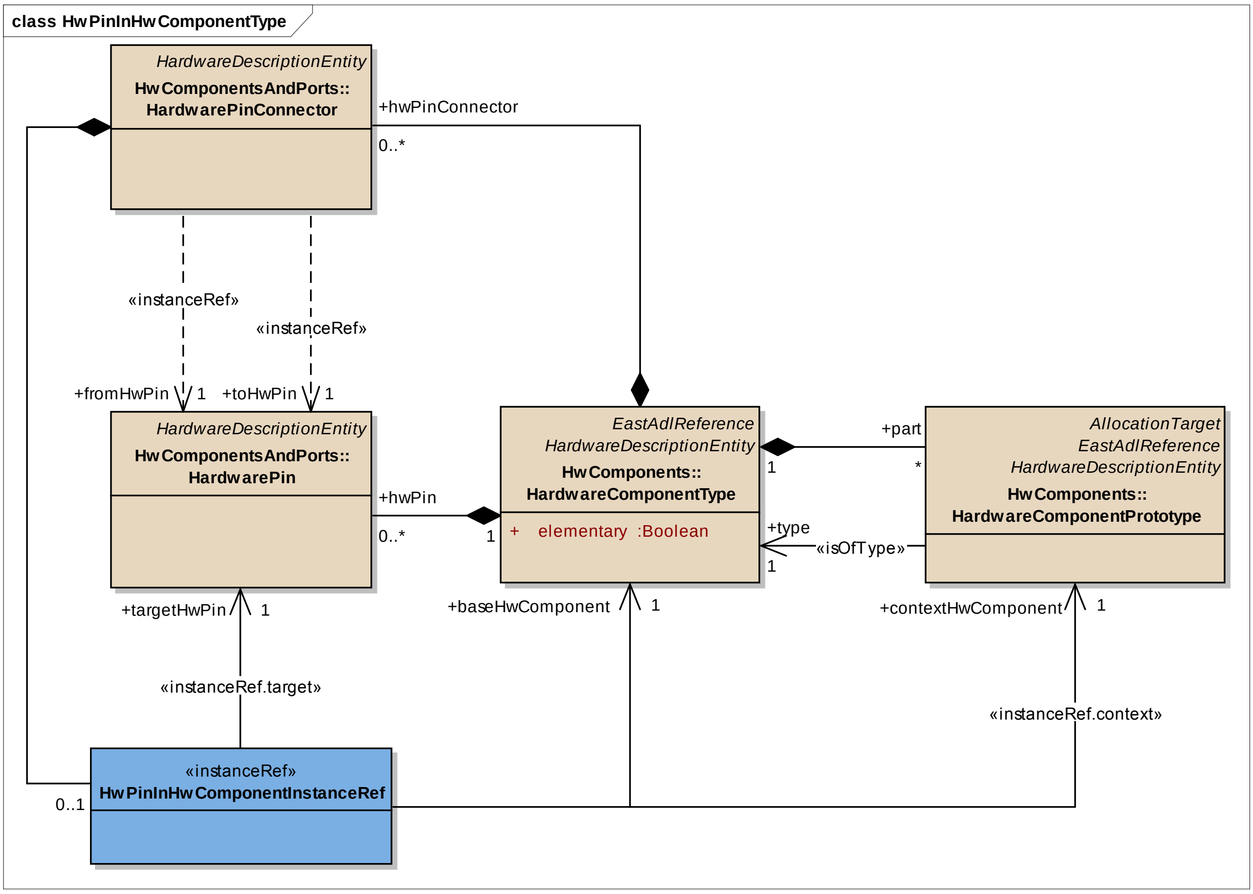

Figure 8: **HwPinInHwComponentType** - *(Class diagram)*

Diagram Notes:

This class diagram represents the definition of the instanceRef target, base and context for HardwarePin in the use of HardwarePinConnector.

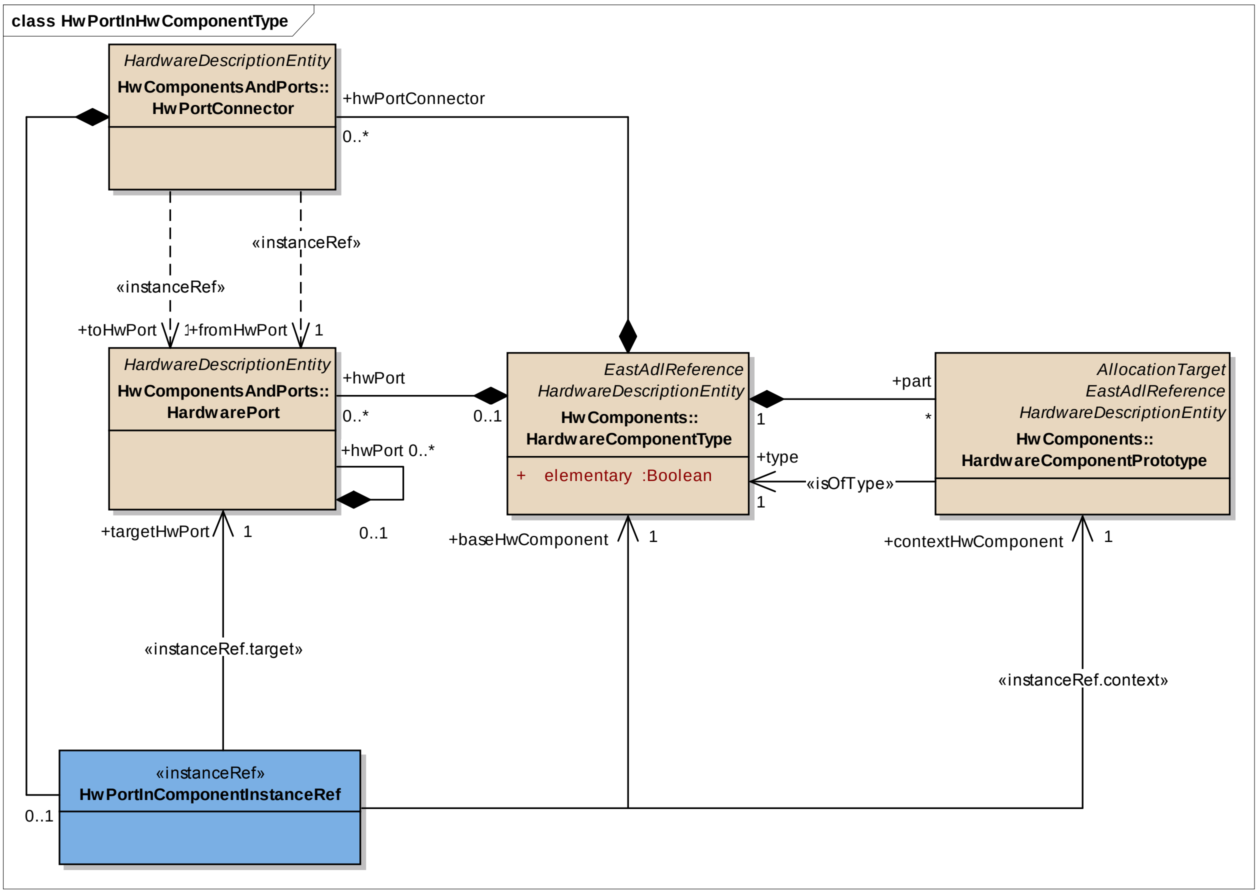

Figure 9: **HwPortInHwComponentType** - *(Class diagram)*

Diagram Notes:

This class diagram represents the definition of the instanceRef target, base and context for HardwarePort in the use of HardwarePortConnector.

9.2.1.6.1 *Class FunctionPortInFunctionTypeHwAbstrRef*

Element Base Classes:

Element Notes:

This "instanceRef" meta-class is the container for holding the relation of HardwarePin in context of HardwareComponentType for the use of HwAbstractionFunction (from HwSwInterface).

Connections

Connector	Source	Target
Aggregation Source -> Destination	FunctionPortInFunctionTypeHwAbstrRef	HwAbstractionFunction
Association Source -> Destination	FunctionPortInFunctionTypeHwAbstrRef	FunctionPrototype

Connector	Source	Target
Association Source -> Destination	FunctionPortInFunctionTypeHwAbstrRef	FunctionType
Association Source -> Destination	FunctionPortInFunctionTypeHwAbstrRef	FunctionPort

9.2.1.6.2 *Class HardwarePinInHardwareTypeHwAbstrRef*

Element Base Classes:

Element Notes:

This "instanceRef" meta-class is the container for the holding the relation of FunctionPort in context of FunctionType for the use of HwAbstractionFunction (from HwSwInterface).

Connections

Connector	Source	Target
Aggregation Source -> Destination	HardwarePinInHardwareTypeHwAbstrRef	HwAbstractionFunction
Association Source -> Destination	HardwarePinInHardwareTypeHwAbstrRef	HardwarePin
Association Source -> Destination	HardwarePinInHardwareTypeHwAbstrRef	HardwareComponentPrototype
Association Source -> Destination	HardwarePinInHardwareTypeHwAbstrRef	HardwareComponentType

9.2.1.6.3 *Class HwPinInHwComponentInstanceRef*

Element Base Classes:

Element Notes:

This "instanceRef" meta-class is the container for holding the relation of HardwarePin in context of HardwareComponentType for the use of HardwarePinConnector.

Connections

Connector	Source	Target
Association	HwPinInHwComponentInstanceRef	HardwarePin

Connector	Source	Target
Source -> Destination		
Aggregation Source -> Destination	HwPinInHwComponentInstanceRef	HardwarePinConnector
Association Source -> Destination	HwPinInHwComponentInstanceRef	HardwareComponentType
Association Source -> Destination	HwPinInHwComponentInstanceRef	HardwareComponentPrototype

9.2.1.6.4 *Class HwPortInComponentInstanceRef*

Element Base Classes:

Element Notes:

This "instanceRef" meta-class reference is the container for holding the relation of HardwarePort in context of HardwareComponentType for the use of HardwarePortConnector.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HwPortInComponentInstanceRef	HwPortConnector
Association Source -> Destination	HwPortInComponentInstanceRef	HardwarePort
Association Source -> Destination	HwPortInComponentInstanceRef	HardwareComponentType
Association Source -> Destination	HwPortInComponentInstanceRef	HardwareComponentPrototype

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
- HWQuantitativeMeasure
- HWArchitecturalMetrics
- ProbabilisticMethods
- Traceability

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

Package Notes:

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

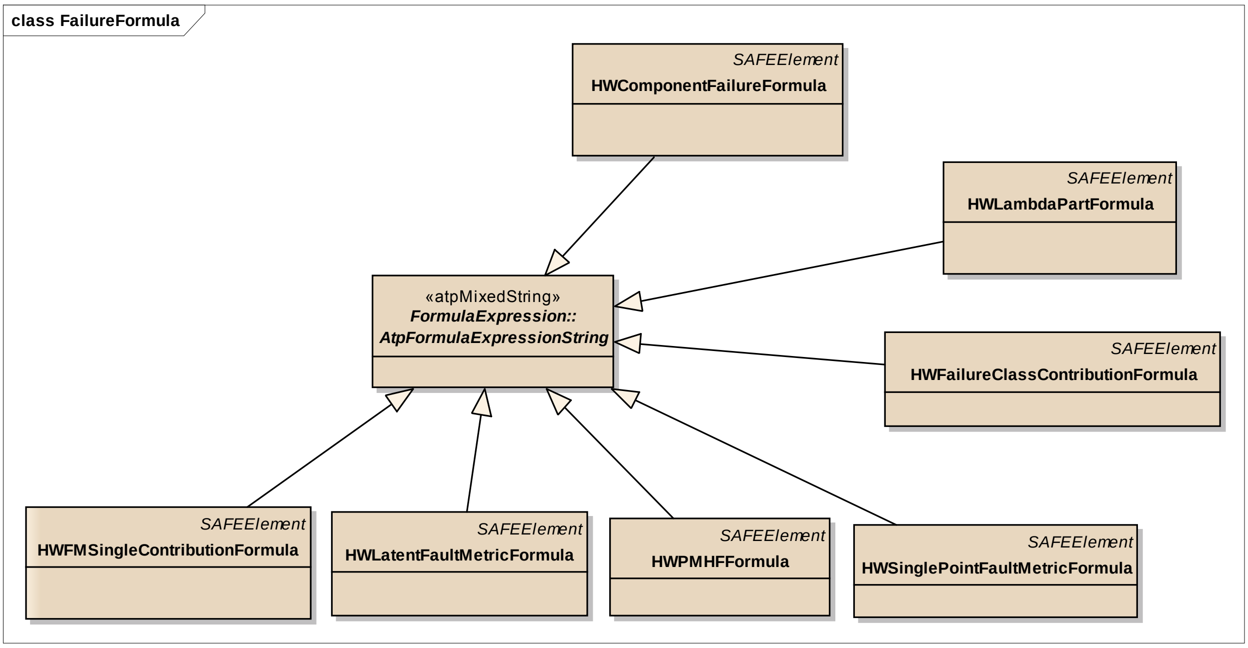

Figure 1: **FailureFormula** - *(Class diagram)*

Diagram Notes:

This diagram shows all formula expressions required for the evaluation of the hardware architecture, all derived from the class AtpFormulaExpressionString.

AtpFormulaExpressionString is derived from AUTOSAR AtpMixedString used to describe calculation formula.

