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 D3.5.b:

Intermediary release of System, SW, HW reference meta-model definition

Due date of deliverable: 30/05/13

Actual submission date: 29/05/13

Start date of the project: 01/07/2011 Duration: 36 months

Project coordinator name: Stefan Voget

Organization name of lead contractor for this deliverable: ITEMIS

Contributors: Aquintos, BMW, Conti-F, Conti-G, Dassault Systèmes, fortiss, FZI, Itemis, LaBRi, OFFIS, PS, TTTech, VEEM.

Revision chart and history log

Version	Date	Reason
0.1	2013-05-21	Initialization of document
0.2	2013-05-23	Conti-F review, change in section 2 and 5
1.0	29.05.2013	Generation of data-model

1 Table of contents

2 Introduction

This document describes the language specification of SAFE.

The meta-model is based on the integration of the results of the conceptual work of all work tasks 3.x. It defines a domain specific language for safety modeling and safety analysis on M2 level. The meta-model provides extensions to EAST-ADL and AUTOSAR which are linked via references.

This document describes the meta-modeling rules, technical principles and the language specification based on an Enterprise Architect file.

1. The meta-model

The language specification is the core of the methods of the SAFE project. Its main objective is to enhance existing techniques to be able to reach the ISO26262 process in the context of model based development of E/E-architectures in vehicles or sub-systems of vehicles.

The target is to use model based techniques to represent the required entities capable to support safety case evaluation and documentation of vehicle architectures (and/or the subsystems). Results of safety assessments are recorded and documented including hazard identification and scenario goal description, in relation to safety requirements, their decomposition in the functional and technical architecture.

It delivers an open automotive meta-model for description of all design artefacts supporting the engineering process. It is based on the experience from the automotive domain and other domains and on existing techniques such as EAST-ADL for functional abstraction, AUTOSAR for software component, and a hardware description. The hardware part is improved to support electrical distribution systems and new hardware component descriptions either from the EAST-ADL hardware design extension and AUTOSAR ECU resource template with inspiration/alignment to the IP_XACT standard. This meta-model covers specific modelling techniques to support analysis methods, to compare and manage architecture variants during the architecture design phase and as final product variant.

For the safety evaluation, the following objectives are considered:

- Failure error modelling and propagation to be able to perform safety and cut-set analysis thanks to model based technique, aligned with extending analysis methods such as FTA, FMEA with improved quantified value based on system execution context or new methods
- Hardware and software COTS evaluation methods for safety test conformity and integration in safety systems
- Clarification of needs via explicit elicitation of safety requirements and especially the decision tracing the safety measures
- Specification of criteria and methods for architecture safety evaluation and comparison to support design decisions regarding functional safety
- Generation of safety case documentation with respect to safety activities and related model based work-product

3 Modeling Rules

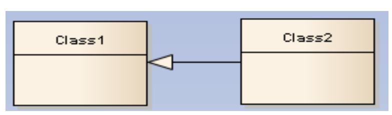
This chapter collects some basic principles used in the definition of the SAFE meta-model.

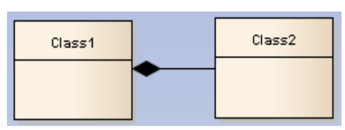
To define a homogeneous meta-model and enable automatic code generation, a set of rules have been defined.

2. *Use of Enterprise Architect*

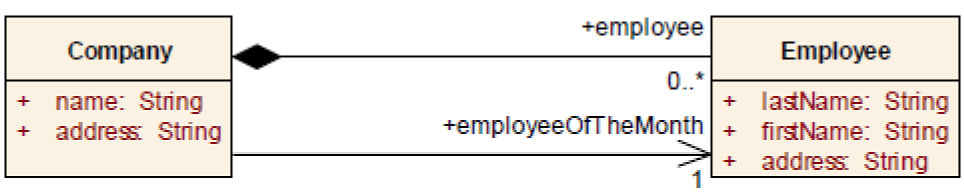
- **Rule 0:** Work on a replica
  - Rationale: Enables the merge function in EA, required to export a unique MM
- **Rule 1:** Provide documentation on
    - Diagrams
    - Packages
    - Classes
    - Attributes
    - Associations.
  - Rationale: Export from EA to deliverables.
    - It helps partners to understand your intentions
    - It can be used as javadoc in generated code
    - It can be included in generated documention

3. *Class relations*

- **Rule 2:** Only 3 class relation types are allowed:
  - Generalization 

  - Composition (i.e. contain) 

  - Direct Association



- **Rule 2.1:** For Compose and Association relations, multiplicity shall be explicitly defined
  - Rationale: otherwise generator take assumption.
- **Rule 2.2:** For Compose and Association have to be named at the navigable end, not the connector itself.

  - Rationale: otherwise relation have no mean for code generation
- **Rule 2.3:** Don't use Compose or Association relation with data-type or enumeration
  - Rationale: data-type and enumeration aren't class. (i.e. issue in code generation)
- **Rule 2.4:** always start creating the composition relation from the class to be aggregated
  - Rationale: Ecore generation process assumes that the source always aggregates the target.
- **Rule 2.5:** All associations starting at one class have to have different names at the navigable end.
  - Rationale: in Ecore the „features" have to be unique

4. Naming convention

- **Rule 3:** Only use java compliant names.
  - Rationale: Names used in the model will be used in generated code
  - Java compliant names (see http://java.sun.com/)
  - E.g.: no reserved keyword
  - Advice 1: Similar concept, different meta-model
    - Rationale: simplify understanding
    - Similar name. Ex: ARPackage <-> SAFEPackage
- **Rule 3.1:** Spaces in entries of enumerations are not allowed

5. Basic Data types

- **Rule 4:** Only use basic datatypes from the CommonStructure package
  - Rationale: what if a float is not a float ?
- **Rule 4.1 :** These types have the stereotype <<primitive>> or <<enumeration>>
- **Rule 4.2:** All attributes shall have a type
- **Rule 4.3:** Default value shall have the same type

6. Main structural principle of meta-model

SAFE uses SAFEElement as a root of the meta-model

- **Rule 5.1:** All classes inherit directly or indirectly from SAFEElement

SAFE uses SafetyExtension for structuring the meta-model along abstraction levels (the abstraction levels are taken from EAST-ADL and AUTOSAR).

- **Rule 5.2:** All classes are directly or indirectly contained in one of the subclasses of SafetyExtension

7. References

Elements from foreign meta-models (e.g. Chromosome, EAST-ADL, AUTOSAR, …) can be referenced by using proxy elements. These elements are located in the package CommonStructure / References

- **Rule 6.1:** reference elements are used by containment relationship only
    - Rationale: if one links them via an association, the referenced element can not be created in a tool.
- **Rule 6.2:** the reference element has the same name as the name in the original meta-model.
- **Rule 6.3:** The meta-model is chosen by the package in which the element is located in

4 Information Model Structure

The SAFE meta-model is structured in the following packages.

CommonStructure	Technical package defining basic structures
	• DataTypes
	  ○ Collects all used primitive data types
	• FormulaExpression
	  ○ Enables the definition of formula language
	• References
	  ○ Enables the link and usage of external meta-models (e.g. EAST-ADL, AUTOSAR, CHROMOSOME, …)
	• SafetyExtensions
	  ○ Basic structuring principle based on abstraction levels of SAFE meta-model
	• TopLevel
	  ○ 2nd structuring principle used in SAFE meta-model, based on hierarchies

Configuration	Link to variant management
ErrorModel	Basic component failure description as well as result of safety analysis
Hardware	Safety extensions on hardware level.
Hazards	Definition of hazard, risk, event, controllability
Requirements	Provides links to a requirements perspective and extends safety elements enabling the requirements traceability necessary to fulfill a safety process.
Software	Safety extensions on software level
System	Safety extensions on system level

5 Data Model documentation

This section provides the specification of the SAFE meta-model.

Package "SAFE Meta-Model"

Type of Package: **Package**

Parent Package: Model

Notes:

Diagram "**SAFE Meta-Model Overview**"

Notes:

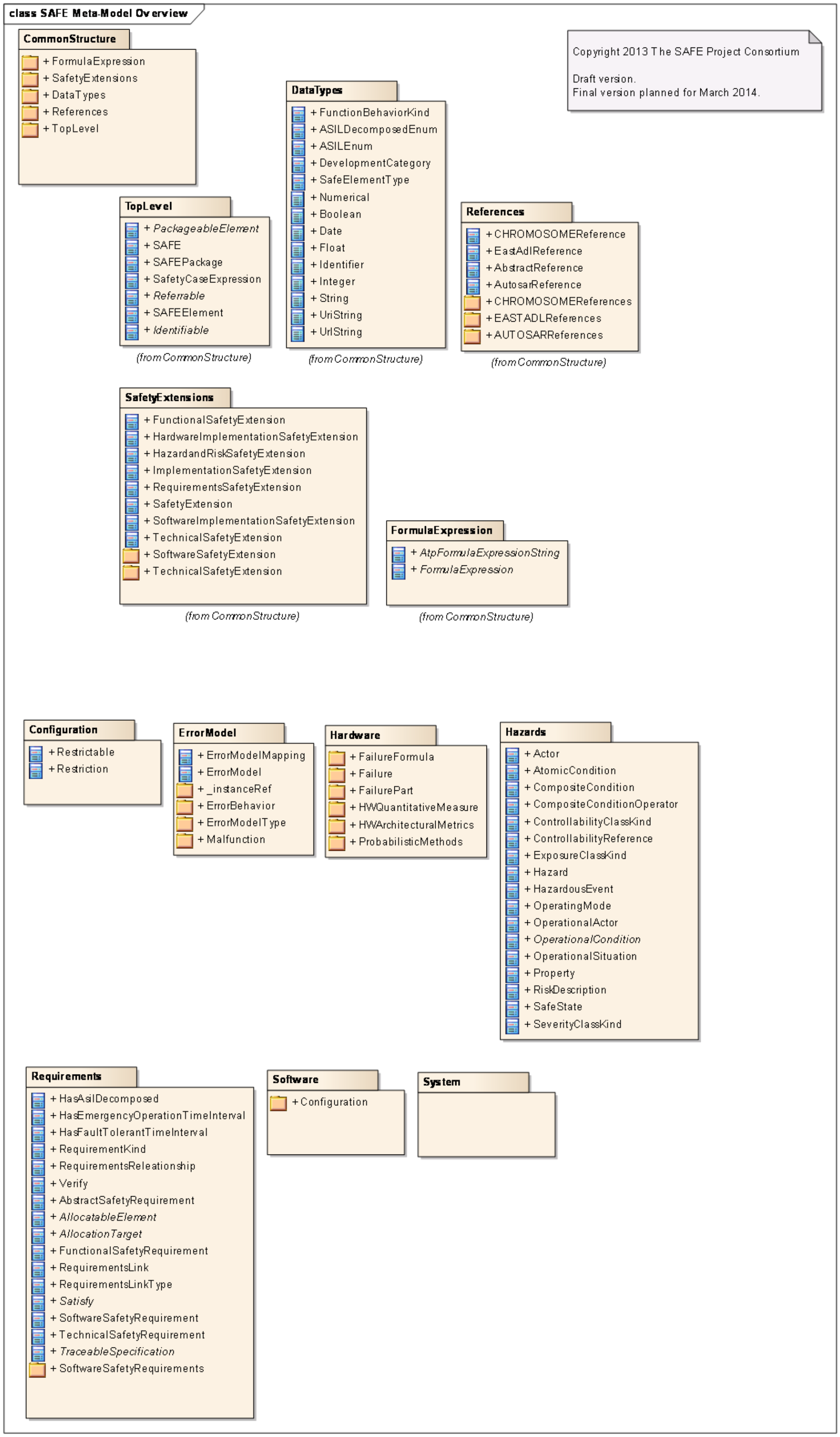

class SAFE Meta-Model Overview
Copyright 2013 The SAFE Project Consortium
Draft version.
Final version planned for March 2014.
CommonStructure
+ FormulaExpression
+ SafetyExtensions
+ DataTypes
+ References
+ TopLevel
DataTypes
+ FunctionBehaviorKind
+ ASILDecomposedEnum
+ ASILEnum
+ DevelopmentCategory
+ SafeElementType
+ Numerical
+ Boolean
+ Date
+ Float
+ Identifier
+ Integer
+ String
+ UriString
+ UrlString
(from CommonStructure)
TopLevel
+ PackageableElement
+ SAFE
+ SAFEPackage
+ SafetyCaseExpression
+ Referrable
+ SAFEElement
+ Identifiable
(from CommonStructure)
References
+ CHROMOSOMEReference
+ EastAdlReference
+ AbstractReference
+ AutosarReference
+ CHROMOSOMEReferences
+ EASTADLReferences
+ AUTOSARReferences
(from CommonStructure)
SafetyExtensions
+ FunctionalSafetyExtension
+ HardwareImplementationSafetyExtension
+ HazardandRiskSafetyExtension
+ ImplementationSafetyExtension
+ RequirementsSafetyExtension
+ SafetyExtension
+ SoftwareImplementationSafetyExtension
+ TechnicalSafetyExtension
+ SoftwareSafetyExtension
+ TechnicalSafetyExtension
(from CommonStructure)
FormulaExpression
+ AtpFormulaExpressionString
+ FormulaExpression
(from CommonStructure)
Configuration
+ Restrictable
+ Restriction
ErrorModel
+ ErrorModelMapping
+ ErrorModel
+ _instanceRef
+ ErrorBehavior
+ ErrorModelType
+ Malfunction
Hardware
+ FailureFormula
+ Failure
+ FailurePart
+ HWQuantitativeMeasure
+ HWArchitecturalMetrics
+ ProbabilisticMethods
Hazards
+ Actor
+ AtomicCondition
+ CompositeCondition
+ CompositeConditionOperator
+ ControllabilityClassKind
+ ControllabilityReference
+ ExposureClassKind
+ Hazard
+ HazardousEvent
+ OperatingMode
+ OperationalActor
+ OperationalCondition
+ OperationalSituation
+ Property
+ RiskDescription
+ SafeState
+ SeverityClassKind
Requirements
+ HasAsilDecomposed
+ HasEmergencyOperationTimeInterval
+ HasFaultTolerantTimeInterval
+ RequirementKind
+ RequirementsReleationship
+ Verify
+ AbstractSafetyRequirement
+ AllocatableElement
+ AllocationTarget
+ FunctionalSafetyRequirement
+ RequirementsLink
+ RequirementsLinkType
+ Satisfy
+ SoftwareSafetyRequirement
+ TechnicalSafetyRequirement
+ TraceableSpecification
+ SoftwareSafetyRequirements
Software
+ Configuration
System

Figure: 1

Package "CommonStructure"

Type of Package: **Package**

Parent Package: SAFE Meta-Model

Notes:

Package "DataTypes"

Type of Package: **Package**

Parent Package: CommonStructure

Notes:

Diagram "**Datatypes**"

Notes:

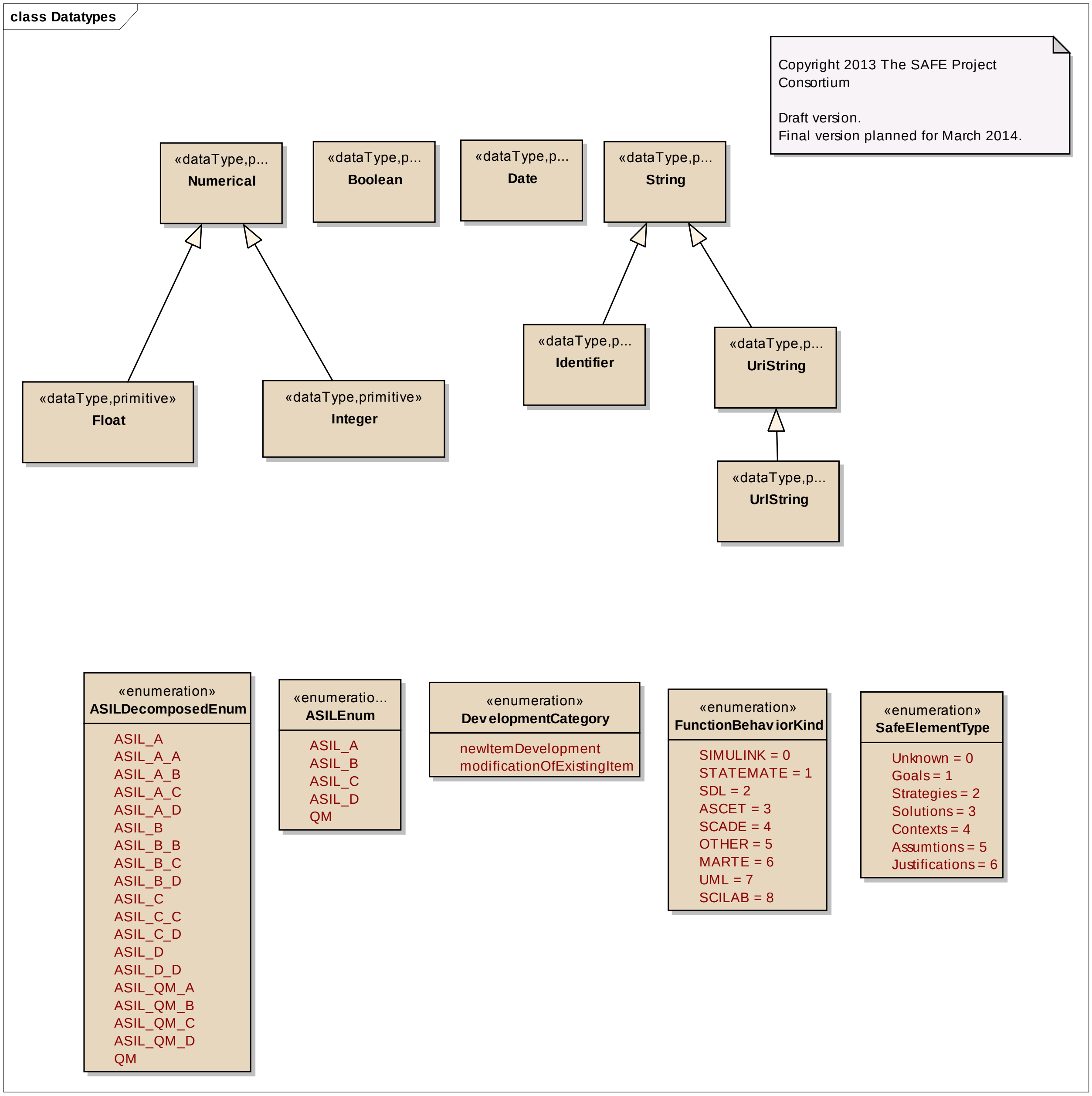


Figure: 2

Element "ASILDecomposedEnum"

Parent Package: DataTypes

Stereotype: «enumeration»,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	ASIL_A								
	ASIL_A_A								
	ASIL_A_B								
	ASIL_A_C								
	ASIL_A_D								
	ASIL_B								
	ASIL_B_B								
	ASIL_B_C								
	ASIL_B_D								
	ASIL_C								
	ASIL_C_C								
	ASIL_C_D								
	ASIL_D								
	ASIL_D_D								
	ASIL_QM_A								
	ASIL_QM_B								
	ASIL_QM_C								
	ASIL_QM_D								
	QM								

Element "ASILEnum"

Parent Package: DataTypes

Stereotype: «enumeration»,

Notes:

An ASIL shall be determined by using the parameters

- severity
- probability of exposure
- controllability

ISO 26262 Reference: Part 3 Chapter 7.4.4

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes

	ASIL_A								
	ASIL_B								
	ASIL_C								
	ASIL_D								
	QM								

Element "DevelopmentCategory"

Parent Package: DataTypes

Stereotype: «enumeration»,

Notes:

This element is an enumeration for the development kind of an item.

Values:

- new

- modification

ISO 262626 Reference: Part 3 Chapter 6.4.1

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	newItemDevelopment				0	0	0		
	modificationOfExistingItem				0	0	0		

Element "FunctionBehaviorKind"

Parent Package: DataTypes

Stereotype: «enumeration»,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	SIMULINK							0	
	STATEMATE							1	

	SDL							2	
	ASCET							3	
	SCADE							4	
	OTHER							5	
	MARTE							6	
	UML							7	
	SCILAB							8	

Element "SafeElementType"

Parent Package: DataTypes

Stereotype: «enumeration»,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	Unknown				0	0	0	0	
	Goals				0	0	0	1	
	Strategies				0	0	0	2	
	Solutions				0	0	0	3	
	Contexts				0	0	0	4	
	Assumtions				0	0	0	5	
	Justifications				0	0	0	6	

Element "Boolean"

Parent Package: DataTypes

Stereotype: «primitive»,

Notes:

Element "Date"

Parent Package: DataTypes

Stereotype: «primitive»,

Notes:

Element "Float"

Parent Package: DataTypes

Stereotype: «primitive»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: Float. **Target:** Numerical.	

Element "Identifier"

Parent Package: DataTypes

Stereotype: «primitive»,

Notes:

An Identifier is a string with a number of constraints on its appearance, satisfying the requirements typical programming languages define for their Identifiers.

It needs to start with a letter, may consist of letters, digits and underscore. It must not have two consecutive underscores (to support subsequent name mangling based on "__").

Relationships

Name	Source/Target	Notes
	Source: Identifier. **Target:** String.	

Element "Integer"

Parent Package: DataTypes

Stereotype: «primitive»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: Integer. **Target:** Numerical.	

Element "Numerical"

Parent Package: DataTypes

Stereotype: «primitive»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: Float. **Target:** Numerical.	
	Source: Integer. **Target:** Numerical.	

Element "String"

Parent Package: DataTypes

Stereotype: «primitive»,

Notes:

Any string.

This primitive type is redefined here to solve an issue with entreprise architect ecore generation.

Relationships

Name	Source/Target	Notes
	Source: UriString. **Target:** String.	
	Source: Identifier. **Target:** String.	

Element "UriString"

Parent Package: DataTypes

Stereotype: «primitive»,

Notes:

A Uniform Resource Identifier (URI), is a compact string of characters used to identify or name a resource.

Relationships

Name	Source/Target	Notes
	Source: UrlString. **Target:** UriString.	
	Source: UriString. **Target:** String.	

Element "UrlString"

Parent Package: DataTypes

Stereotype: «primitive»,

Notes:

A Uniform Resource Location

Relationships

Name	Source/Target	Notes
	Source: UrlString. **Target:** UriString.	