9.3.1.1 Class HWComponentFailureFormula

Element Base Classes: **AtpFormulaExpressionString, SAFEElement**

Element Notes:

This class describes the calculation of the failure rate and its distribution for an HwComponent based on the contribution of all HWPartFailureMode of the AUTOSAR HW Element as hardware Part. The formula expression shall be for each safety-related HWComponent (part of the item).

calculatedValue(HWFailureRate) = Sum[lambdaFailureMode(HWComponentQuantifiedFailureMode)]

calculatedFailureRateDistribution(HWFailureMode) = lambdaFailureMode(HWComponentQuantifiedFailureMode) / calculatedValue(HWFailureRate)

Notes that only Hardware Component safety relevant are considered

Sum[lambdaFailureMode(HWComponentQuantifiedFailureMode)] is performed for each HWComponent.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HWComponentFailureFormula	HWFailureRate
Aggregation Source -> Destination	HWComponentFailureFormula	HWFailureMode
Generalization Source -> Destination	HWComponentFailureFormula	SAFEElement
Generalization Source -> Destination	HWComponentFailureFormula	AtpFormulaExpressionString
Association Source -> Destination	HWComponentFailureFormula	HWComponentQuantifiedFailureMode

9.3.1.2 Class HWFMSingleContributionFormula

Element Base Classes: **AtpFormulaExpressionString, SAFEElement**

Element Notes:

This class describes the individual contribution of an HWFailureMode of an HWComponent to ResidualFault, SinglePointFault or Multiple Fault Latent (in FIT). It is assumed that the HWFailureMode lead to the top level malfunction (link to violation of a SafetyGoal) given by the relation to HWFault connected to a malfunction.

The formula expression shall be for each FailureMode of a safety-related HwComponent (part of the item).

lambdaSafetyComponent = Value(HWFailureRate)

SafetyComponentName = HardwareComponent Class name // to allow detect multiple counting of lambdaSafetyComponent

If (HWFault == Safe)

lambdaSafeFault(HWFMSingleContribution) = [Value(HWFailureRate) * failureRateDistribution(HWFailureMode)]

Else

lambdaSafeFault(HWFMSingleContribution) = 0

Endif

If (HWFault == SPF) lambdaSinglePointFault(HWFMSingleContribution) = [Value(HWFailureRate) * failureRateDistribution(HWFailureMode)]

Else lambdaSinglePointFault(HWFMSingleContribution) = 0

Endif

If (HWFault == MPF)

If (HWSafetyMechanism covers the FailureMode) //residual Fault as HWFailureMode.HWSafetyMechanism = null lambdaResidualFault(HWFMSingleContribution) = [Value(HWFailureRate) * failureRateDistribution(HWFailureMode)] * [hwDiagnosisCoverageRF(HWSafetyMechanism/100]

lambdaMultiplePointFaultLatent(HWFMSingleContribution) = [Value(HWFailureRate) * failureRateDistribution(HWFailureMode) * hwDiagnosisCoverageRF(HWSafetyMechanism)] * [(1 - hwDiagnosisCoverageLF(HWSafetyMechanism)/100)]

Else

lambdaResidualFault(HWFMSingleContribution) = 0

lambdaMultiplePointFaultLatent(HWFMSingleContribution) = [Value(HWFailureRate) * failureRateDistribution(HWFailureMode)]

Endif

Notes that value(HWFailureRate) and failureRateDistribition(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 calculated and estimated value is a tool feature that allow first a calculation for estimation based on allocation field of failure rate and distribution.

Connections

Connector	**Source**	**Target**
Association Source -> Destination	HWFMSingleContributionFormula	HWFailureRate
Aggregation	HWFMSingleContributionFormula	HWFMSingleContribution

Connector	**Source**	**Target**
Source -> Destination		
Generalization Source -> Destination	HWFMSingleContributionFormula	AtpFormulaExpressionString
Generalization Source -> Destination	HWFMSingleContributionFormula	SAFEElement
Association Source -> Destination	HWFMSingleContributionFormula	HWFailureMode
Association Source -> Destination	HWFMSingleContributionFormula	HWFault
Association Source -> Destination	HWFMSingleContributionFormula	HWSafetyMechanism

9.3.1.3 Class HWFailureClassContributionFormula

Element Base Classes: **AtpFormulaExpressionString, SAFEElement**

Element Notes:

This class describes the calculation of the diagnostic coverage of a HWElement (from HWComponent) for the Failure Rate Class method (in %) as ratio of all fault coverage of the HW Component (safe fault, single-point fault and residual fault) and the calculation of the element FailureRateClass defined by its failure rate.

The formula expression shall be calculated for each FailureMode of a safety-related HwComponent (part of the item).

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))

The formula expression shall be for each the top level malfunction (link to violation of the SafetyGoal).

HWElementFailureRateClass(HWElementFailureClass) = HWValueRateClassEnum(LambdaSafetyComponent)

HWElementFailureRateClass(HWElementResidualDiagnosisCoverage) = { 1 - [(Sum (lambdaSinglePointFault(HWFMSingleContribution) + lambdaResidualFault(HWFMSingleContribution)) / LambdaSafetyComponent] } * 100

HWElementFailureRateClass(HWElementLatentDiagnosisCoverage) = { 1 - [Sum (lambdaMultipleFaultLatent(HWFMSingleContribution) / [LambdaSafetyComponent - Sum (lambdaSinglePointFault(HWFMSingleContribution) + lambdaResidualFault(HWFMSingleContribution)]] } * 100

Note that Value(hwElementDiagnosisCoverage) is applied on estimatedValue from electronic design level to perform the final calculation and verification of the individual HWElement FailureRateClass and ElementDiagnosisCoverage. 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).

Connections

Connector	Source	Target
Generalization Source -> Destination	HWFailureClassContributionFormula	AtpFormulaExpressionString
Association Source -> Destination	HWFailureClassContributionFormula	HWFMSingleContribution
Generalization Source -> Destination	HWFailureClassContributionFormula	SAFEElement
Aggregation Source -> Destination	HWFailureClassContributionFormula	HWElementFailureRateClass

9.3.1.4 Class HWLambdaPartFormula

Element Base Classes: **AtpFormulaExpressionString, SAFEElement**

Element Notes:

This class describes the lambda failure rate contribution of all HWPartFailureMode of HardwarePart to a dedicated HWFailureMode of an HWComponent.

The formula expression shall be for each HWFailureMode of a Safety Related HWComponent (related as parts of the Item) expressed from the different safety-related AUTOSAR HW Element (part of the item).

lambdaFailureMode = function all HWPartFailureMode [Value(HWPartFailureRate) * FailureRateDistribution(HWPartFailureMode), AutosarHwElement)

Connections

Connector	Source	Target
Aggregation Source -> Destination	HWLambdaPartFormula	HWComponentQuantifiedFailureMode
Association Source -> Destination	HWLambdaPartFormula	HWPartFailureModeInstanceRef

Connector	Source	Target
Generalization Source -> Destination	HWLambdaPartFormula	SAFEElement
Generalization Source -> Destination	HWLambdaPartFormula	AtpFormulaExpressionString
Association Source -> Destination	HWLambdaPartFormula	HWPartFailureRateInstanceRef

9.3.1.5 Class HWLatentFaultMetricFormula

Element Base Classes: **AtpFormulaExpressionString, SAFEElement**

Element Notes:

This class describes the latent fault metric (in %) as ratio of impact of latent faults for a top level malfunction (link to violation of a SafetyGoal).

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))

The formula expression shall be for each SafetyGoal:

Value(MultipleLatentFaultMetric) = { 1 - [Sum (lambdaMultipleFaultLatent(FMSingleContribution) / [Sum(LambdaSafetyComponent) - Sum (lambdaSinglePointFault(FMSingleContribution) + lambdaResidualFault(FMSingleContribution)]] } * 100

Value(MutiplePointFaultMteric) is applied on estimatedValue 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.

Sum(LambdaSafetyComponent) is only counted once for a HWElement (identical safetyComponentClassName).

Connections

Connector	Source	Target
Generalization Source -> Destination	HWLatentFaultMetricFormula	AtpFormulaExpressionString
Aggregation Source -> Destination	HWLatentFaultMetricFormula	HWLatentFaultMetric
Generalization Source -> Destination	HWLatentFaultMetricFormula	SAFEElement
Association Source -> Destination	HWLatentFaultMetricFormula	HWFMSingleContribution

9.3.1.6 Class HWPMHFFormula

Element Base Classes: **AtpFormulaExpressionString, SAFEElement**

Element Notes:

This class describes the individual PMHF (in FIT) as probabilistic evaluation of violation of a top level malfunction (link to violation of a SafetyGoal).

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

The formula expression shall be for each SafetyGoal:

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

Value(HWPMHF) is applied on calculatedValue 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 Components safety relevant are considered.

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

Connections

Connector	Source	Target
Generalization Source -> Destination	HWPMHFFormula	AtpFormulaExpressionString
Association Source -> Destination	HWPMHFFormula	HWFMSingleContribution
Generalization Source -> Destination	HWPMHFFormula	SAFEElement
Aggregation Source -> Destination	HWPMHFFormula	HWPMHF

9.3.1.7 Class HWSinglePointFaultMetricFormula

Element Base Classes: **AtpFormulaExpressionString, SAFEElement**

Element Notes:

This class describes the single-point fault metric (in %) as ratio of impact of single-point and residual faults for a top level malfunction (link to violation of a SafetyGoal).

SPF metric = 100% - total (single point faults failure rate + residual faults failure rate) / total (safety related HWComponent failure rate)

The formula expression shall be for each SafetyGoal:

Value(SinglePointFaultMetric) = { 1 - [(Sum (lambdaSinglePointFault(FMSingleContribution) + lambdaResidualFault(FMSingleContribution)) / Sum(LambdaSafetyComponent)] } * 100

Value(SinglePointFaultMetric) is applied on estimatedValue 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.

Sum(LambdaSafetyComponent) is only counted once for a HWElement (identical safetyComponentClassName).

Connections

Connector	Source	Target
Association Source -> Destination	HWSinglePointFaultMetricFormula	HWFMSingleContribution
Generalization Source -> Destination	HWSinglePointFaultMetricFormula	SAFEElement
Generalization Source -> Destination	HWSinglePointFaultMetricFormula	AtpFormulaExpressionString
Aggregation Source -> Destination	HWSinglePointFaultMetricFormula	HWSinglePointFaultMetric

9.3.2 Package Failure

Package Notes:

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

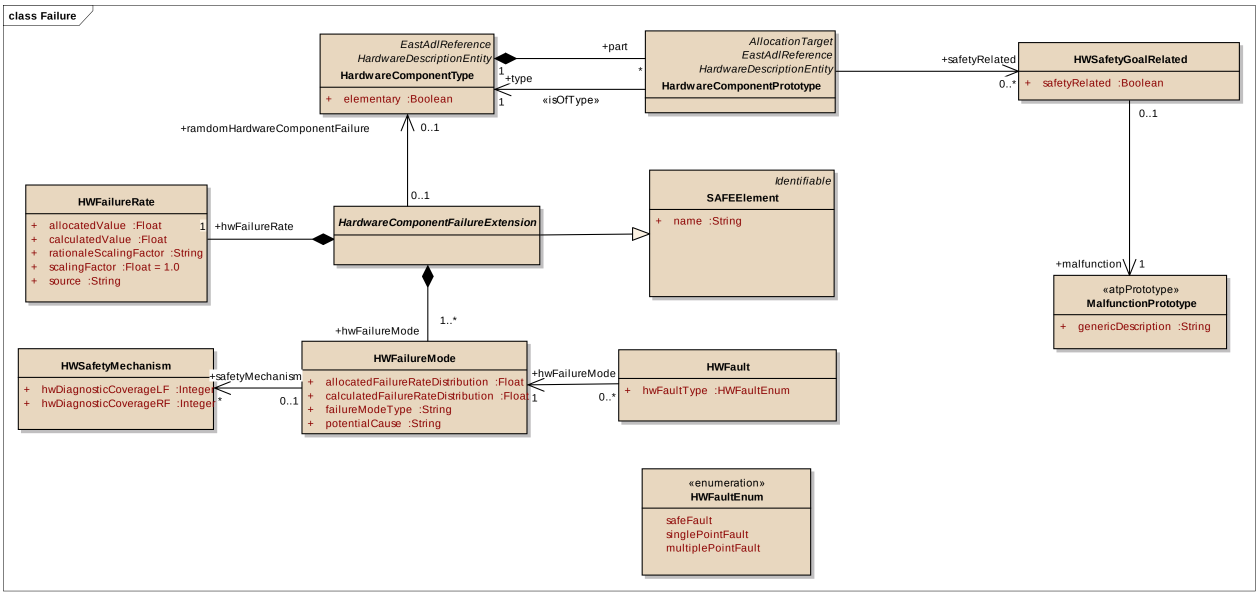

Figure 2: **Failure** - *(Class diagram)*

Diagram Notes:

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

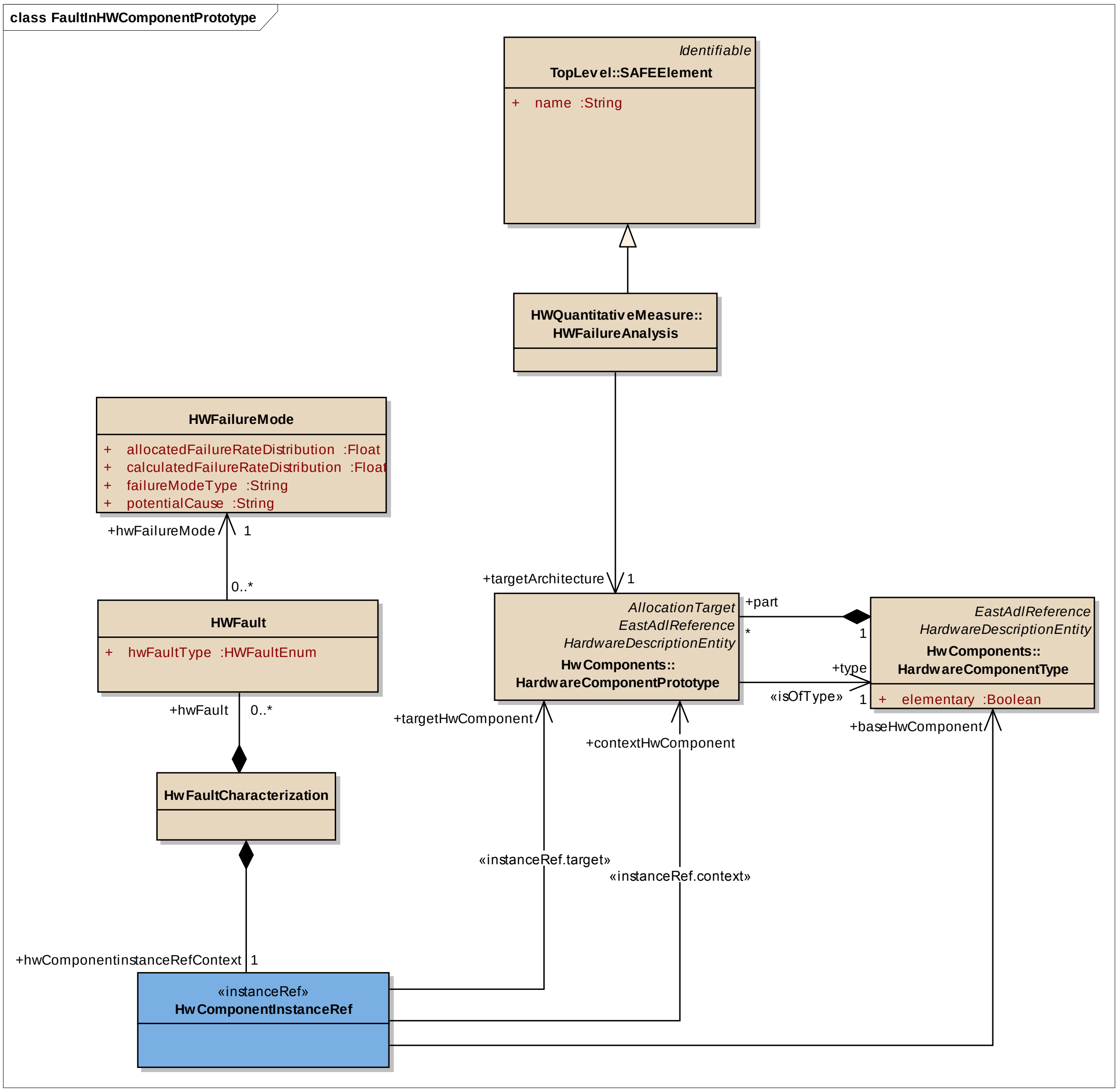

Figure 3: **FaultInHWComponentPrototype** - *(Class diagram)*

Diagram Notes:

This diagram shows the fault and its characterization for an instance of a hardware component.

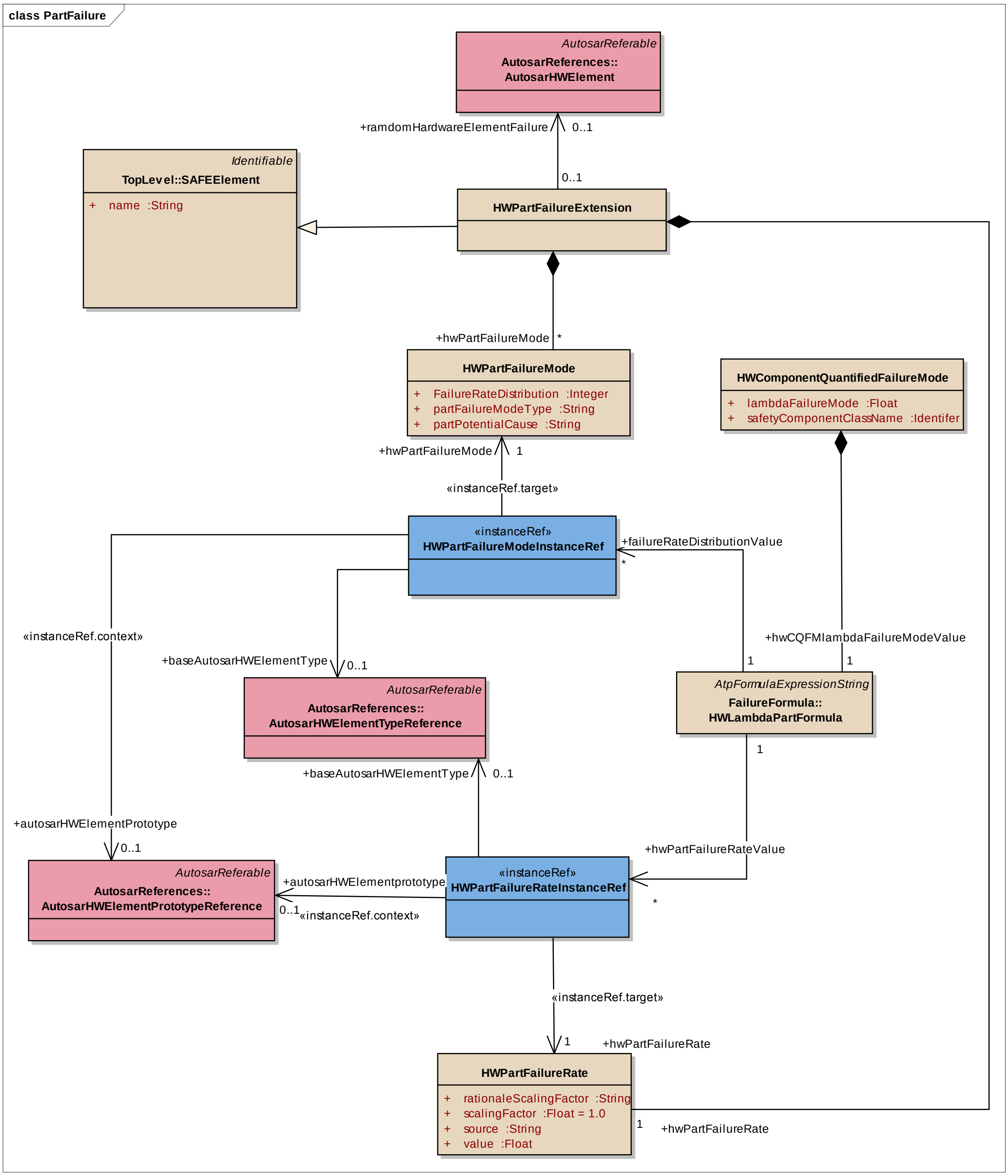

Figure 4: **PartFailure** - *(Class diagram)*

Diagram Notes:

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

© 2012 The SAFE Consortium

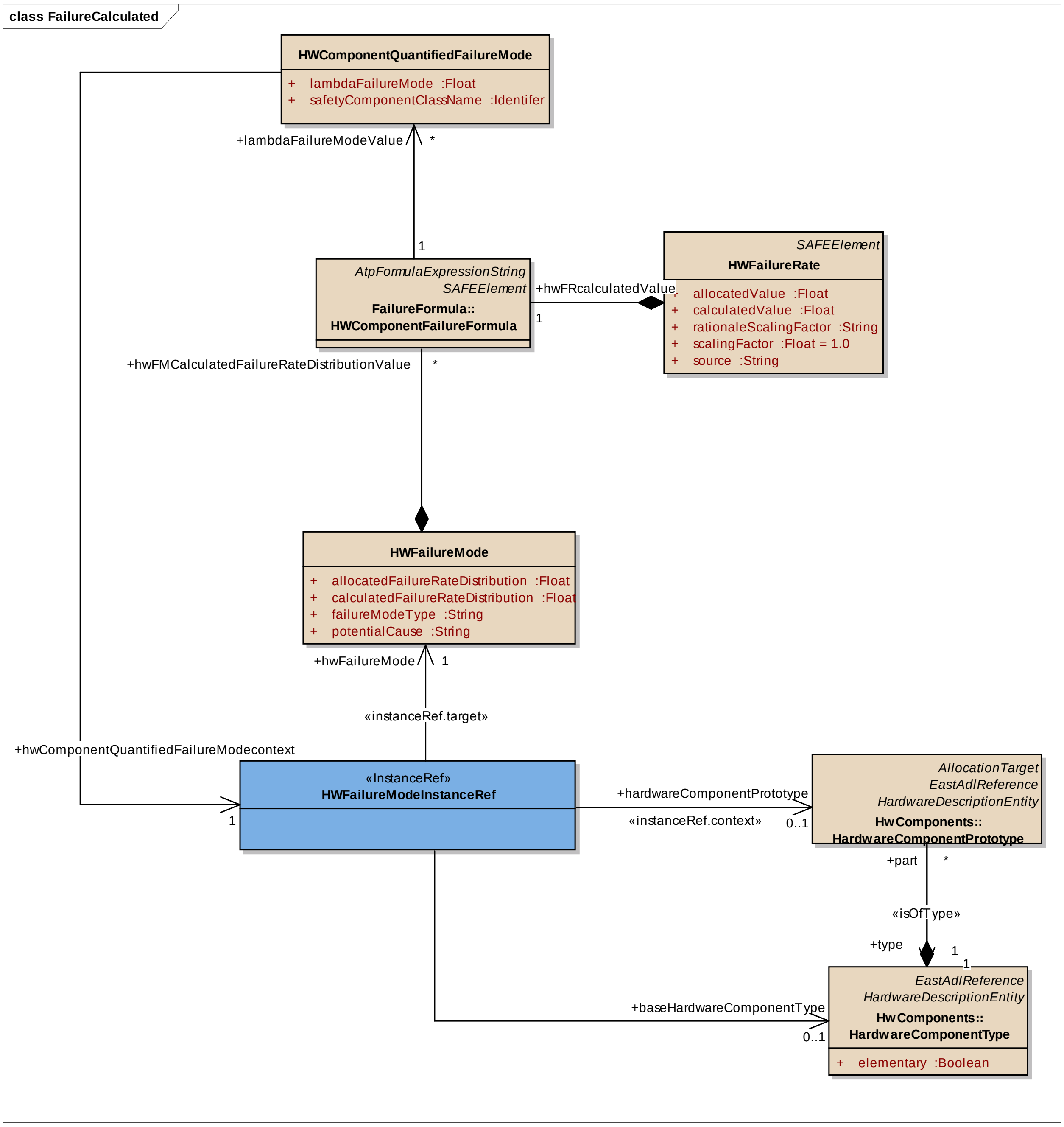

Figure 5: **FailureCalculated** - *(Class diagram)*

Diagram Notes:

This diagram shows the instance reference of a failure mode of a hardware component on higher level and its interference with hardware element part calculations.

9.3.2.1 Class HWComponentQuantifiedFailureMode

Element Base Classes:

Element Notes:

This class describes the quantified failure rate of a failure mode of a HWComponent based on the contribution of each HWPartFailureMode of the related HWPart as AUTOSAR HW Element (calculated with the formula and stored in the attribute lambdaFailureMode).

The attribute SafetyComponentClassName is used to identify the HWComponent Class name for further calculation of all failure mode to the same HWComponent.

A quantified HW ComponentFailureMode must identify the related HWFailureMode of the HWComponent.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HWLambdaPartFormula	HWComponentQuantifiedFailureMode
Association Source -> Destination	HWComponentQuantifiedFailureMode	HWFailureModeInstanceRef
Association Source -> Destination	HWComponentFailureFormula	HWComponentQuantifiedFailureMode

Attributes

Attribute	Notes	Default
lambdaFailureMode Float	This attribute contains the quantified failure rate for the corresponding failure mode of the hardware component.	
safetyComponentClassName Identifer	This attribute stores the name of the hardware component class.	

9.3.2.2 Class HWFailureMode

Element Base Classes:

Element Notes:

This class describes a HWFailureMode of a HWComponent.

Each HWFailureMode of the HWComponent must have its own characterization for each linked malfunction (linked to violation of a SafetyGoal).

The HWFailureMode and HWFailureRateDistribution can be derived from e.g. Industry Source (see ISO Part 5 8.4.3).

Connections

Connector	Source	Target
Aggregation	HWComponentFailureFormula	HWFailureMode

Connector	Source	Target
Source -> Destination		
Association Source -> Destination	HWFMSingleContributionFormula	HWFailureMode
Association Source -> Destination	HWFailureModeInstanceRef	HWFailureMode
Association Source -> Destination	HWFault	HWFailureMode
Association Source -> Destination	HWFailureMode	HWSafetyMechanism
Aggregation Source -> Destination	HWFailureMode	HardwareComponentFailureExtension

Attributes

Attribute	Notes	Default
allocatedFailureRateDistribution Float	This attribute describes the allocated distribution of the failure rate of the specific failure mode (in percentage) of a 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).	
calculatedFailureRateDistribution Float	This attribute describes the distribution of the failure rate given by calculation of HWPart (AUTOSAR HWElement) for a specific failure mode (in percentage) of a 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).	
failureModeType String	This attribute textually describes the type of a failure mode of an HWComponent (e.g. "No value" for a sensor).	
potentialCause String	This attribute allows the documentation of the potential cause of the HWComponent failure mode (e.g. high temperature).	

9.3.2.3 Class HWFailureModeInstanceRef

Element Base Classes:

Element Notes:

This "instanceRef" meta-class is the container for holding the relation of HWComponentQuantifiedFailureMode in context of HardwareComponentType for the use of HWFailureMode.

Connections

Connector	Source	Target
Association Source -> Destination	HWFailureModeInstanceRef	HardwareComponentType
Association Source -> Destination	HWFailureModeInstanceRef	HardwareComponentPrototype
Association Source -> Destination	HWFailureModeInstanceRef	HWFailureMode
Association Source -> Destination	HWComponentQuantifiedFailureMode	HWFailureModeInstanceRef

9.3.2.4 Class HWFailureRate

Element Base Classes: **SAFEElement**

Element Notes:

This class captures the HWFailureRate of an HWComponent.

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.

Connections

Connector	Source	Target
Association Source -> Destination	HWFMSingleContributionFormula	HWFailureRate
Aggregation Source -> Destination	HWComponentFailureFormula	HWFailureRate
Aggregation Source -> Destination	HWFailureRate	HardwareComponentFailureExtension
Generalization Source -> Destination	HWFailureRate	SAFEElement

Attributes

Attribute	Notes	Default
allocatedValue Float	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.	