Package "FormulaExpression"

Type of Package: **Package**

Parent Package: CommonStructure

Notes:

Diagram "**FormulaExpression**"

Notes:

Figure: 3

Element "AtpFormulaExpressionString"

Parent Package: FormulaExpression

Stereotype: «atpMixedString»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: HWPMHFFormula. **Target:** AtpFormulaExpressionString.	
	Source: HWFailureClassContributionFormula. **Target:** AtpFormulaExpressionString.	
	Source: HWLambdaPartFormula. **Target:** AtpFormulaExpressionString.	
	Source: HWFMSingleContributionFormula. **Target:** AtpFormulaExpressionString.	
	Source: HWLatentFaultMetricFormula. **Target:** AtpFormulaExpressionString.	

Name	Source/Target	Notes
	Source: FormulaExpression. **Target:** **1** AtpFormulaExpressionString.atpMixedString	
	Source: HWSinglePointFaultMetricFormula. **Target:** AtpFormulaExpressionString.	
	Source: FailureLogicFormula. **Target:** AtpFormulaExpressionString.	

Element "FormulaExpression"

Parent Package: FormulaExpression

Stereotype: ,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	Formula	String							

Relationships

Name	Source/Target	Notes
	Source: FormulaExpression. **Target:** **1** AtpFormulaExpressionString.atpMixedString	
	Source: **1** FormulaExpression.value **Target:** **0..1** Filter.	The formula defining how the value of the filter shall be computed

Package "References"

Type of Package: **Package**

Parent Package: CommonStructure

Notes:

Diagram "**References**"

Notes:

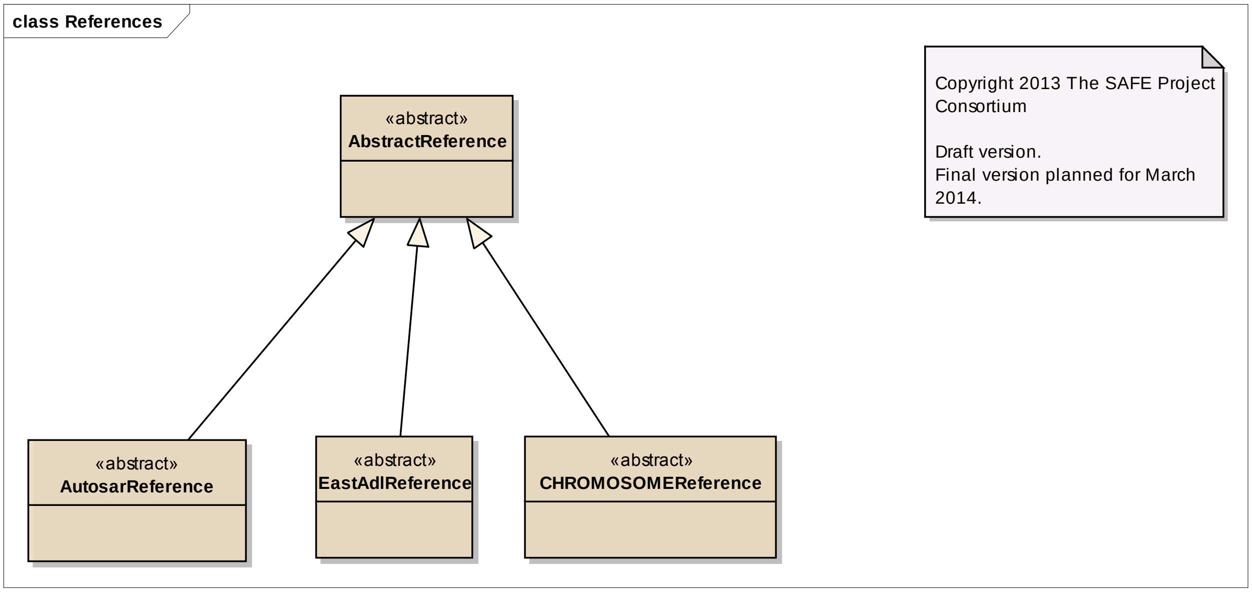


Figure: 4

Element "AbstractReference"

Parent Package: References

Stereotype: «abstract»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: CHROMOSOMEReference. **Target:** AbstractReference.	
	Source: EastAdlReference. **Target:** AbstractReference.	
	Source: AutosarReference. **Target:** AbstractReference.	

Element "AutosarReference"

Parent Package: References

Stereotype: «abstract»,

Notes:

The base reference element which allows references to AUTOSAR elements defined outside SAFE models.

Relationships

Name	Source/Target	Notes
	Source: VariableDataPrototype. **Target:** AutosarReference.	
	Source: HwElementPrototype. **Target:** AutosarReference.	
	Source: SwComponentType. **Target:** AutosarReference.	
	Source: ModeDeclaration. **Target:** AutosarReference.	
	Source: ComponentInCompositionInstanceRef. **Target:** AutosarReference.	
	Source: HwElementType. **Target:** AutosarReference.	
	Source: ModeDeclarationGroupPrototype. **Target:** AutosarReference.	
	Source: PortPrototype. **Target:** AutosarReference.	
	Source: RunnableEntity. **Target:** AutosarReference.	
	Source: CompositionSwComponentType. **Target:** AutosarReference.	
	Source: BswModuleEntry. **Target:** AutosarReference.	
	Source: System. **Target:** AutosarReference.	
	Source: AutosarReference. **Target:** AbstractReference.	
	Source: BswModuleDescription. **Target:** AutosarReference.	
	Source: ClientServerOperation. **Target:** AutosarReference.	

Element "CHROMOSOMEReference"

Parent Package: References

Stereotype: «abstract»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: Node. **Target:** CHROMOSOMEReference.	
	Source: System. **Target:** CHROMOSOMEReference.	
	Source: CHROMOSOMEReference. **Target:** AbstractReference.	
	Source: Application. **Target:** CHROMOSOMEReference.	
	Source: Component. **Target:** CHROMOSOMEReference.	
	Source: ComponentModeInstanceRef. **Target:** CHROMOSOMEReference.	
	Source: Topic. **Target:** CHROMOSOMEReference.	

Element "EastAdlReference"

Parent Package: References

Stereotype: «abstract»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: FunctionPort. **Target:** EastAdlReference.	

Name	Source/Target	Notes
	Source: HardwarePin. **Target:** EastAdlReference.	
	Source: AnalysisFunctionType. **Target:** EastAdlReference.	
	Source: EastAdlReference. **Target:** AbstractReference.	
	Source: HardwareComponentPrototype. **Target:** EastAdlReference.	
	Source: FunctionPrototype. **Target:** EastAdlReference.	
	Source: SystemModel. **Target:** EastAdlReference.	
	Source: Environment. **Target:** EastAdlReference.	
	Source: Requirement. **Target:** EastAdlReference.	
	Source: Item. **Target:** EastAdlReference.	
	Source: HardwareComponentType. **Target:** EastAdlReference.	
	Source: DesignFunctionType. **Target:** EastAdlReference.	

Package "CHROMOSOMEReferences"

Type of Package: **Package**

Parent Package: References

Notes:

Diagram "CHROMOSOMEReferences"

Notes:

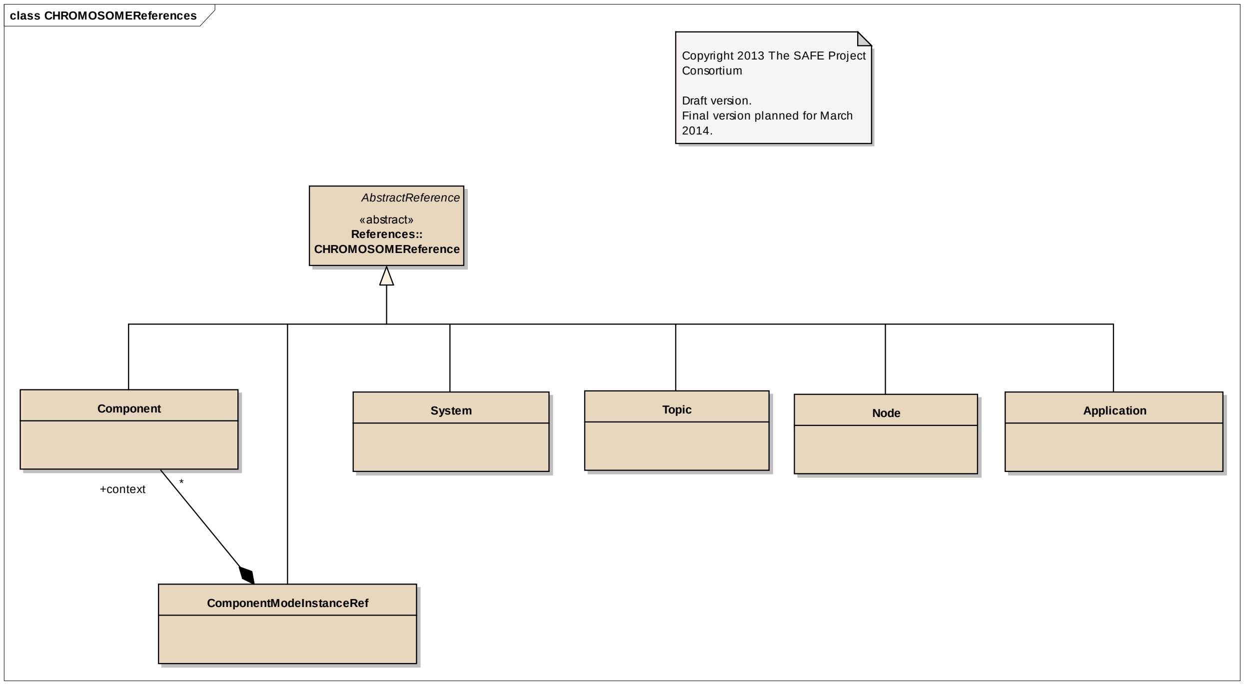


Figure: 5

Element "Application"

Parent Package: CHROMOSOMEReferences

Stereotype: ,

Notes:

A reference to a specific CHROMOSOME application.

Relationships

Name	Source/Target	Notes
	Source: 1 Application.monitors **Target:** 1 ChromosomeApplicationHealthMonitor.	
	Source: Application. **Target:** CHROMOSOMEReference.	
	Source: 1 Application.safeguards **Target:** ChromosomeApplicationSafetyExtension.	

Element "Component"

Parent Package: CHROMOSOMEReferences

Stereotype: ,

Notes:

A reference to a specific CHROMOSOME component instance.

Relationships

Name	Source/Target	Notes
	Source: **1** ChromosomeNodeHealthMonitor. **Target:** **1** Component.satisfiedBy	Generated CHROMOSOME component that satisfies the SSR.
	Source: **1** ChromosomeHeartbeatSender. **Target:** **1** Component.satisfiedBy	Generated CHROMOSOME component that satisfies the SSR.
	Source: **1** ChromosomeApplicationHealthMonitor. **Target:** **1** Component.satisfiedBy	Generated CHROMOSOME component that satisfies the SSR.
	Source: **1** ChromosomeMemoryTest. **Target:** **1** Component.satisfiedBy	A satisfy relation mapping MemorySelfTest SSRs to generated CHROMOSOME components.
	Source: **1** Component.supervises **Target:** ChromosomeHealthMonitor.	Defines the set of supervised CHROMOSOME components.
	Source: **1** ChromosomeCpuSelfTest. **Target:** **1** Component.satisfiedBy	Generated CHROMOSOME component that satisfies the SSR.
	Source: Component. **Target:** CHROMOSOMEReference.	
	Source: ***** Component.context **Target:** ComponentModeInstanceRef.	
	Source: **1** ChromosomeHeartbeatReceiver. **Target:** **1** Component.satisfiedBy	Generated CHROMOSOME component that satisfies the SSR.
	Source: **1** ChromosomeVoting.	Generated CHROMOSOME

Name	Source/Target	Notes
	Target: **1** Component.satisfiedBy	component that satisfies the SSR implementing SSM.
	Source: **1** Component.target **Target:** ChromosomeComparison.	

Element "ComponentModeInstanceRef"

Parent Package: CHROMOSOMEReferences

Stereotype: ,

Notes:

Enables referencing of mode of a specific CHROMOSOME component instance.

Relationships

Name	Source/Target	Notes
	Source: **1** ComponentModeInstanceRef.triggeredBy **Target:** **1** ChromosomeComponentModeCondition.	Specifies a set of modes of CHROMOSOME component instances, which triggers the component mode condition.
	Source: ComponentModeInstanceRef. **Target:** CHROMOSOMEReference.	
	Source: ***** Component.context **Target:** ComponentModeInstanceRef.	

Element "Node"

Parent Package: CHROMOSOMEReferences

Stereotype: ,

Notes:

A reference to a CHROMOSOME node. A node in CHROMOSOME is a hardware unit, which acts as a deployment target for software components (like ECU in Autosar).

Relationships

Name	Source/Target	Notes

Name	Source/Target			Notes
	Source:		Node.	
	Target:		CHROMOSOMEReference.	
	Source:	**1**	Node.checks	
	Target:	**1**	ChromosomeCpuSelfTest.	
	Source:	**1**	Node.tests	
	Target:	**1..***	ChromosomeMemoryTest.	
	Source:	**1**	Node.monitors	
	Target:	**1**	ChromosomeNodeHealthMonitor.	

Element "System"

Parent Package: CHROMOSOMEReferences

Stereotype: ,

Notes:

Relationships

Name	Source/Target			Notes
	Source:		System.	
	Target:		CHROMOSOMEReference.	
	Source:	**1**	System.safeguards	
	Target:		ChromosomeSystemSafetyExtension.	

Element "Topic"

Parent Package: CHROMOSOMEReferences

Stereotype: ,

Notes:

A reference to a specific CHROMOSOME topic.

Relationships

Name	Source/Target			Notes
	Source:	**1**	Topic.target	
	Target:		ChromosomeComparisonParameter.	
	Source:	**1**	Topic.target	References the topic instances,

Name	Source/Target	Notes
	Target: 1..* ChromosomeVotingParameter.	which represent the data for voting and output values
	Source: Topic. **Target:** CHROMOSOMEReference.	
	Source: 1 Topic.triggeredBy **Target: 1** ChromosomeErrorEventCondition.	Includes a reference to a CHROMOSOME topic used to transport error notifications from different componens / subsystems, and acting as a trigger for error condition.
	Source: 1 Topic.publishedVia **Target: 1** CHROMOSOMEHealthMonitorNotification.	Includes a reference to a CHROMOSOME topic used to transport error notifications.

Package "EASTADLReferences"

Type of Package: **Package**

Parent Package: References

Notes:

Diagram "EASTADLReferences"

Notes:

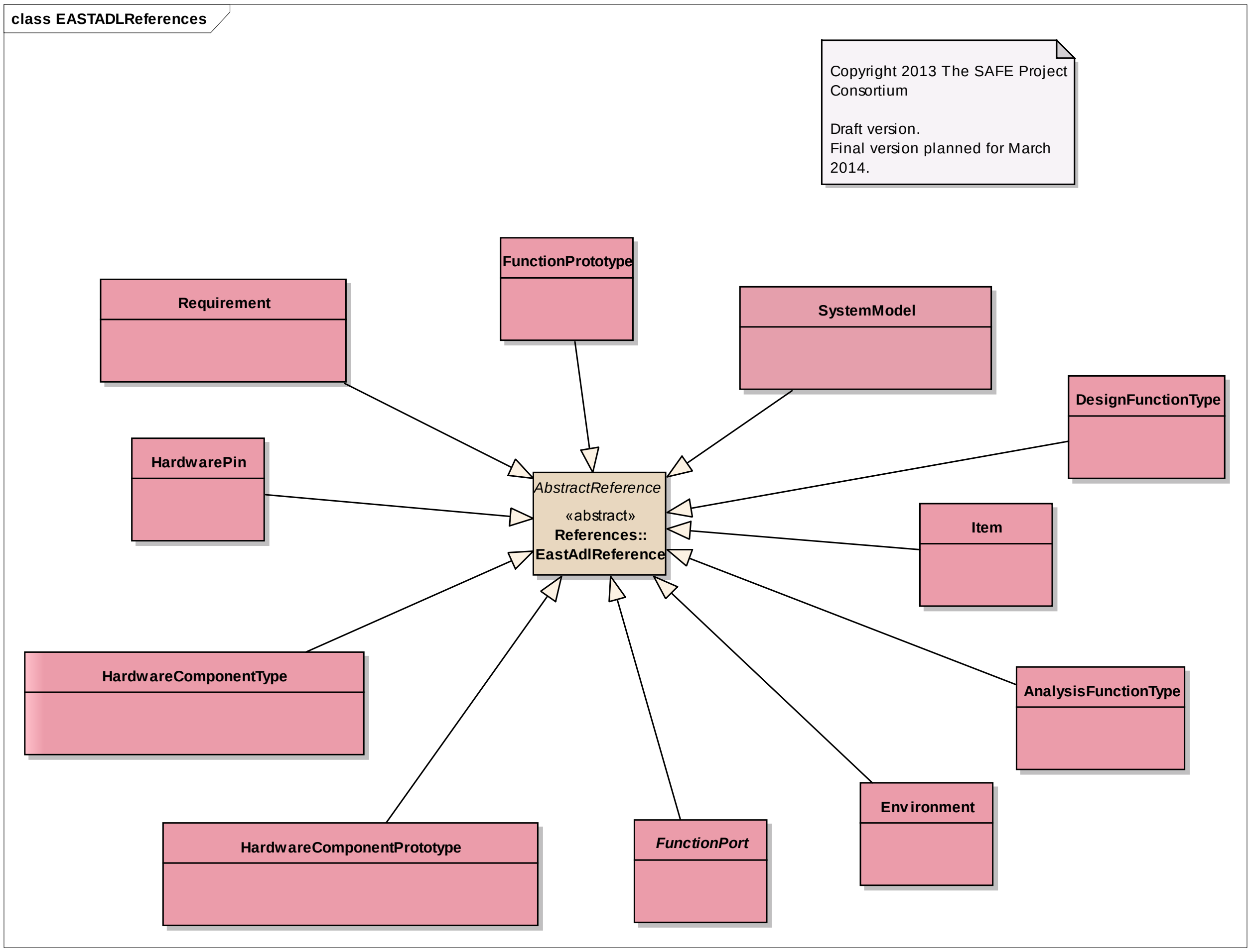


Figure: 6

Element "AnalysisFunctionType"

Parent Package: EASTADLReferences

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: AnalysisFunctionType. **Target:** EastAdlReference.	
	Source: **1** AnalysisFunctionType.scope **Target:** FunctionalSafetyExtension.	

Element "DesignFunctionType"

Parent Package: EASTADLReferences

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: **1** DesignFunctionType.scope **Target:** SoftwareSafetyDesign.	
	Source: DesignFunctionType. **Target:** EastAdlReference.	

Element "Environment"

Parent Package: EASTADLReferences

Stereotype: ,

Notes:

Class used to reference the EAST-ADL Environment Element

Environmental Elements describe elements that have the potential to influence the vehicle behavior during the analyzed operational situation.

(e.g. main road, trees next to the road, buildings next to the road, snow,...)

Relationships

Name	Source/Target	Notes
	Source: OperationalSituation. **Target:** **0..1** Environment.environment	
	Source: **1..*** Environment.environmentalElement **Target:** HazardandRiskSafetyExtension.	
	Source: Environment. **Target:** EastAdlReference.	

Element "FunctionPort"

Parent Package: EASTADLReferences

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: MFPFunctionPort. **Target:** **0..1** FunctionPort.functionTarget	
	Source: FunctionPort. **Target:** EastAdlReference.	
	Source: **1** FunctionPort.target **Target:** FaultFailurePort_functionTarget.	

Element "FunctionPrototype"

Parent Package: EASTADLReferences

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: **0..*** FunctionPrototype.context **Target:** FaultFailurePort_functionTarget.	
	Source: FunctionPrototype. **Target:** EastAdlReference.	
	Source: **1** FunctionPrototype.target **Target:** ErrorModelPrototype_functionTarget.	
	Source: EMPFunction. **Target:** **0..1** FunctionPrototype.functionTarget	the target function instance
	Source: **0..*** FunctionPrototype.context **Target:** ErrorModelPrototype_functionTarget.	

Element "HardwareComponentPrototype"

Parent Package: EASTADLReferences

Stereotype: ,

Notes:

Class used to reference an EAST-ADL HardwareComponentPrototype element.

Relationships

Name	Source/Target	Notes
	Source: HardwareComponentPrototype. **Target:** EastAdlReference.	
	Source: EMPHwComponent. **Target:** **1** HardwareComponentPrototype.hwTarget	
	Source: HwComponentScopeInstanceRef. **Target:** **0..1** HardwareComponentPrototype.contextHwComponentScope	
	Source: **1** HardwareComponentPrototype.target **Target:** ErrorModelPrototype_hwTarget.	
	Source: **0..*** HardwareComponentPrototype.context **Target:** ErrorModelPrototype_hwTarget.	
	Source: HwComponentScopeInstanceRef. **Target:** **0..1** HardwareComponentPrototype.targetHwComponentScope	

Element "HardwareComponentType"

Parent Package: EASTADLReferences

Stereotype: ,

Notes:

Class used to reference an EAST-ADL HardwareComponentType element.

Relationships

Name	Source/Target	Notes
	Source: **1** HardwareComponentType.scope **Target:** EMTypeHwComponent.	the target hardware component
	Source: HwComponentScopeInstanceRef. **Target:** **0..1** HardwareComponentType.baseHwcomponentScope	
	Source: **1** HardwareComponentType.scope	

Name	Source/Target	Notes
	Target: HardwareSafetyDesign.	
	Source: HardwareComponentType. **Target:** EastAdlReference.	

Element "HardwarePin"

Parent Package: EASTADLReferences

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: HardwarePin. **Target:** EastAdlReference.	
	Source: 1 HardwarePin.target **Target:** FaultFailurePort_hwTarget.	
	Source: MFPHardwarePin. **Target: 0..1** HardwarePin.hwTarget	

Element "Item"

Parent Package: EASTADLReferences

Stereotype: ,

Notes:

class used to reference the EAST-ADL item element

Relationships

Name	Source/Target	Notes
	Source: Item. **Target:** EastAdlReference.	

Element "Requirement"

Parent Package: EASTADLReferences

Stereotype: ,

Notes:

Class used to reference an EAST-ADL Requirement element.