Attribute	Notes	Default
calculatedValue Float	Calculated failure rate of the HWComponent for architectural verification by architectural metrics and probabilistic value. It shall be expressed in FIT.	
rationaleScalingFactor String	The rationaleScalingFactor shall provide a rationale, if a scaling factor different to 1.0 is applied.	
scalingFactor Float	The scalingFactor allows potential scaling between different sources of failure rates as described in ISO Part 5 Annex F.	1.0
source String	FIT rate source shall documented according to possible source as described in ISO 26262 Part 5 8.4.3: a) failure rate from industry source (IEC/TR 62380, IEC 61709, ...) b) statistic based on return field or test c) Expert judgement	

9.3.2.5 Class HWFault

Element Base Classes: **SAFEElement**

Element Notes:

This class HWFault represents the characterization of an HWComponent Fault defined by tags as Safe Fault, SinglePointFault or MultiplePointFault of a specific FailureMode in a context of a Hardware Architecture.

HardwareFault can only exist for HardwareComponentPrototype when HWComponent are used.

The related malfunction (link to violation of a SafetyGoals) is already linked with the FailureMode of the HardwareComponent via the HWSafetyGoalRelated meta class.

The different values are:

SafeFault: no violation of safety goal

ResidualOrSinglePointFault: direct violation of the SafetyGoal (1st order fault)

MultiplePointFault: violation of the SafetyGoal in combination with an independent failure of another component (minimum 2nd order)

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).

Connections

Connector	Source	Target
Association Source -> Destination	HWFMSingleContributionFormula	HWFault
Association Source -> Destination	HWFault	HWFailureMode

© 2012 The SAFE Consortium

Connector	Source	Target
Aggregation Source -> Destination	HWFault	HwFaultCharacterization
Generalization Source -> Destination	HWFault	SAFEElement

Attributes

Attribute	Notes	Default
hwFaultType HWFaultEnum	Characterization of the Failure Mode for a single related malfunction (linked to violation of a Safety Goal). Possible Types are: • SafeFault (no violation of Safety Goal) • ResidualOrSinglePointFault (direct violation of Safety Goal (either covered by Safety Mechanism or not) • Multiple-Point-Fault (violation of Safety Goal in combination with another independent fault)	

9.3.2.6 Enumeration HWFaultEnum

Element Base Classes: **SAFEElement**

Element Notes:

This enumeration includes the possible characterizations for the attribute hwFaultType in Class HWFault.

For simplification and clarification only SafeFault, SinglePointFault and MultiplePointFault are derived from the ISO Part 5 7.4.3.2.

SinglePointFault represents a first order fault, while multiplePointFault represents a higher order. For the hardware fault description, an cut set order of two is adequate. Therefore, an limited order of two (see ISO Part 5 7.4.3.2) can be defined. This means, that multiplePointFault represents an second order fault (dualPointFault).

The precise characterization of a HWFault (e.g. Multiple-Point-Latent) can be derived from the value of the attribute hwFaultType and a possible existence of a SafetyMechanism.

Connections

Connector	Source	Target
Generalization Source -> Destination	HWFaultEnum	SAFEElement

Attributes

Attribute	Notes	Default
safeFault	This literal describes the characterization as a safe fault.	

Attribute	Notes	Default
singlePointFault	This literal describes the characterization as a single-point of failure (direct violation).	
multiplePointFault	This literal describes the characterization as a multiple-point fault (violation in combination with another independent fault).	

9.3.2.7 Class HWPartFailureExtension

Element Base Classes: **SAFEElement**

Element Notes:

This class describes the failure data extension for all HWPart elements, including part failure rate and part failure mode.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HWPartFailureMode	HWPartFailureExtension
Association Source -> Destination	HWPartFailureExtension	AutosarHWElement
Aggregation Source -> Destination	HWPartFailureRate	HWPartFailureExtension
Generalization Source -> Destination	HWPartFailureExtension	SAFEElement

9.3.2.8 Class HWPartFailureMode

Element Base Classes: **SAFEElement**

Element Notes:

This class describes HWPartFailureModes of an HWPart as AUTOSAR HWElement. It aso captures the potential cause for an HWFailureMode as String (for documentation).

Each HWPartFailureMode of the Autosar HardwareElement must define a relation and contribution to a HWFailureMode of HardwareComponent (from hardware design level).

The HWFailureMode and HWFailureRateDistribution can be derived from e.g. Industry Source.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HWPartFailureMode	HWPartFailureExtension

Connector	Source	Target
Generalization Source -> Destination	HWPartFailureMode	SAFEElement
Association Source -> Destination	HWPartFailureModeInstanceRef	HWPartFailureMode

Attributes

Attribute	Notes	Default
FailureRateDistribution Integer	This attribute describes the distribution of the failure rate of the HWPart element for the specific hardware part failure mode in percentage.	
partFailureModeType String	This attribute textually describes the type of a failure mode of an HWPart element (e.g. "ShortCircuit" for a resistor).	
partPotentialCause String	This attribute allows the documentation of the potential cause of the HWPart failure mode (e.g. high temperature).	

9.3.2.9 Class HWPartFailureModeInstanceRef

Element Base Classes:

Element Notes:

This "instanceRef" meta-class is the container for holding the relation of HWLambadPartFormula in context of AutosarHWElementType for the use of HWPartFailureMode.

Connections

Connector	Source	Target
Association Source -> Destination	HWLambdaPartFormula	HWPartFailureModeInstanceRef
Association Source -> Destination	HWPartFailureModeInstanceRef	AutosarHWElementTypeReference
Association Source -> Destination	HWPartFailureModeInstanceRef	AutosarHWElementPrototypeReference
Association Source -> Destination	HWPartFailureModeInstanceRef	HWPartFailureMode

9.3.2.10 Class HWPartFailureRate

Element Base Classes: **SAFEElement**

Element Notes:

This class captures the HWPartFailureRate of a AUTOSAR HWElement. Each AUTOSAR HWElement has one single Part HWFailureRate.

The appropriate Part FailureRate can be derived from e.g. Industry Source.

Connections

Connector	Source	Target
Generalization Source -> Destination	HWPartFailureRate	SAFEElement
Association Source -> Destination	HWPartFailureRateInstanceRef	HWPartFailureRate
Aggregation Source -> Destination	HWPartFailureRate	HWPartFailureExtension

Attributes

Attribute	Notes	Default
rationaleScalingFactor String	The rationaleScalingFactor shall provide a rationale, if a scaling factor different to 1.0 is applied.	
scalingFactor Float	The scalingFactor allows potential scaling between different sources of failure rates as described in ISO Part 5 Annex F.	1.0
source String	FIT rate source shall documented according to possible source as described in ISO 26262 Part 5 8.4.3: a) failure rate from industry source (IEC/TR 62380, IEC 61709, ...) b) statistic based on return field or test c) Expert judgement	
value Float	FIT rate for the hardware part element. It shall be expressed in FIT.	

9.3.2.11 Class HWPartFailureRateInstanceRef

Element Base Classes:

Element Notes:

This "instanceRef" meta-class is the container for holding the relation of HWLambadPartFormula in context of AutosarHWElementType for the use of HWPartFailureRate.

© 2012 The SAFE Consortium

Connections

Connector	Source	Target
Association Source -> Destination	HWPartFailureRateInstanceRef	AutosarHWElementTypeReference
Association Source -> Destination	HWPartFailureRateInstanceRef	HWPartFailureRate
Association Source -> Destination	HWPartFailureRateInstanceRef	AutosarHWElementPrototypeReferenc e
Association Source -> Destination	HWLambdaPartFormula	HWPartFailureRateInstanceRef

9.3.2.12 Class HWSafetyGoalRelated

Element Base Classes: **SAFEElement**

Element Notes:

This class describes the relation for contribution of a hardware component to a malfunction (link to violation of a SafetyGoal).

Connections

Connector	Source	Target
Association Source -> Destination	HWSafetyGoalRelated	MalfunctionPrototype
Generalization Source -> Destination	HWSafetyGoalRelated	SAFEElement
Association Source -> Destination	HardwareComponentPrototype	HWSafetyGoalRelated

Attributes

Attribute	Notes	Default
safetyRelated Boolean	This attribute stores the contribution of the HWComponent as boolean.	

9.3.2.13 Class HWSafetyMechanism

Element Base Classes: **SAFEElement**

Element Notes:

This class refers to a single SafetyMechanism with its DiagnosticCoverage for ResidualFaults (DiagnosticCoverageRF) and for Multiple-Point-Latent-Faults (DiagnosticCoverageMPF-L).

A SafetyMechanism can be related to one or more FailureModes of one or more HWComponent. A SafetyMechanism is an architectural element (either HWComponent of FunctionDesign).

Connections

Connector	Source	Target
Aggregation Source -> Destination	HWSafetyMechanism	TechnicalSafetyConcept
Aggregation Source -> Destination	HWSafetyMechanism	FunctionalSafetyConcept
Association Source -> Destination	HWFailureMode	HWSafetyMechanism
Association Source -> Destination	HWFMSingleContributionFormula	HWSafetyMechanism
Generalization Source -> Destination	HWSafetyMechanism	SAFEElement

Attributes

Attribute	Notes	Default
hwDiagnosticCoverageLF Integer	This attribute describes the Diagnostic Coverage for Residual-Faults of the Safety Mechanism. It shall be expressed as [%].	
hwDiagnosticCoverageRF Integer	This attribute describes the Diagnostic Coverage for Multiple-Point-Latent-Faults of the Safety Mechanism. It shall be expressed as a [%]	

9.3.2.14 Class HardwareComponentFailureExtension

Element Base Classes: **SAFEElement**

Element Notes:

This class describes the failure data extension for all HWComponents, including failure rate and failure mode.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HWFailureMode	HardwareComponentFailureExtension
Aggregation Source -> Destination	HWFailureRate	HardwareComponentFailureExtension

Connector	Source	Target
Association Source -> Destination	HardwareComponentFailureExtension	HardwareComponentType
Generalization Source -> Destination	HardwareComponentFailureExtension	SAFEElement

9.3.2.15 Class HwComponentInstanceRef

Element Base Classes: **SAFEElement**

Element Notes:

This "instanceRef" meta-class is the container for holding the relation of HWFaultCharacterisation in context of HWComponentType for the use of HWComponentPrototype.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HwComponentInstanceRef	HwFaultCharacterization
Association Source -> Destination	HwComponentInstanceRef	HardwareComponentPrototype
Association Source -> Destination	HwComponentInstanceRef	HardwareComponentType
Association Source -> Destination	HwComponentInstanceRef	HardwareComponentPrototype
Generalization Source -> Destination	HwComponentInstanceRef	SAFEElement

9.3.2.16 Class HwFaultCharacterization

Element Base Classes: **HWFailureAnalysis**

Element Notes:

HwFaultCharacterization is the container to store the context of HWFault applicable to the instanceRef of the HardwareComponentPrototype

Connections

Connector	Source	Target
Aggregation Source -> Destination	HwComponentInstanceRef	HwFaultCharacterization

Connector	Source	Target
Generalization Source -> Destination	HwFaultCharacterization	HWFailureAnalysis
Aggregation Source -> Destination	HWFault	HwFaultCharacterization

9.3.3 Package HWQuantitativeMeasure

Package Notes:

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.

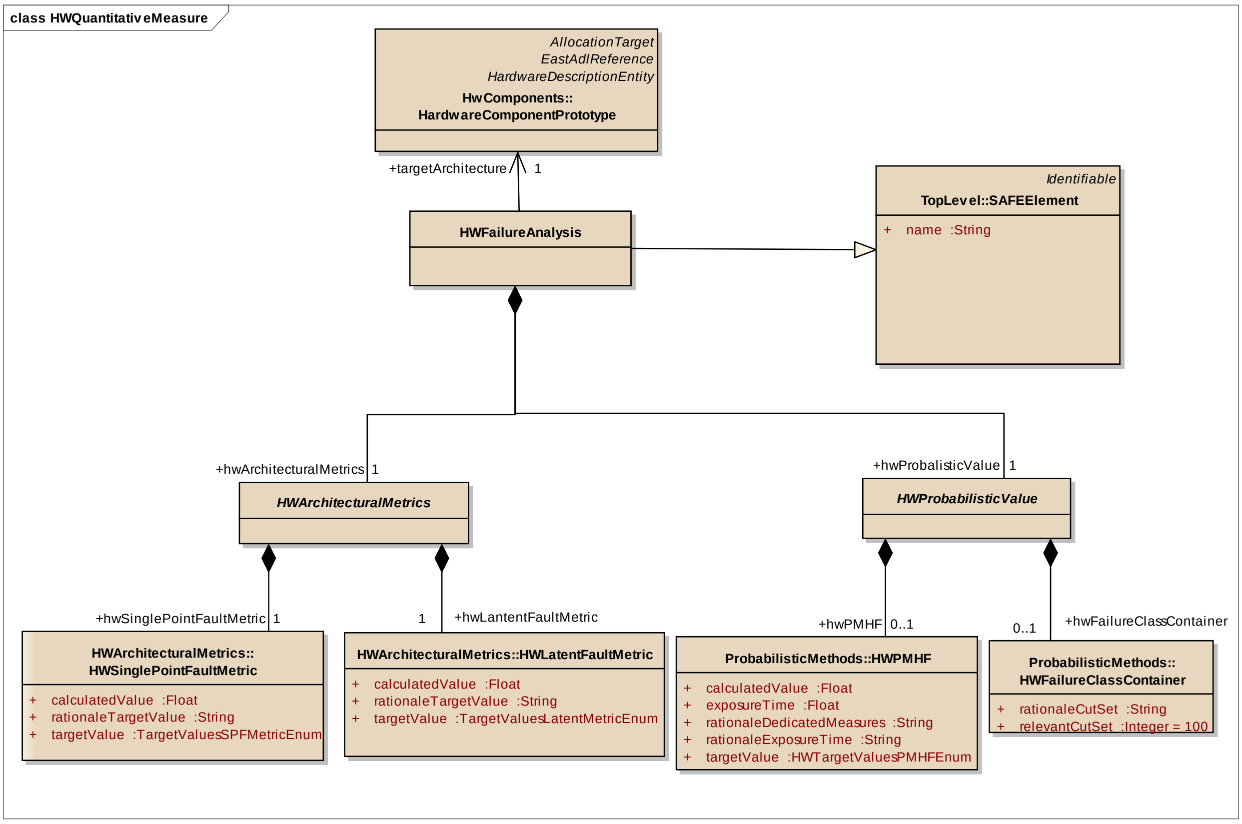

Figure 6: **HWQuantitativeMeasure** - *(Class diagram)*

Diagram Notes:

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

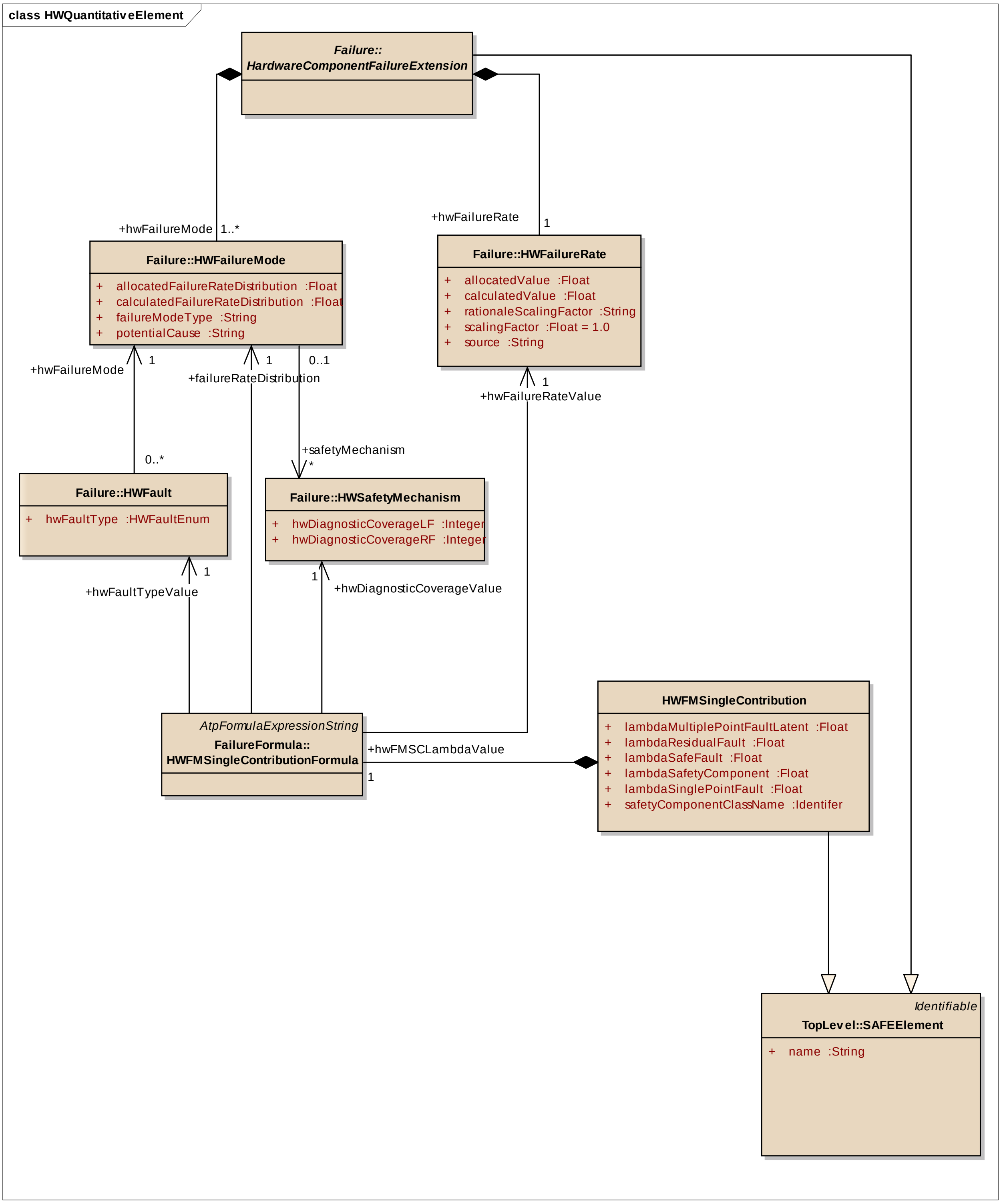

Figure 7: **HWQuantitativeElement** - *(Class diagram)*

Diagram Notes:

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

© 2012 The SAFE Consortium

9.3.3.1 Class HWArchitecturalMetrics

Element Base Classes: **SAFEElement**

Element Notes:

This class represents an abstract definition of all quantified failure analysis required by the ISO Part 5 Clause 8. This class allows mapping all meta class for the HWArchitecturalMetrics also described in the ISO Part 5-Annex C (Single-Point-Fault Metric, Latent-Fault Metric).

Each HWQuantifiedFailureAnalysis belongs to exactly one malfunction (link to violation of a SafetyGoal). The ASIL-TargetValue (e.g. ASIL-D) is derived from the SafetyGoal.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HWArchitecturalMetrics	HWFailureAnalysis
Generalization Source -> Destination	HWArchitecturalMetrics	SAFEElement
Aggregation Source -> Destination	HWLatentFaultMetric	HWArchitecturalMetrics
Aggregation Source -> Destination	HWSinglePointFaultMetric	HWArchitecturalMetrics

9.3.3.2 Class HWFMSingleContribution

Element Base Classes: **SAFEElement**

Element Notes:

This class describes the single contribution in term of failure rate (lambda) to the elementary metrics of the HW Fault for each failure mode of a HWComponent. This entity is used to store preliminary element used in the context of architectural metrics and probabilistic measurement.

The calculation of the attribute is derived from the Formula Expression HWFMSingleContributionFormula

Connections

Connector	Source	Target
Generalization Source -> Destination	HWFMSingleContribution	SAFEElement
Association Source -> Destination	HWPMHFFormula	HWFMSingleContribution
Association Source -> Destination	HWFailureClassContributionFormula	HWFMSingleContribution

Connector	Source	Target
Aggregation Source -> Destination	HWFMSingleContributionFormula	HWFMSingleContribution
Association Source -> Destination	HWSinglePointFaultMetricFormula	HWFMSingleContribution
Association Source -> Destination	HWLatentFaultMetricFormula	HWFMSingleContribution

Attributes

Attribute	Notes	Default
lambdaMultiplePointFaultLatent Float	This attribute stores the specific failure rate for single failure mode contribution as multiple-point latent, lambda(MPF,L).	
lambdaResidualFault Float	This attribute stores the specific failure rate for single failure mode contribution as residual fault, lambda(RF).	
lambdaSafeFault Float	This attribute stores the specific failure rate for single failure mode contribution as safe fault, lambda(SF).	
lambdaSafetyComponent Float	This attribute stores the sum of specific failure rates for the hardware component for verification.	
lambdaSinglePointFault Float	This attribute stores the specific failure rate for single failure mode contribution as single-point fault, lambda(SPF).	
safetyComponentClassName Identifer	This attribute stores the name of the hardware component class.	

9.3.3.3 Class HWFailureAnalysis

Element Base Classes: **SAFEElement**

Element Notes:

This class represents the container for all quantified failure analysis required by the ISO 26262 Part 5 for a dedicated SafetyGoal. This class allows clustering all meta class for the HWArchitecturalMetrics described in the ISO Part 5 Clause 8 (Single-Point-Fault Metric, Latent-Fault Metric) and probabilistic value for violation of safety goal (PMH) or Failure Class Method described in the ISO Part 5 Clause 9.

Each HWFailureAnalysis belongs to exactly one SafetyGoal. The ASIL-TargetValue (e.g. ASIL-D) is derived from the SafetyGoal.

Connections

Connector	Source	Target
Association	HWFailureAnalysis	HardwareComponentPrototype

© 2012 The SAFE Consortium

Connector	Source	Target
Source -> Destination		
Aggregation Source -> Destination	HWProbabilisticValue	HWFailureAnalysis
Aggregation Source -> Destination	HWArchitecturalMetrics	HWFailureAnalysis
Generalization Source -> Destination	HwFaultCharacterization	HWFailureAnalysis
Generalization Source -> Destination	HWFailureAnalysis	SAFEElement

9.3.3.4 Class HWProbabilisticValue

Element Base Classes: **SAFEElement**

Element Notes:

This class represents an abstract definition of all failure analysis required by the ISO Part 5 Clause 9. This class allows mapping all meta class for the evaluation of safety goal violation (PMHF and Failure Rate Class).

Each HWQuantifiedFailureAnalysis belongs to exactly one malfunction (link to violation of a SafetyGoal). The ASIL-TargetValue (e.g. ASIL-D) is derived from the SafetyGoal.

Connections

Connector	Source	Target
Generalization Source -> Destination	HWProbabilisticValue	SAFEElement
Aggregation Source -> Destination	HWProbabilisticValue	HWFailureAnalysis
Aggregation Source -> Destination	HWFailureClassContainer	HWProbabilisticValue
Aggregation Source -> Destination	HWPMHF	HWProbabilisticValue

9.3.4 Package HWArchitecturalMetrics

Package Notes:

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.

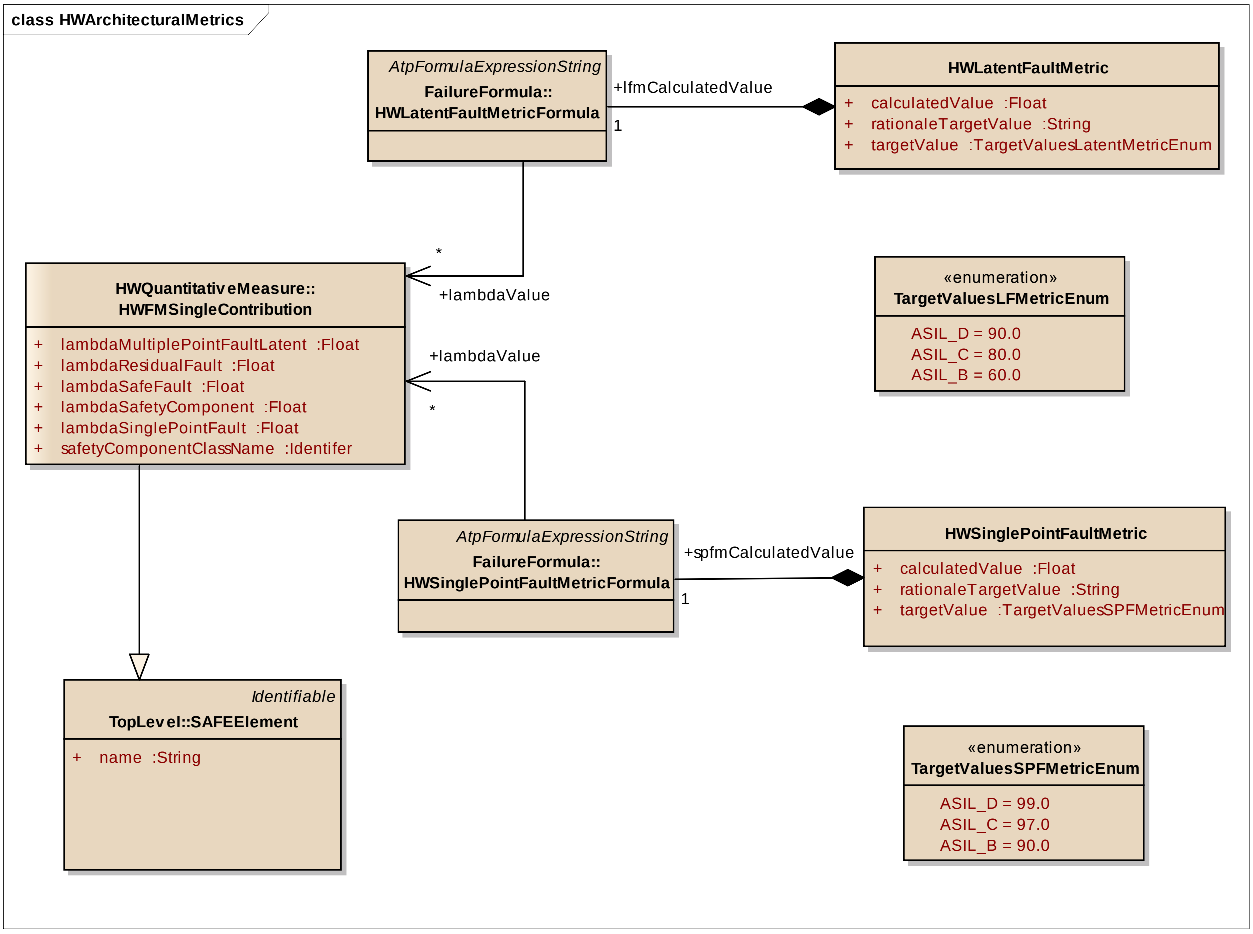

Figure 8: **HWArchitecturalMetrics** - *(Class diagram)*

Diagram Notes:

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

9.3.4.1 Class HWLatentFaultMetric

Element Base Classes: **SAFEElement**

Element Notes:

This class is 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).

The calculation is included in the class HWLatentFaultMetricFormula.

Connections

Connector	Source	Target
Generalization Source -> Destination	HWLatentFaultMetric	SAFEElement
Aggregation	HWLatentFaultMetric	HWArchitecturalMetrics

Connector	Source	Target
Source -> Destination		
Aggregation Source -> Destination	HWLatentFaultMetricFormula	HWLatentFaultMetric

Attributes

Attribute	Notes	Default
calculatedValue Float	The calculatedValue is the result of the calculation of the latent fault metric (in %).	
rationaleTargetValue String	The attribute rationaleTargetValue provides a textual rationale, if other target values are applied (see ISO Part 5 8.4.6).	
targetValue TargetValuesLatentMetricEnum	The attribute targetValue describes the target value for the HWLatentFaultMetric, derived from the ASIL of the SafetyGoal (see ISO Part 5 8.4.6).	