Relationships

Name	Source/Target	Notes
	Source: Requirement. **Target:** EastAdlReference.	
	Source: AbstractSafetyRequirement. **Target:** **0..1** Requirement.requirement	

Element "SystemModel"

Parent Package: EASTADLReferences

Stereotype: ,

Notes:

Class used to reference an EAST-ADL SystemModel element.

Relationships

Name	Source/Target	Notes
	Source: SystemModel. **Target:** EastAdlReference.	

Package "AUTOSARReferences"

Type of Package: **Package**

Parent Package: References

Notes:

Diagram "AUTOSARReferences"

Notes:

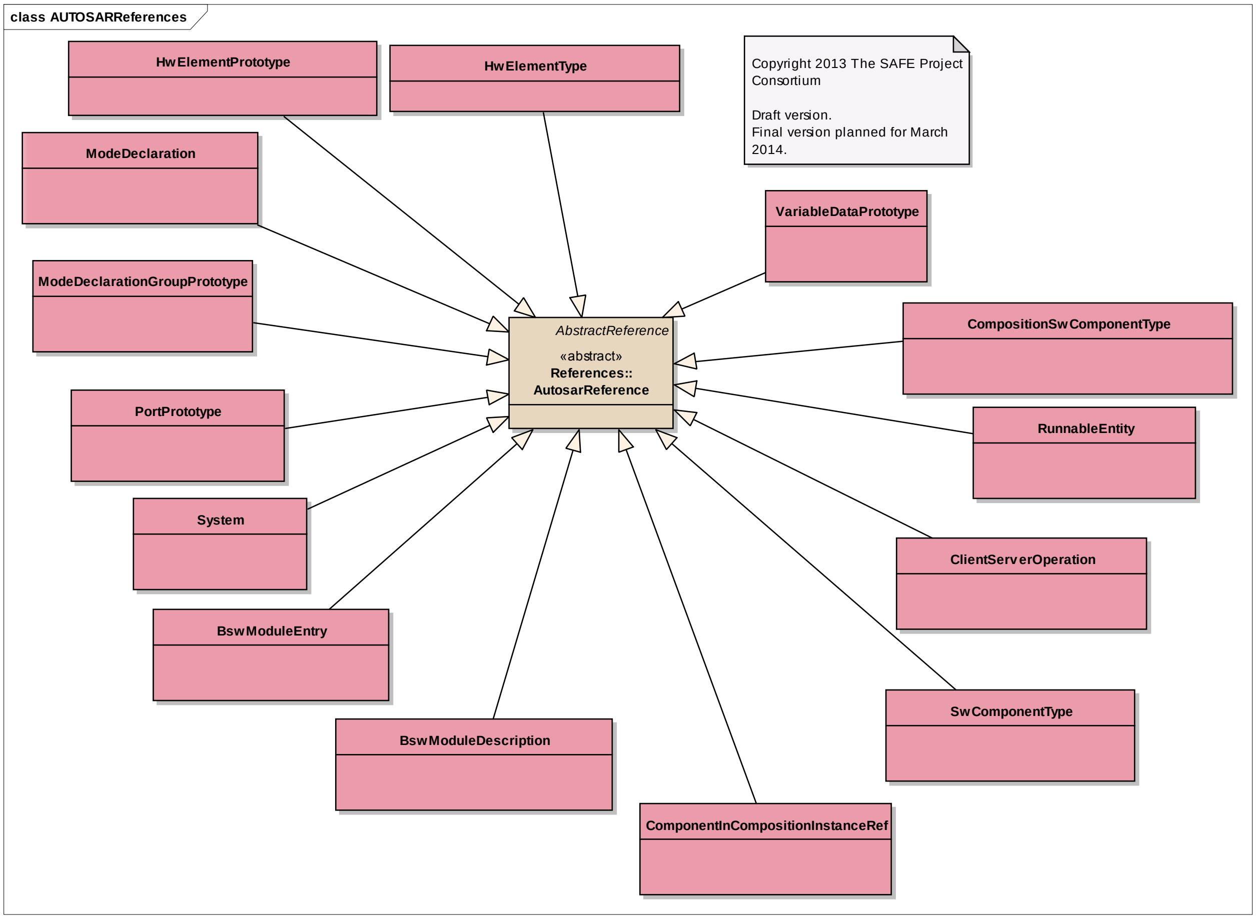


Figure: 7

Element "BswModuleDescription"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: **1** BswModuleDescription.bswTarget **Target:** EMPBswModule.	The target basic software module
	Source: BswModuleDescription. **Target:** AutosarReference.	
	Source: **1** BswModuleDescription.scope	the target basic

Name	Source/Target		Notes
	Target:	EMTypeBswModule.	software module

Element "BswModuleEntry"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

Relationships

Name	Source/Target		Notes
	Source: **Target:**	BswModuleEntry. AutosarReference.	
	Source: 1 **Target:**	BswModuleEntry.bswEntry MFPBswPort.	the target bsw module entry

Element "ClientServerOperation"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

Relationships

Name	Source/Target		Notes
	Source: 1 **Target:**	ClientServerOperation.operation MFPOperation.	the target operation prototype instance
	Source: **Target:**	ClientServerOperation. AutosarReference.	

Element "ComponentInCompositionInstanceRef"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

A reference to an AUTOSAR software component within a software component composition instance which finds itself in another model / file.

Relationships

Name	Source/Target	Notes
	Source: **1** ComponentInCompositionInstanceRef.component **Target:** AutosarCheckPoint.	The target component containing the internal behavior whose runnable entity is monitored
	Source: **1** ComponentInCompositionInstanceRef.actuator **Target:** AutosarActuatorMonitor.	The instance reference to the monitored actuator component in AUTOSAR
	Source: ComponentInCompositionInstanceRef. **Target:** AutosarReference.	
	Source: **1** ComponentInCompositionInstanceRef.swcTarget **Target:** EMPSwComponent.	the target software component instance

Element "CompositionSwComponentType"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

Class used to reference instances of software component compositions in an AUTOSAR model.

Relationships

Name	Source/Target	Notes
	Source: **1** CompositionSwComponentType.safeguards **Target:** AutosarVfbSafetyExtension.	The AUTOSAR root software component safeguarded by this safety extension.
	Source: CompositionSwComponentType.	

Name	Source/Target	Notes
	Target: AutosarReference.	

Element "HwElementPrototype"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

class used to reference an AUTOSAR-Element used to model a safety hardware component

Relationships

Name	Source/Target	Notes
	Source: HwElementPrototype. **Target:** AutosarReference.	
	Source: HWElementScopeInstanceRef.instanceRef.context **Target:** **0..1** HwElementPrototype.ContextHwElementScope	
	Source: HWElementScopeInstanceRef.instanceRef.target **Target:** **1** HwElementPrototype.targetHwElementScope	

Element "HwElementType"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: HWElementScopeInstanceRef. **Target:** **0..1** HwElementType.baseHardwareElementScope	
	Source: HwElementType. **Target:** AutosarReference.	
	Source: **1** HwElementType.scope **Target:** HardwareImplementationSafetyExtension.	

Element "ModeDeclaration"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

Class used to reference an AUTOSAR mode declaration.

Relationships

Name	Source/Target	Notes
	Source: 1 ModeDeclaration.modeDeclaration **Target: 1** AutosarContext.	The mode used to define a context based range
	Source: ModeDeclaration. **Target:** AutosarReference.	

Element "ModeDeclarationGroupPrototype"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

A reference to an AUTOSAR mode for a specific port of a specific software component within a software component composition defined in another model / file.

Relationships

Name	Source/Target	Notes
	Source: 1 ModeDeclarationGroupPrototype.context **Target: 1** AutosarContext.	The instance of a mode group declaration whose modes are used to define a value range context
	Source: ModeDeclarationGroupPrototype. **Target:** AutosarReference.	

Element "PortPrototype"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

Class used to reference AUTOSAR port prototype elements.

Relationships

Name	Source/Target	Notes
	Source: PortPrototype. **Target:** AutosarReference.	

Element "RunnableEntity"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

Class used to reference instances of AUTOSAR runnable entity elements.

Relationships

Name	Source/Target	Notes
	Source: 1 RunnableEntity.targetRunnable **Target:** AutosarCheckPoint.	The target runnable entity to be observed
	Source: RunnableEntity. **Target:** AutosarReference.	

Element "SwComponentType"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: 1 SwComponentType.scope **Target:** SoftwareImplementationSafetyExtension.	
	Source: 1 SwComponentType.scope **Target:** EMTypeSwComponent.	the target software component
	Source: SwComponentType. **Target:** AutosarReference.	

Element "System"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

Class used to reference an instance of an AUTOSAR System element.

Relationships

Name	Source/Target	Notes
	Source: 1 System.safeguards **Target:** AutosarSystemSafetyExtension.	The AUTOSAR system element safeguarded by this safety extension
	Source: System. **Target:** AutosarReference.	

Element "VariableDataPrototype"

Parent Package: AUTOSARReferences

Stereotype: ,

Notes:

A reference to an AUTOSAR variable data prototype of a specific software component within a software component composition which is defined in another model / file.

Relationships

Name	Source/Target	Notes
	Source: 1 VariableDataPrototype.checks **Target:** ComponentPrototypeCRC.	The AUTOSAR variable data prototype of a software component whose value is check using the CRC SSR.
	Source: 1 VariableDataPrototype.variable **Target:** MFPVariable.	the target variable data prototype instance
	Source: VariableDataPrototype. **Target:** AutosarReference.	
	Source: 1 VariableDataPrototype.checks **Target:** InterfaceCRC.	The AUTOSAR variable data prototype of an

Name	Source/Target	Notes
		interface whose value is check using the CRC SSR.
	Source: **1** VariableDataPrototype.checked **Target:** **1** AutosarContextRangeCheck.	The variable data prototype of an AUTOSAR software component whose range is monitored
	Source: ***** VariableDataPrototype.parameter **Target:** AutosarEvaluationFunction.	
	Source: **1** VariableDataPrototype.observed **Target:** AutosarGradientCheck.	The AUTOSAR variable data prototype instance to be monitored by the gradient check
	Source: **1** VariableDataPrototype.target **Target:** AutosarComparisonParam.	The target variable data prototype of an AUTOSAR software component which provides a value for the comparison operation.