9.3.4.2 Class HWSinglePointFaultMetric

Element Base Classes: **SAFEElement**

Element Notes:

This class is 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).

The calculation is included in the class HWSinglePointFaultMetricFormula.

Connections

Connector	Source	Target
Generalization Source -> Destination	HWSinglePointFaultMetric	SAFEElement
Aggregation Source -> Destination	HWSinglePointFaultMetric	HWArchitecturalMetrics
Aggregation Source -> Destination	HWSinglePointFaultMetricFormula	HWSinglePointFaultMetric

Attributes

Attribute	Notes	Default
calculatedValue Float	The calculatedValue is the result of the calculation of the single-point fault metric (in %).	

Attribute	Notes	Default
rationaleTargetValue String	The attribute rationaleTargetValue provides a textual rationale, if other target values are applied (see ISO Part 5 8.4.5).	
targetValue TargetValuesSPFMetricEnum	The attribute targetValue describes the target value for the HWSinglePointFaultMetric, derived from the ASIL of the SafetyGoal (see ISO Part 5 8.4.5).	

9.3.4.3 Enumeration TargetValuesLFMetricEnum

Element Base Classes: **SAFEElement**

Element Notes:

Part 5-8.4.6 Table 5 (Possible source for the derivation of the target "latent-fault-metric" value)

Connections

Connector	Source	Target
Generalization Source -> Destination	TargetValuesLFMetricEnum	SAFEElement

Attributes

Attribute	Notes	Default
ASIL_D Float	This literal contains the target value for latent-fault metric for ASIL-D.	90.0
ASIL_C Float	This literal contains the target value for latent-fault metric for ASIL-C.	80.0
ASIL_B Float	This literal contains the target value for latent-fault metric for ASIL-B.	60.0

9.3.4.4 Enumeration TargetValuesSPFMetricEnum

Element Base Classes: **SAFEElement**

Element Notes:

Part 5-8.4.5 Table 4 (Possible source for the derivation of the target "single-point-fault-metric" value)

Connections

Connector	Source	Target
Generalization Source -> Destination	TargetValuesSPFMetricEnum	SAFEElement

Attributes

Attribute	Notes	Default
ASIL_D Float	This literal contains the target value for single-point fault metric for ASIL-D.	99.0
ASIL_C Float	This literal contains the target value for single-point fault metric for ASIL-C.	97.0
ASIL_B Float	This literal contains the target value for single-point fault metric for ASIL-B.	90.0

9.3.5 Package ProbabilisticMethods

Package Notes:

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).

© 2012 The SAFE Consortium

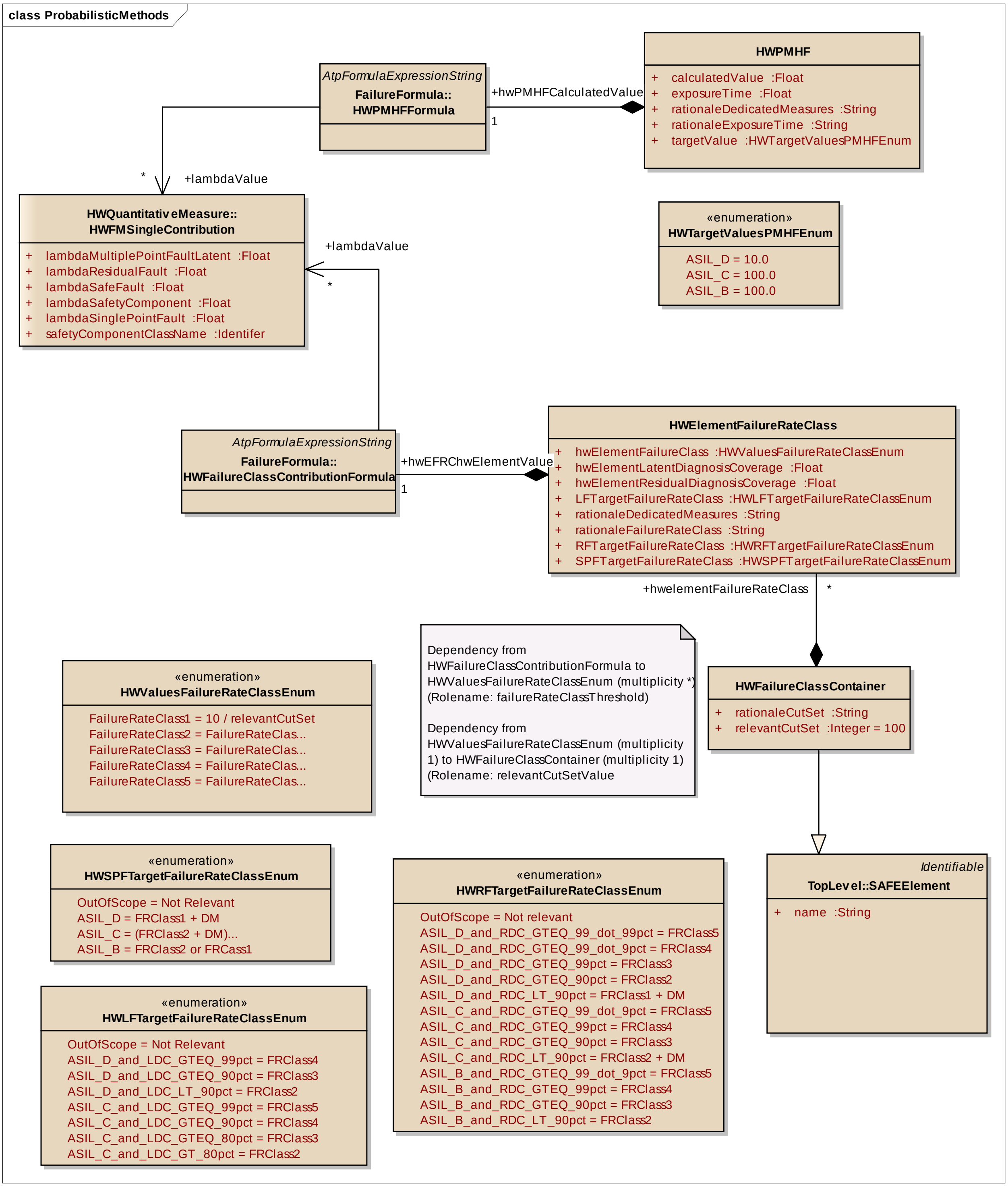

Figure 9: **ProbabilisticMethods** - *(Class diagram)*

Diagram Notes:

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

9.3.5.1 Class HWElementFailureRateClass

Element Base Classes: **SAFEElement**

Element Notes:

This class describes for a HWComponent, the FailureRateClass 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 diagnosis 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 HWElementDiagnosisCoverage is derived from the Formula Expression FMSingleContributionFormula.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HWElementFailureRateClass	HWFailureClassContainer
Generalization Source -> Destination	HWElementFailureRateClass	SAFEElement
Aggregation Source -> Destination	HWFailureClassContributionFormula	HWElementFailureRateClass

Attributes

Attribute	Notes	Default
hwElementFailureClass HWValuesFailureRateClassEnum	Failure Rate Class taken from HWValuesRateClassEnum based on the failure rate of the hardware component.	
hwElementLatentDiagnosisCoverage Float	The diagnostic coverage 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.	
hwElementResidualDiagnosisCoverage Float	The diagnostic coverage with respect to residual faults on hardware element level, calculated with the specific failure rate of all single-point and residual faults and the overall failure rate of the hardware part element.	
LFTargetFailureRateClass HWLFTargetFailureRateClassEnum	Target Failure Rate Class for multiple-point latent faults, taken from HWLFTargetFailureRateClassEnum.	
rationaleDedicatedMeasures String	Provides 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	

Attribute	Notes	Default
	(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.	
rationaleFailureRateClass String	Rationale for matching criteria on Failure Rate Class.	
RFTargetFailureRateClass HWRFTargetFailureRateClassEnum	Target Failure Rate Class for residual faults, taken from HWRFTargetFailureRateClassEnum.	
SPFTargetFailureRateClass HWSPFTargetFailureRateClassEnum	Target Failure Rate Class for single-point faults, taken from HWSPFTargetFailureRateClassEnum.	

9.3.5.2 Class HWFailureClassContainer

Element Base Classes: **SAFEElement**

Element Notes:

This class is container to store all HW element failure class results and associated assumptions taken (number of cut-set as typical).

Connections

Connector	Source	Target
Aggregation Source -> Destination	HWElementFailureRateClass	HWFailureClassContainer
Aggregation Source -> Destination	HWFailureClassContainer	HWProbabilisticValue
Generalization Source -> Destination	HWFailureClassContainer	SAFEElement

Attributes

Attribute	Notes	Default
rationaleCutSet String	This attribute provides a textual rationale for the number of relevant cut-sets.	
relevantCutSet	This attributes stores the number of relevant cut sets.	100

Attribute	Notes	Default
Integer		

9.3.5.3 Enumeration HWLFTargetFailureRateClassEnum

Element Base Classes: **SAFEElement**

Element Notes:

ISO 26262 Part 5 9.4.3.11 -Table 9 (Targets of failure rate class and coverage of hardware part regarding dual-point faults)

DM: Dedicated measures

LDC: Diagnostic coverage with respect to latent faults

Additionally, OUT-OF-SCOPE was added.

Connections

Connector	Source	Target
Generalization Source -> Destination	HWLFTargetFailureRateClassEnum	SAFEElement

Attributes

Attribute	Notes	Default
OutOfScope String	This literal describes values which are out of scope for the analysis.	Not Relevant
ASIL_D_and_LDC_GTEQ_99pct String	This literal describes a single cell with value target failure rate class 4 in the table for ASIL-D and latent diagnostic coverage >= 99%.	FRClass4
ASIL_D_and_LDC_GTEQ_90pct String	This literal describes a single cell with value target failure rate class 3 in the table for ASIL-D and latent diagnostic coverage >= 90%.	FRClass3
ASIL_D_and_LDC_LT_90pct String	This literal describes a single cell with value target failure rate class 2 in the table for ASIL-D and latent diagnostic coverage < 90%.	FRClass2
ASIL_C_and_LDC_GTEQ_99pct String	This literal describes a single cell with value target failure rate class 5 in the table for ASIL-C and latent diagnostic coverage >= 99%.	FRClass5
ASIL_C_and_LDC_GTEQ_90pct String	This literal describes a single cell with value target failure rate class 4 in the table for ASIL-C and latent diagnostic coverage >= 90%.	FRClass4

© 2012 The SAFE Consortium

Attribute	Notes	Default
ASIL_C_and_LDC_GTEQ_80pct String	This literal describes a single cell with value target failure rate class 3 in the table for ASIL-C and latent diagnostic coverage >= 80%. Rationale provided by ISO 26262 Part 5 9.4.3.9.	FRClass3
ASIL_C_and_LDC_GT_80pct String	This literal describes a single cell with value target failure rate class 2 in the table for ASIL-C and latent diagnostic coverage < 80%. Rationale provided by ISO 26262 Part 5 9.4.3.9.	FRClass2

9.3.5.4 Class HWPMHF

Element Base Classes: **SAFEElement**

Element Notes:

This class describes the Probabilistic Metric for random Hardware Failures (PMHF) as in ISO Part 5 Clause 9.4.2.

A simplified calculation is included in the class HWPMHFFormula.

Connections

Connector	Source	Target
Aggregation Source -> Destination	HWPMHFFormula	HWPMHF
Generalization Source -> Destination	HWPMHF	SAFEElement
Aggregation Source -> Destination	HWPMHF	HWProbabilisticValue

Attributes

Attribute	Notes	Default
calculatedValue Float	The calculatedValue is the result of the calculation of the PMHF (in FIT).	
exposureTime Float	The exposure time is the duration of exposure, shall be expressed in h.	
rationaleDedicatedMeasures String	The attribute rationaleDedicatedMeasures shall allow to define a rationale for applied dedicated measures in the design.	
rationaleExposureTime	The attribute rationaleExposureTime is for Documentation	

Attribute	Notes	Default
String	of rationale for Exposure Time.	
targetValue HWTargetValuesPMHFEnum	The attribute targetValue describes the target value for the PMHF, derived from the ASIL of the SafetyGoal (see ISO Part 5 9.4.2.1).	

9.3.5.5 Enumeration HWRFTargetFailureRateClassEnum

Element Base Classes: **SAFEElement**

Element Notes:

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).

DM: Dedicated measures

RDC: Diagnostic coverage with respect to residual faults

This class 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 hwElementResidualDiagnosisCoverage parameter of the HWElementFailureRateClass

Additionally, "OUT-OF-SCOPE" and "ASIL-D and RDC >=99.99%" according to ISO 26262 Part 5 9.4.3.7.