Package "SafetyExtensions"

Type of Package: **Package**

Parent Package: CommonStructure

Notes:

Diagram "SafetyExtensions"

Notes:

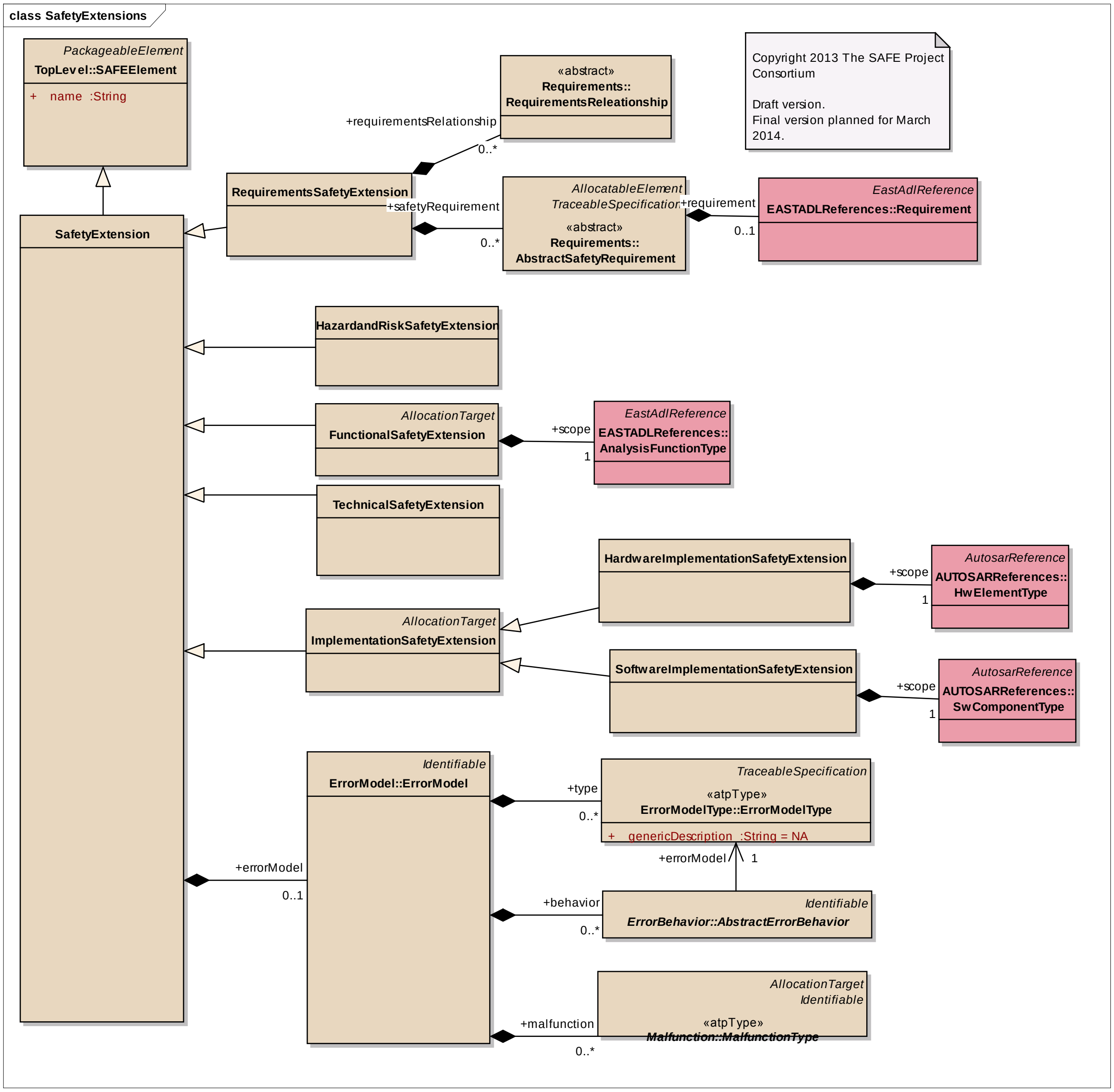


Figure: 8

Element "FunctionalSafetyExtension"

Parent Package: SafetyExtensions

Stereotype: ,

Notes:

The FunctionalSafetyExtension is used as interface to the AnalysisLevel defined in EAST-ADL. This extension specifies the add-on needed to model the functional safety concept defined in the ISO 26262 part 3 chapter 8.

Relationships

Name	Source/Target	Notes
	Source: FunctionalSafetyExtension. **Target:** AllocationTarget.	
	Source: **1** AnalysisFunctionType.scope **Target:** FunctionalSafetyExtension.	
	Source: FunctionalSafetyExtension. **Target:** SafetyExtension.	

Element "HardwareImplementationSafetyExtension"

Parent Package: SafetyExtensions

Stereotype: ,

Notes:

This class represent the Safety Extension point for referenced element of Hardware Element as part of a component (respectively AUTOSAR HW Element Type) to allow the capture of hardware failure and summary failure quantified contribution to HWComponent.

Relationships

Name	Source/Target	Notes
	Source: HardwareImplementationSafetyExtension. **Target:** **0..*** HWPartFailureAnalysis.hardwarePartSafetyAnalysis	
	Source: **0..1** HWPartFailure.ramdomHardwarePartFailure **Target:** HardwareImplementationSafetyExtension.	
	Source: HardwareImplementationSafetyExtension. **Target:** ImplementationSafetyExtension.	
	Source: **1** HwElementType.scope **Target:** HardwareImplementationSafetyExtension.	

Element "HazardandRiskSafetyExtension"

Parent Package: SafetyExtensions

Stereotype: ,

Notes:

The HazardAndRiskSafetyExtension is used as interface to the VehicleFeature defined in EAST-ADL. This extension specifies the add-on needed to model the hazard analysis and risk assessment defined in ISO 26262 part 3 chapter 7. Further details according to modeling of hazards and safety goals are described in D3.1.1.b

Relationships

Name	Source/Target	Notes
	Source: HazardandRiskSafetyExtension. **Target:** SafetyExtension.	
	Source: **1..*** Environment.environmentalElement **Target:** HazardandRiskSafetyExtension.	
	Source: **0..*** Hazard.hazard **Target:** **0..1** HazardandRiskSafetyExtension.	
	Source: **0..*** ControllabilityReference.controllability **Target:** HazardandRiskSafetyExtension.	
	Source: **0..*** Actor.actor **Target:** HazardandRiskSafetyExtension.	
	Source: **0..*** RiskDescription.risk **Target:** **0..1** HazardandRiskSafetyExtension.	

Element "ImplementationSafetyExtension"

Parent Package: SafetyExtensions

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: ImplementationSafetyExtension. **Target:** SafetyExtension.	
	Source: HardwareImplementationSafetyExtension. **Target:** ImplementationSafetyExtension.	
	Source: SoftwareImplementationSafetyExtension. **Target:** ImplementationSafetyExtension.	
	Source: ImplementationSafetyExtension. **Target:** AllocationTarget.	

Element "RequirementsSafetyExtension"

Parent Package: SafetyExtensions

Stereotype: ,

Notes:

Container for safety-requirements related elements.

Relationships

Name	Source/Target	Notes
	Source: RequirementsSafetyExtension. **Target:** SafetyExtension.	
	Source: **0..*** RequirementsReleationship.requirementsRelationship **Target:** RequirementsSafetyExtension.	
	Source: **0..*** AbstractSafetyRequirement.safetyRequirement **Target:** RequirementsSafetyExtension.	

Element "SafetyExtension"

Parent Package: SafetyExtensions

Stereotype: ,

Notes:

The abstract parent class of the different abstraction-level specific safety extensions.

Depending on the specific level of abstraction (HazardRiskModel, FunctionalSafetyExtension, TechnicalSafetyExtension, ...), the following restriction apply:

- the subtypes of AbstractSafetyRequirement (TechnicalSafetyRequirement, FunctionalSafetyRequirement, SafetyGoal, HW/SWSafetyRequirement) shall be used in the respective level of abstraction via the "requirements" relation of the safety extension

- the error model associated with the safety extension via the "erroModel" relation shall only allow to reference system model artifacts which are visibly at the level of abstraction of the safety extension (e.g. an AUTOSAR software component is not visible in the HazardandRiskModel)

Relationships

Name	Source/Target	Notes
	Source: ImplementationSafetyExtension. **Target:** SafetyExtension.	
	Source: HazardandRiskSafetyExtension.	

Name	Source/Target	Notes
	Target: SafetyExtension.	
	Source: RequirementsSafetyExtension. **Target:** SafetyExtension.	
	Source: SafetyExtension. **Target:** SAFEElement.	
	Source: TechnicalSafetyExtension. **Target:** SafetyExtension.	
	Source: FunctionalSafetyExtension. **Target:** SafetyExtension.	
	Source: **0..1** ErrorModel.errorModel **Target:** SafetyExtension.	An error model associated with the respective safety extension. The error model is valid in the context of this safety extension.

Element "SoftwareImplementationSafetyExtension"

Parent Package: SafetyExtensions

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: **1** SwComponentType.scope **Target:** SoftwareImplementationSafetyExtension.	
	Source: **0..*** CodeGenerationConfiguration.codeGenerationConfiguration **Target:** SoftwareImplementationSafetyExtension.	
	Source: SoftwareImplementationSafetyExtension. **Target:** ImplementationSafetyExtension.	

Element "TechnicalSafetyExtension"

Parent Package: SafetyExtensions

Stereotype: ,

Notes:

The TechnicalSafetyExtension is used as interface to the DesignLevel defined in EAST-ADL. This extension specifies the add-on needed to model a specific technical solution that is derived based on the functional safety concept. It contains the

- technical safety concept (ISO 26262 part 4 chapter 7)
- hardware software interface specification (ISO 26262 part 4 chapter 7.4.6)

Relationships

Name	Source/Target	Notes
	Source: TechnicalSafetyExtension. **Target:** SafetyExtension.	
	Source: **0..*** SoftwareSafetyDesign.softwareArchitecture **Target:** TechnicalSafetyExtension.	
	Source: **0..*** HardwareSafetyDesign.hardwareArchitecture **Target:** TechnicalSafetyExtension.	

Package "SoftwareSafetyExtension"

Type of Package: **Package**

Parent Package: SafetyExtensions

Notes:

Diagram "**SoftwareSafetyExtension**"

Notes:

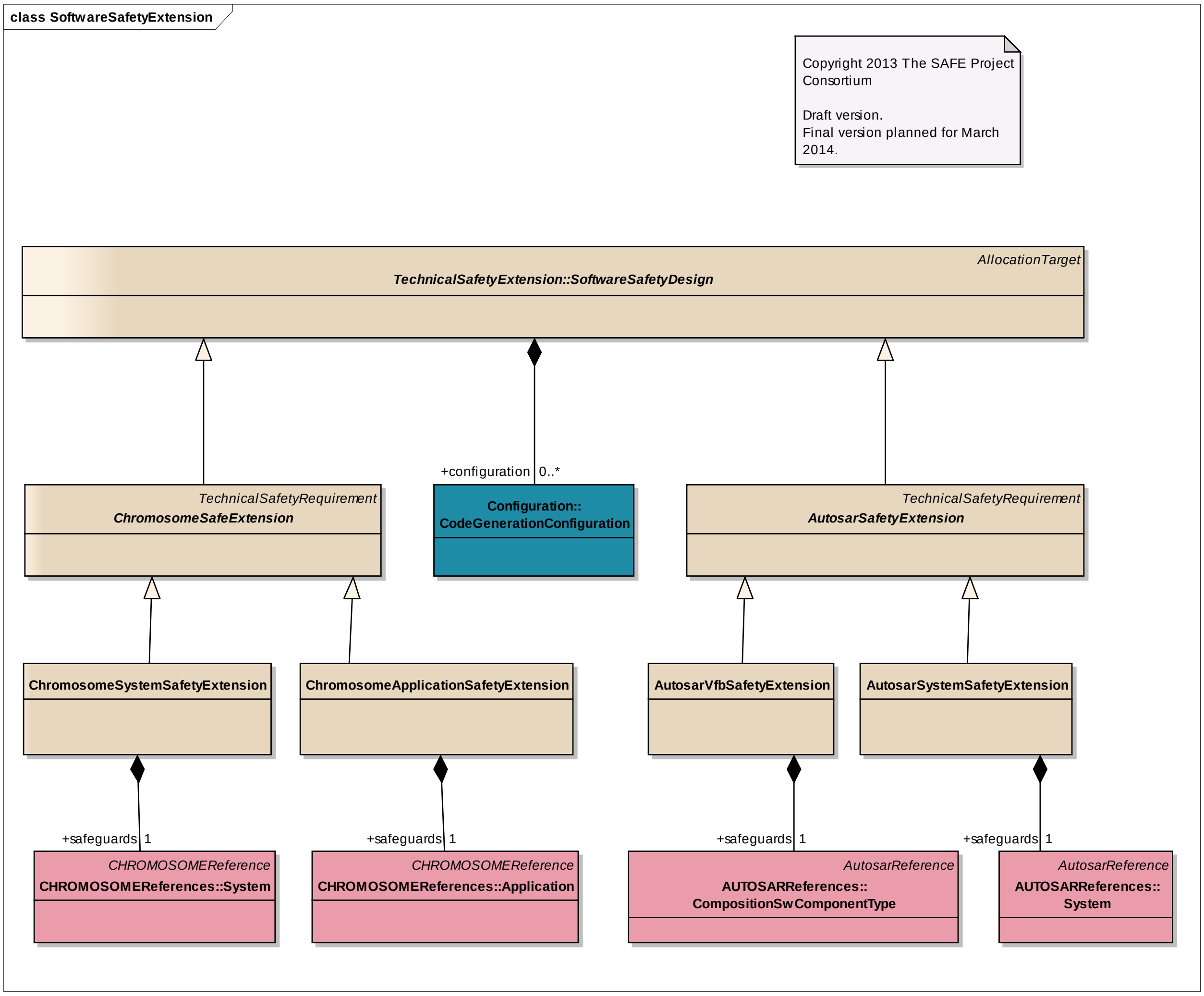


Figure: 9

Element "AutosarSafetyExtension"

Parent Package: SoftwareSafetyExtension

Stereotype: ,

Notes:

This element represents abstract AUTOSAR SAFE extensions. These are extensions to AUTOSAR which define software safety mechanisms related to AUTOSAR elements.

Relationships

Name	Source/Target	Notes
	Source: AutosarSafetyExtension. **Target:** TechnicalSafetyRequirement.	

Name	Source/Target	Notes
	Source: AutosarVfbSafetyExtension. **Target:** AutosarSafetyExtension.	
	Source: AutosarSafetyExtension. **Target:** SoftwareSafetyDesign.	
	Source: AutosarSystemSafetyExtension. **Target:** AutosarSafetyExtension.	

Element "AutosarSystemSafetyExtension"

Parent Package: SoftwareSafetyExtension

Stereotype: ,

Notes:

This meta-class defines SAFE extensions which are specified for the AUTOSAR system level.

Relationships

Name	Source/Target	Notes
	Source: **1** System.safeguards **Target:** AutosarSystemSafetyExtension.	The AUTOSAR system element safeguarded by this safety extension
	Source: AutosarSystemSafetyExtension. **Target:** AutosarSafetyExtension.	

Element "AutosarVfbSafetyExtension"

Parent Package: SoftwareSafetyExtension

Stereotype: ,

Notes:

This meta-class defines the element used for specifications of SAFE extensions at VFB level on AUTOSAR. This means elements which relate to a composition directly, not yet in a given system context.

Relationships

Name	Source/Target	Notes
	Source: **1** CompositionSwComponentType.safeguards **Target:** AutosarVfbSafetyExtension.	The AUTOSAR root software component safeguarded by

Name	Source/Target	Notes
		this safety extension.
	Source: AutosarVfbSafetyExtension. **Target:** AutosarSafetyExtension.	

Element "ChromosomeApplicationSafetyExtension"

Parent Package: SoftwareSafetyExtension

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: **1** Application.safeguards **Target:** ChromosomeApplicationSafetyExtension.	
	Source: ChromosomeApplicationSafetyExtension. **Target:** ChromosomeSafeExtension.	

Element "ChromosomeSafeExtension"

Parent Package: SoftwareSafetyExtension

Stereotype: ,

Notes:

This element represents abstract Chromosome SAFE extensions.

Relationships

Name	Source/Target	Notes
	Source: ChromosomeSafeExtension. **Target:** SoftwareSafetyDesign.	
	Source: ChromosomeSystemSafetyExtension. **Target:** ChromosomeSafeExtension.	
	Source: ChromosomeSafeExtension. **Target:** TechnicalSafetyRequirement.	
	Source: ChromosomeApplicationSafetyExtension.	

Name	Source/Target	Notes
	Target: ChromosomeSafeExtension.	

Element "ChromosomeSystemSafetyExtension"

Parent Package: SoftwareSafetyExtension

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: ChromosomeSystemSafetyExtension. **Target:** ChromosomeSafeExtension.	
	Source: 1 System.safeguards **Target:** ChromosomeSystemSafetyExtension.	

Package "TechnicalSafetyExtension"

Type of Package: **Package**

Parent Package: SafetyExtensions

Notes:

This package describes the TechnicalSafetyExtension as defined in the SafetyExtension-Diagram.

Diagram "TechnicalSafetyExtension"

Notes:

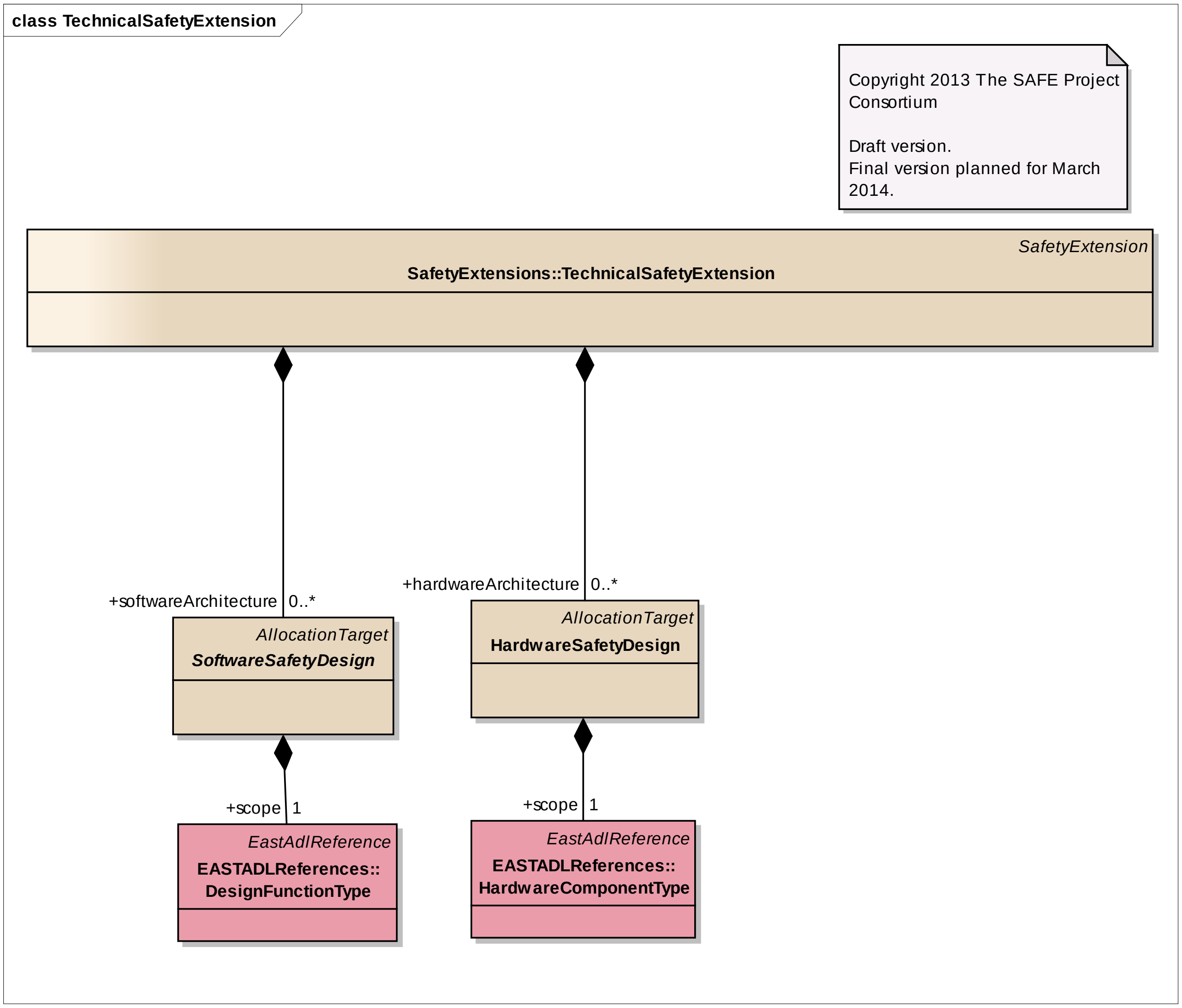


Figure: 10

Element "HardwareSafetyDesign"

Parent Package: TechnicalSafetyExtension

Stereotype: ,

Notes:

This class represent the Safety Extension point for referenced element of Hardware Component Type (respectively from EAST-ADL) to allow the capture of hardware failure and summary failure and analysis results.

Relationships

Name	Source/Target	Notes
	Source: HardwareComponentFailure.randomHarwareFailure **0..1**	

Name	Source/Target	Notes
	Target: HardwareSafetyDesign.	
	Source: HardwareSafetyDesign. **Target:** AllocationTarget.	
	Source: **0..*** HardwareFailureAnalysis.harwdareSafetyAnalysis **Target:** HardwareSafetyDesign.	
	Source: **1** HardwareComponentType.scope **Target:** HardwareSafetyDesign.	
	Source: **0..*** HardwareSafetyDesign.hardwareArchitecture **Target:** TechnicalSafetyExtension.	

Element "SoftwareSafetyDesign"

Parent Package: TechnicalSafetyExtension

Stereotype: ,

Notes:

The SafetySoftwareExtension is used to specify the implementation of the safety relevant SoftwareDesignComponents that are allocated to the SafetySoftwareDesign.

Relationships

Name	Source/Target	Notes
	Source: **1** DesignFunctionType.scope **Target:** SoftwareSafetyDesign.	
	Source: ChromosomeSafeExtension. **Target:** SoftwareSafetyDesign.	
	Source: SoftwareSafetyDesign. **Target:** AllocationTarget.	
	Source: **0..*** CodeGenerationConfiguration.configuration **Target:** SoftwareSafetyDesign.	Configurations contained in the SafeExtension.
	Source: AutosarSafetyExtension. **Target:** SoftwareSafetyDesign.	
	Source: **0..*** SoftwareSafetyDesign.softwareArchitecture **Target:** TechnicalSafetyExtension.	

Package "TopLevel"

Type of Package: **Package**

Parent Package: CommonStructure

Notes:

Diagram "TopLevel"

Notes:

Top-Level structure is built up similar to the TopLevelStructure defined in AUTOSAR.