Connections

Connector	Source	Target
Generalization Source -> Destination	HWRFTargetFailureRateClassEnum	SAFEElement

Attributes

Attribute	Notes	Default
OutOfScope String	This literal describes values which are out of scope for the analysis.	Not relevant
ASIL_D_and_RDC_GTEQ_99_dot_99pct String	This literal describes a single cell with value target failure rate class 5 in the table for ASIL-D and residual fault diagnostic coverage >= 99.99%. Failure Rate Class determined according to ISO 26262 Part 5 9.4.3.7.	FRClass5
ASIL_D_and_RDC_GTEQ_99_dot_9pct String	This literal describes a single cell with value target failure rate class 4 in the table for ASIL-D and residual fault diagnostic coverage >= 99.9%.	FRClass4
ASIL_D_and_RDC_GTEQ_99pct	This literal describes a single cell with value target failure rate class 3 in the table for ASIL-D and residual fault diag-	FRClass3

Attribute	Notes	Default
String	nostic coverage >= 99%.	
ASIL_D_and_RDC_GTEQ_90pct String	This literal describes a single cell with value target failure rate class 2 in the table for ASIL-D and residual fault diagnostic coverage >= 90%.	FRClass2
ASIL_D_and_RDC_LT_90pct String	This literal describes a single cell with value target failure rate class 1 + dedicated measures in the table for ASIL-D and residual fault diagnostic coverage < 90%.	FRClass1 + DM
ASIL_C_and_RDC_GTEQ_99_dot_9pct String	This literal describes a single cell with value target failure rate class 5 in the table for ASIL-C and residual fault diagnostic coverage >= 99.9%.	FRClass5
ASIL_C_and_RDC_GTEQ_99pct String	This literal describes a single cell with value target failure rate class 4 in the table for ASIL-C and residual fault diagnostic coverage >= 99%.	FRClass4
ASIL_C_and_RDC_GTEQ_90pct String	This literal describes a single cell with value target failure rate class 3 in the table for ASIL-C and residual fault diagnostic coverage >= 90%.	FRClass3
ASIL_C_and_RDC_LT_90pct String	This literal describes a single cell with value target failure rate class 2 + dedicated measures in the table for ASIL-C and residual fault diagnostic coverage < 90%.	FRClass2 + DM
ASIL_B_and_RDC_GTEQ_99_dot_9pct String	This literal describes a single cell with value target failure rate class 5 in the table for ASIL-B and residual fault diagnostic coverage >= 99.9%.	FRClass5
ASIL_B_and_RDC_GTEQ_99pct String	This literal describes a single cell with value target failure rate class 4 in the table for ASIL-B and residual fault diagnostic coverage >= 99%.	FRClass4
ASIL_B_and_RDC_GTEQ_90pct String	This literal describes a single cell with value target failure rate class 3 in the table for ASIL-B and residual fault diagnostic coverage >= 90%.	FRClass3
ASIL_B_and_RDC_LT_90pct String	This literal describes a single cell with value target failure rate class 2 in the table for ASIL-B and residual fault diagnostic coverage < 90%.	FRClass2

9.3.5.6 Enumeration HWSPFTargetFailureRateClassEnum

Element Base Classes: **SAFEElement**

Element Notes:

ISO 26262 Part 5 9.4.3.5 -Table 7 (Targets of failure rate classes of hardware parts regarding single-point faults)

DM: Dedicated measures

Additionally, OUT-OF-SCOPE was added.

Connections

Connector	Source	Target
Generalization Source -> Destination	HWSPFTargetFailureRateClassEnum	SAFEElement

Attributes

Attribute	Notes	Default
OutOfScope String	This literal describes values which are out of scope for the analysis.	Not Relevant
ASIL_D String	This literal describes a single cell with value target failure rate class 1 + dedicated measures in the table for ASIL-D.	FRClass1 + DM
ASIL_C String	This literal describes a single cell with value target failure rate class 2 + dedicated measures or failure rate class 1 in the table for ASIL-C.	(FRClass2 + DM) or FRClass1
ASIL_B String	This literal describes a single cell with value target failure rate class 2 or failure rate class 1 in the table for ASIL-B.	FRClass2 or FRCass1

9.3.5.7 Enumeration HWTargetValuesPMHFEnum

Element Base Classes: **SAFEElement**

Element Notes:

Target values for PMHF according to ISO 26262 Part 5 9.4.2.1. The values here are described in FIT (ppm/h) or FIT.

ASIL-D = 1.10-8 h-1 = 10 ppm/h

ASIL-C = ASIL-B = 1.10-7 => 100 ppm/h

Connections

Connector	Source	Target
Generalization Source -> Destination	HWTargetValuesPMHFEnum	SAFEElement

Attributes

Attribute	Notes	Default
ASIL_D Float	This attributes stores target value for PMHF for ASIL-D.	10.0

Attribute	Notes	Default
ASIL_C Float	This attributes stores target value for PMHF for ASIL-C.	100.0
ASIL_B Float	This attributes stores target value for PMHF for ASIL-B.	100.0

9.3.5.8 Enumeration HWValuesFailureRateClassEnum

Element Base Classes: **SAFEElement**

Element Notes:

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 cutset.

Connections

Connector	Source	Target
Generalization Source -> Destination	HWValuesFailureRateClassEnum	SAFEElement
Generalization Source -> Destination	HWValuesFailureRateClassEnum	SAFEElement

Attributes

Attribute	Notes	Default
FailureRateClass1 Float	This attribute contains the maximum value for failure rate class 1 ranking (in FIT).	10 / relevantCutSet
FailureRateClass2 Float	This attribute contains the maximum value for failure rate class 2 ranking (in FIT).	FailureRateClass1 * 10
FailureRateClass3 Float	This attribute contains the maximum value for failure rate class 3 ranking (in FIT).	FailureRateClass2 * 10
FailureRateClass4 Float	This attribute contains the maximum value for failure rate class 4 ranking (in FIT).	FailureRateClass3 * 10
FailureRateClass5 Float	This attribute contains the maximum value for failure rate class 5 ranking (in FIT).	FailureRateClass4 * 10

9.3.6 Package Traceability

Package Notes:

This sub-package contains the traceability of safety requirements.

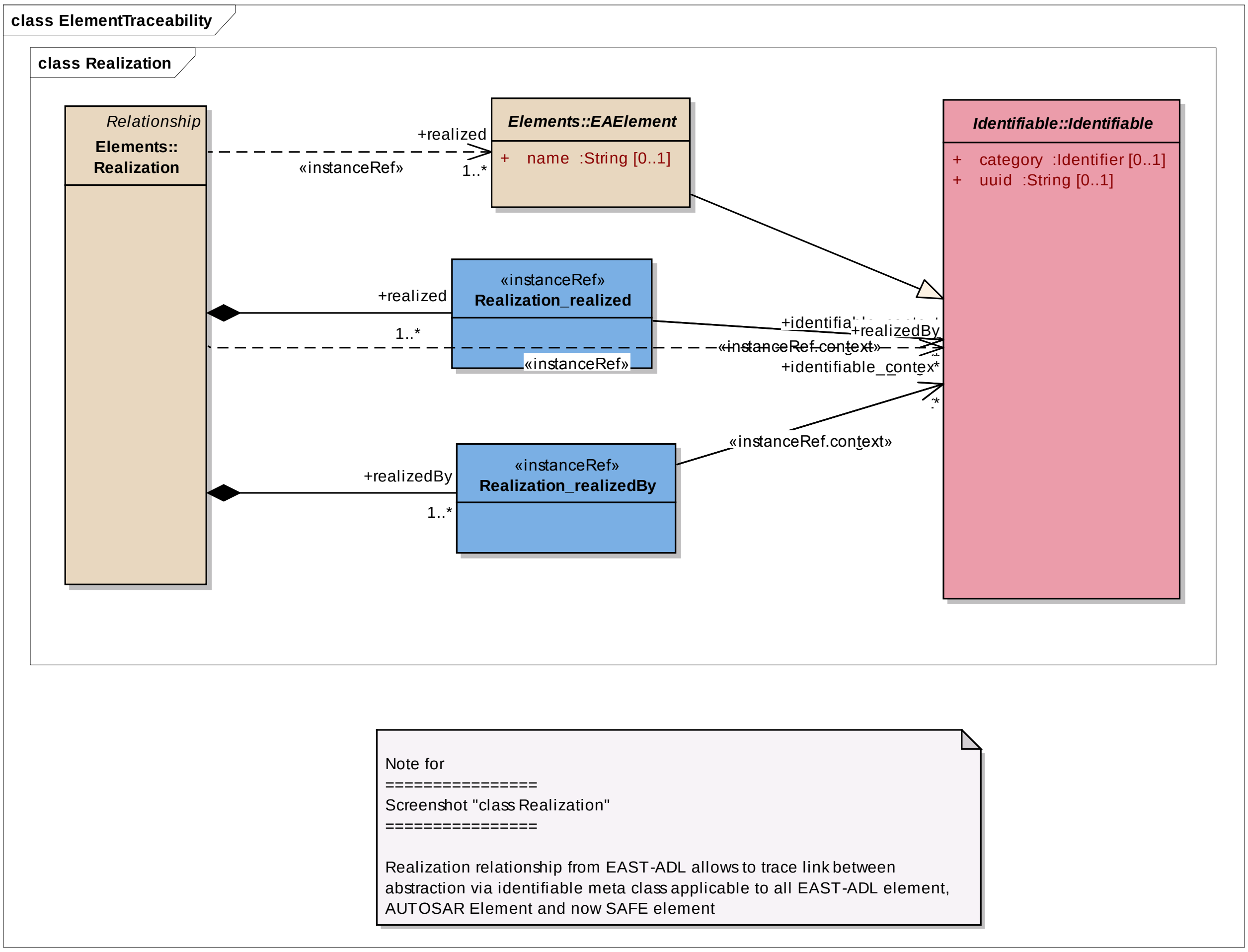

Figure 10: **ElementTraceability** - *(Class diagram)*

Diagram Notes:

This diagram contains the traceability of hardware components to detailed level of abstraction. The class diagram is a simple copy of EAST-ADL tracing artifacts as they will be fully reused.

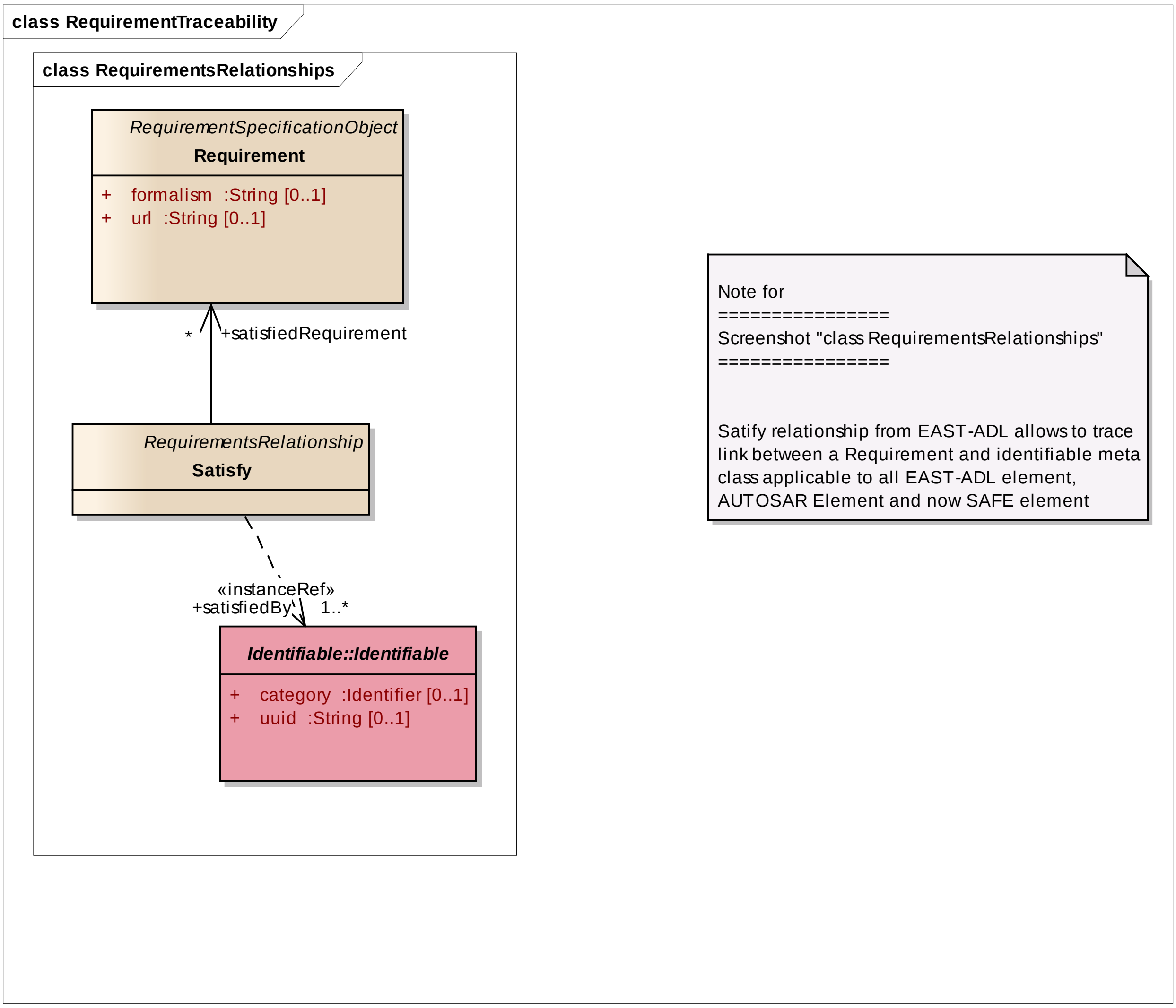

Figure 11: **RequirementTraceability** - *(Class diagram)*

Diagram Notes:

This diagram contains the traceability of requirements.

The class diagram is a simple copy of EAST-ADL requirement tracing artifacts as they will be fully reused.

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.

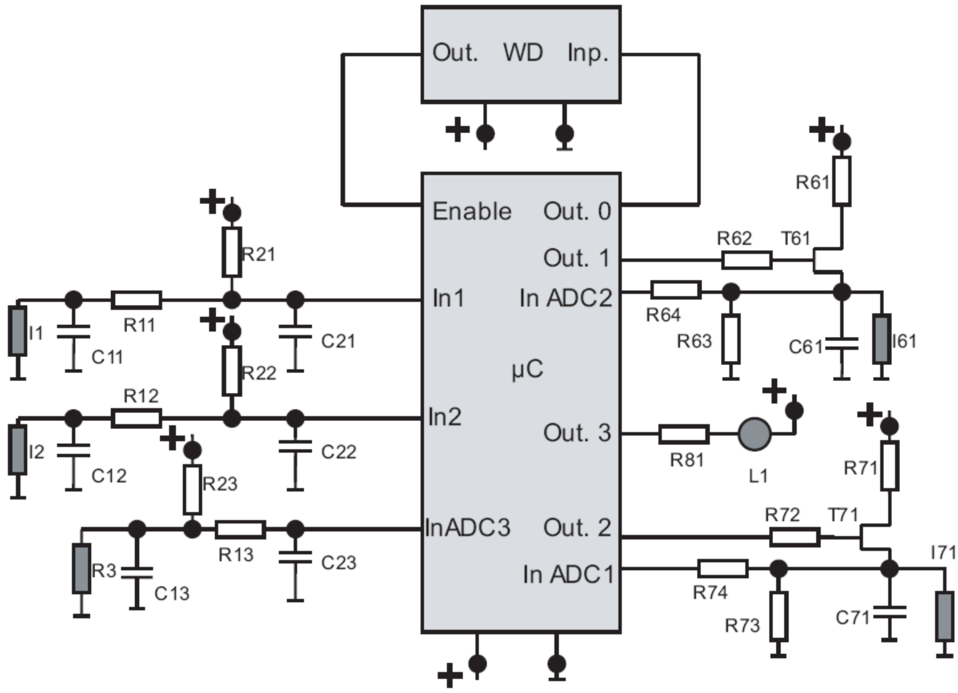

Figure 13: Electronic Schematic diagram ISO26262-Part5 Figure E.1

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 architecture 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. also, notice that the Hardware Software Interface (HIS) required by a standard TSC has not been added, due to not overwrite the figure.

In this following section, the hardware modeling methods, with dependency to failure propagation from WT 3.3.1 contribution, 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.

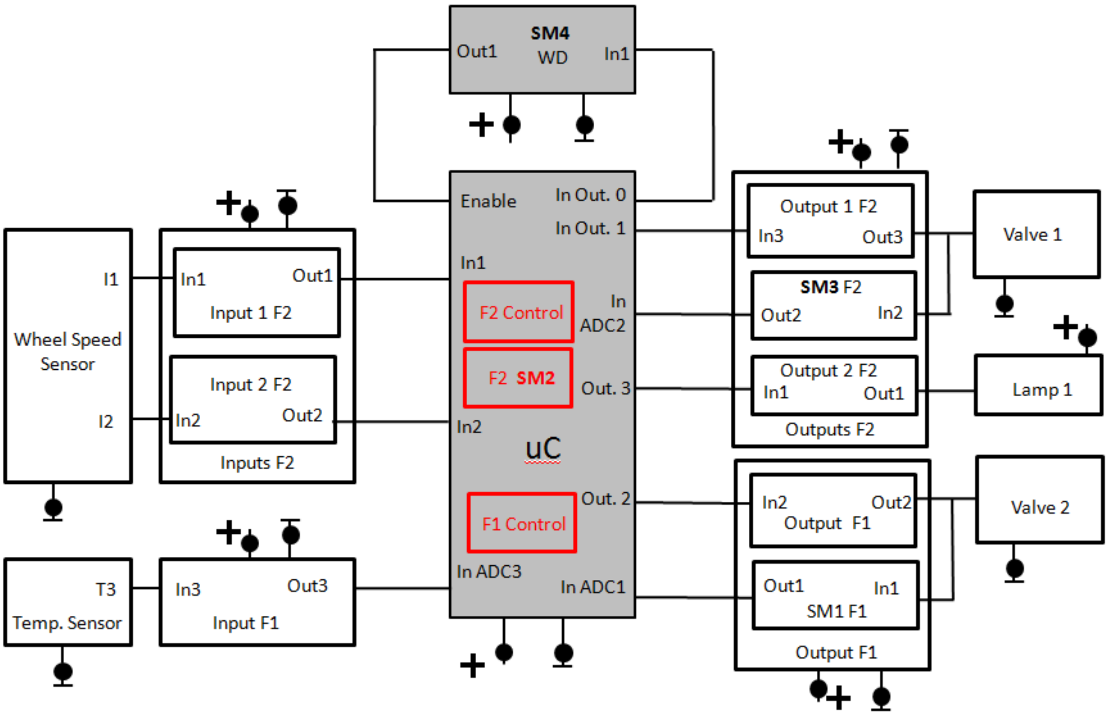

Figure 14: Technical Safety Concept for ISO26262-Part5 Figure E.1

10.1.1 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.

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

- Capture failure behavior interrelation 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; tag failure (Single Point, Residual, Multiple Point Latent
- Identify primary Hardware Safety Requirements based on the top-level malfunction of the HW Architecture Primary Hardware Safety Requirements are defined by the failure modes of the Hardware Components.

10.1.3 Step 3: Define target values for HW Components and calculates 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 and define additional measures (or revise assumption target values)
- Validate the preliminary hardware architectural metric results with the target value of the ASIL

10.1.4 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 Component has the malfunction (primary Hardware Safety Requirement) are used for Hardware Part Analysis

10.1.5 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)

10.1.6 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 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)

10.1.7 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

© 2012 The SAFE Consortium

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.

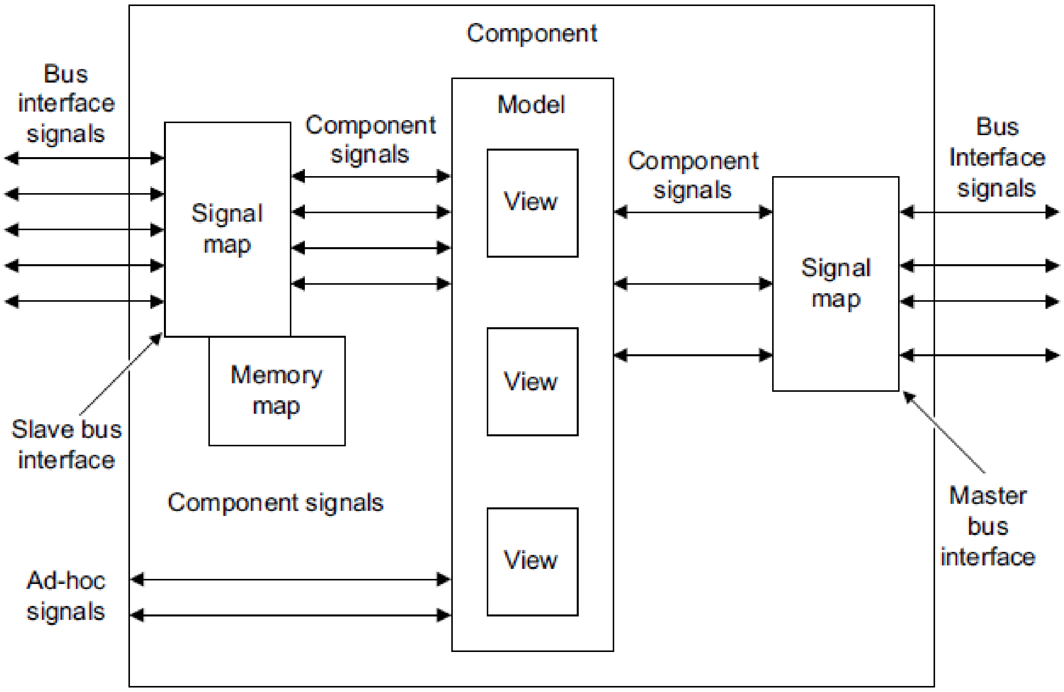

Figure 15: Structure of a component IP-XACT

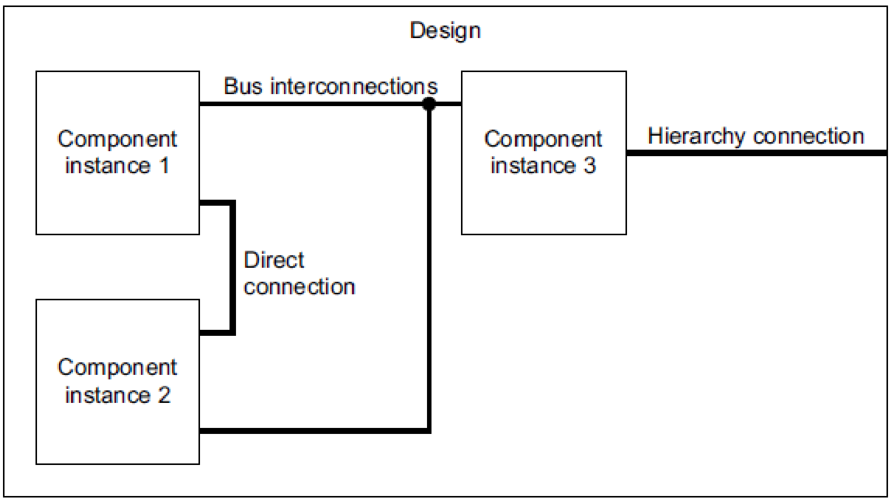

Figure 16: Design representation in IP-XACT

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.

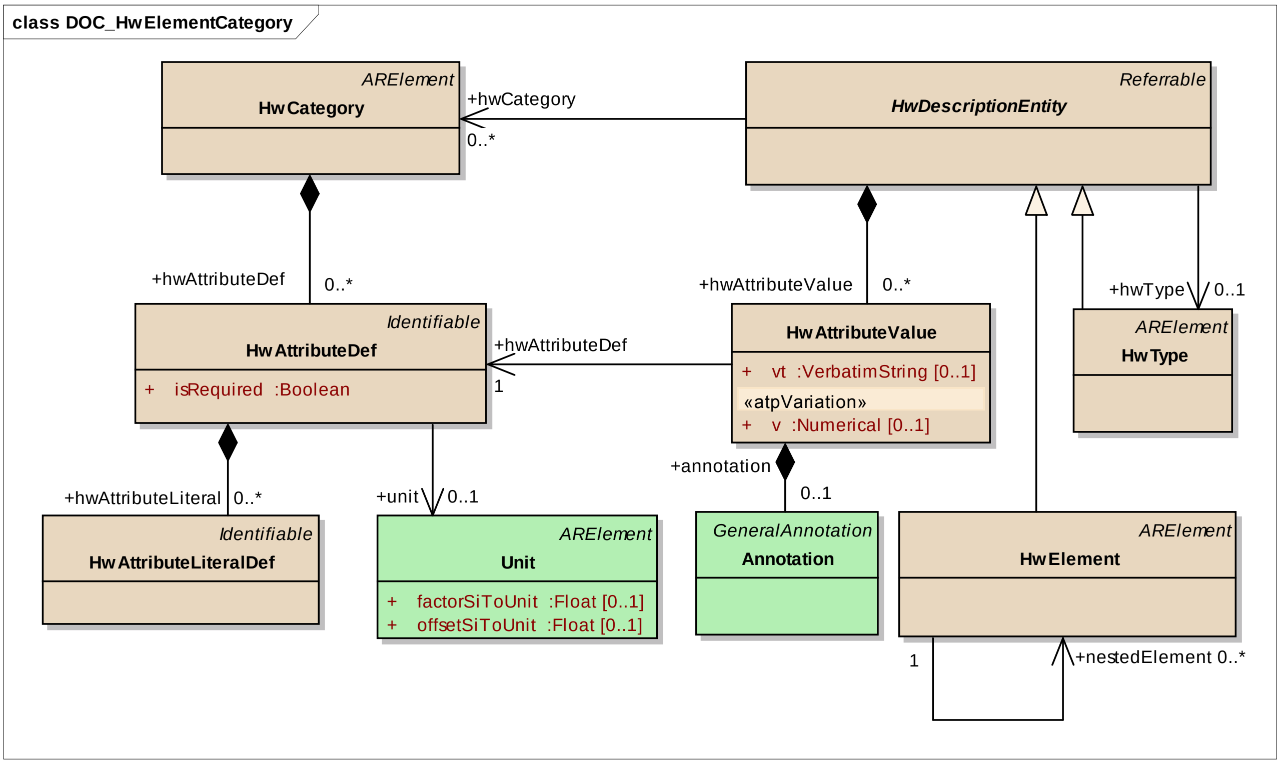

Figure 17: AUTOSAR HWcomponent and HWCategory

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-XCAT 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-XCAT 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.

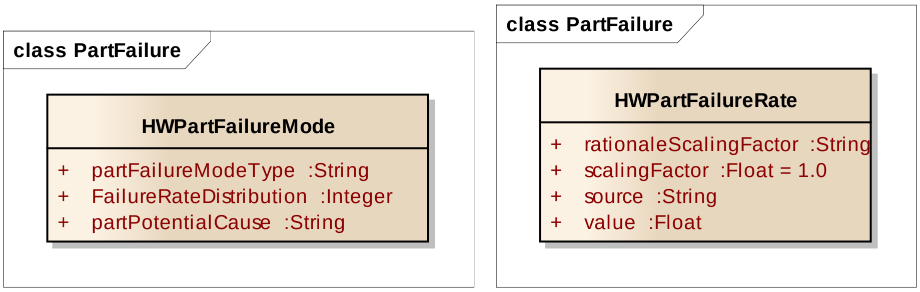

Figure 19: Hardware Part Failure information for IP-XACT

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 is intended to provide information about the proposal for a methodology for the hardware modeling and propose an extension to be able to compute quantitative value of hardware element as architectural metrics and evaluation of the failure rate for violation of the safety goal conform to requirement addressed by the 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 presented and for changes proposed with concrete proposal for future change request in this standard (expressed as meta model solution for EAST-ADL 2.1).

A concrete example for use of the methodology as been proposed, it will use as reference for the D3.3.1.b methodology description in the coming release of SAFE WT3.3.1.

As some discussions are still ongoing for complete SAFE WT synchronization, in particular for malfunction relation to the hazard analysis WT3.1.1, and safety mechanism modeling from relation to WT3.2.1 and WT3.1.2, a new release of the meta-model extensions will be provided shortly before summer time during the completion of the loop two of the SAFE project.

© 2012 The SAFE Consortium

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

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.

© 2012 The SAFE Consortium