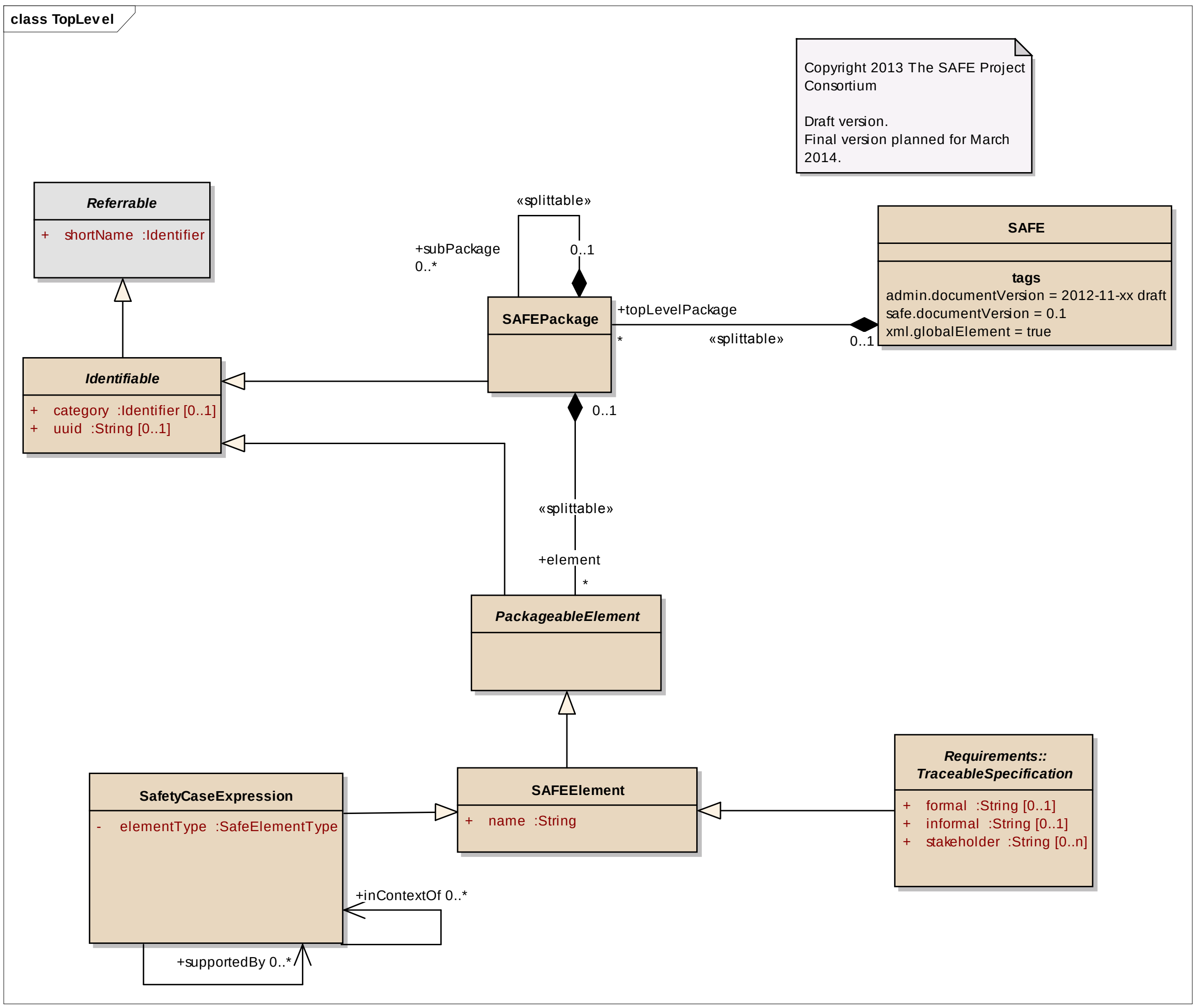


Figure: 11

Element "PackageableElement"

Parent Package: TopLevel

Stereotype: ,

Notes:

This meta-class specifies the ability to be a member of a SAFE package.

Relationships

Name	Source/Target	Notes
	Source: Restrictable. **Target:** PackageableElement.	
	Source: * PackageableElement.element **Target:** 0..1 SAFEPackage.	
	Source: SAFEElement. **Target:** PackageableElement.	
	Source: PackageableElement. **Target:** Identifiable.	

Element "Referrable"

Parent Package: TopLevel

Stereotype: ,

Notes:

Instances of this class can be referred to by their identifier (while adhering to namespace borders).

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	shortName	Identifier			0	0	0		This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference.

Relationships

Name	Source/Target	Notes
	Source: Identifiable. **Target:** Referrable.	
	Source: HWFault.	

Name	Source/Target	Notes
	Target: Referrable.	
	Source: HWFailureMode. **Target:** Referrable.	
	Source: HWComponentQuantifiedFMFromPart. **Target:** Referrable.	
	Source: HWFailureRate. **Target:** Referrable.	

Element "SAFE"

Parent Package: TopLevel

Stereotype: ,

Notes:

The root element of a SAFE description, also the root element in corresponding XML documents.

Relationships

Name	Source/Target	Notes
	Source: ***** SAFEPackage.topLevelPackage **Target:** **0..1** SAFE.	

Element "SAFEElement"

Parent Package: TopLevel

Stereotype: ,

Notes:

This class serves as a base class for all SAFE class that represent something (i.e. not technical class).

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	name	String							Optional descriptive name of the SAFEElement, this name does not have the length restrictions as found for the AUTOSAR Identfiable shortName.

Relationships

Name	Source/Target	Notes
	Source: SafetyCaseExpression. **Target:** SAFEElement.	
	Source: SafetyExtension. **Target:** SAFEElement.	
	Source: TraceableSpecification. **Target:** SAFEElement.	
	Source: SAFEElement. **Target:** PackageableElement.	

Element "SAFEPackage"

Parent Package: TopLevel

Stereotype: ,

Notes:

Used for organization of the packageable elements in the model.

Semantics:

SAFEPackages can be organized hierarchically, where each level may contain a number of SAFEPackageableElements.

Relationships

Name	Source/Target	Notes
	Source: SAFEPackage. **Target:** Identifiable.	
	Source: ***** PackageableElement.element **Target:** **0..1** SAFEPackage.	
	Source: ***** SAFEPackage.topLevelPackage **Target:** **0..1** SAFE.	
	Source: **0..1** SAFEPackage. **Target:** **0..*** SAFEPackage.subPackage	

Element "SafetyCaseExpression"

Parent Package: TopLevel

Stereotype: ,

Notes:

Provide information like justification or explanation on a specific element in safety case.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	elementType	SafeElementType							

Relationships

Name	Source/Target	Notes
	Source: SafetyCaseExpression. **Target:** SAFEElement.	
	Source: SafetyCaseExpression. **Target:** **0..*** SafetyCaseExpression.supportedBy	
	Source: SafetyCaseExpression. **Target:** **0..*** SafetyCaseExpression.inContextOf	

Element "Identifiable"

Parent Package: TopLevel

Stereotype: ,

Notes:

Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	category	Identifier			0	0	0		This element assigns a category to the parent element. The category is intended to specialize the usage and/or the content identifiable object. Such a specialization may also impose

									particular semantic constraints on the entire substructure (not only the identifiable itself).
	uuid	String			0	0	0		The purpose of this attribute is to provide a globally unique identifier for an instance of a metaclass. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and

									uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003".

Relationships

Name	Source/Target	Notes
	Source: MTEnumElement. **Target:** Identifiable.	
	Source: ErrorModelPrototype. **Target:** Identifiable.	
	Source: Identifiable. **Target:** Referrable.	
	Source: MalfunctionType. **Target:** Identifiable.	
	Source: HazardousEvent. **Target:** Identifiable.	
	Source: SAFEPackage. **Target:** Identifiable.	
	Source: MalfunctionPrototype. **Target:** Identifiable.	
	Source: Actor. **Target:** Identifiable.	
	Source: **1** Restrictable. **Target:** **1** Identifiable.VariableElement	
	Source: ErrorModel. **Target:** Identifiable.	
	Source: AbstractErrorBehavior. **Target:** Identifiable.	
	Source: PackageableElement. **Target:** Identifiable.	

Name	Source/Target	Notes
	Source: ControllabilityReference. **Target:** Identifiable.	
	Source: Hazard. **Target:** Identifiable.	

Package "Configuration"

Type of Package: **Package**

Parent Package: SAFE Meta-Model

Notes:

Diagram "Configuration"

Notes:

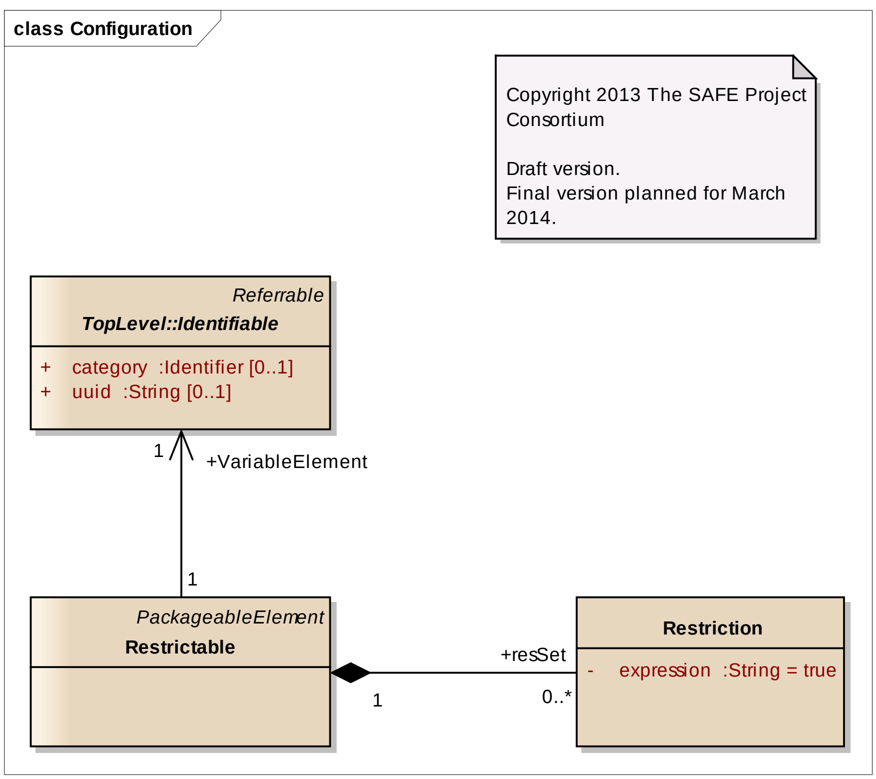


Figure: 12

Element "Restrictable"

Parent Package: Configuration

Stereotype: ,

Notes:

A RestrictableElement allows connecting SAFE model elements with variant information.

The identified element, pointed to by VariableElement of the class RestrictableElement, is part of a variant if at least one of the defined restrictions evaluate to true.

Relationships

Name	Source/Target	Notes
	Source: Restrictable. **Target:** PackageableElement.	
	Source: **1** Restrictable. **Target:** **1** Identifiable.VariableElement	
	Source: **0..*** Restriction.resSet **Target:** **1** Restrictable.	

Element "Restriction"

Parent Package: Configuration

Stereotype: ,

Notes:

A Restriction defines an expression that can be evaluated to true or false.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	expression	String			0	0	0	true	

Relationships

Name	Source/Target	Notes
	Source: **0..*** Restriction.resSet **Target:** **1** Restrictable.	

Package "ErrorModel"

Type of Package: **Package**

Parent Package: SAFE Meta-Model

Notes:

Diagram "ErrorModel"

Notes:

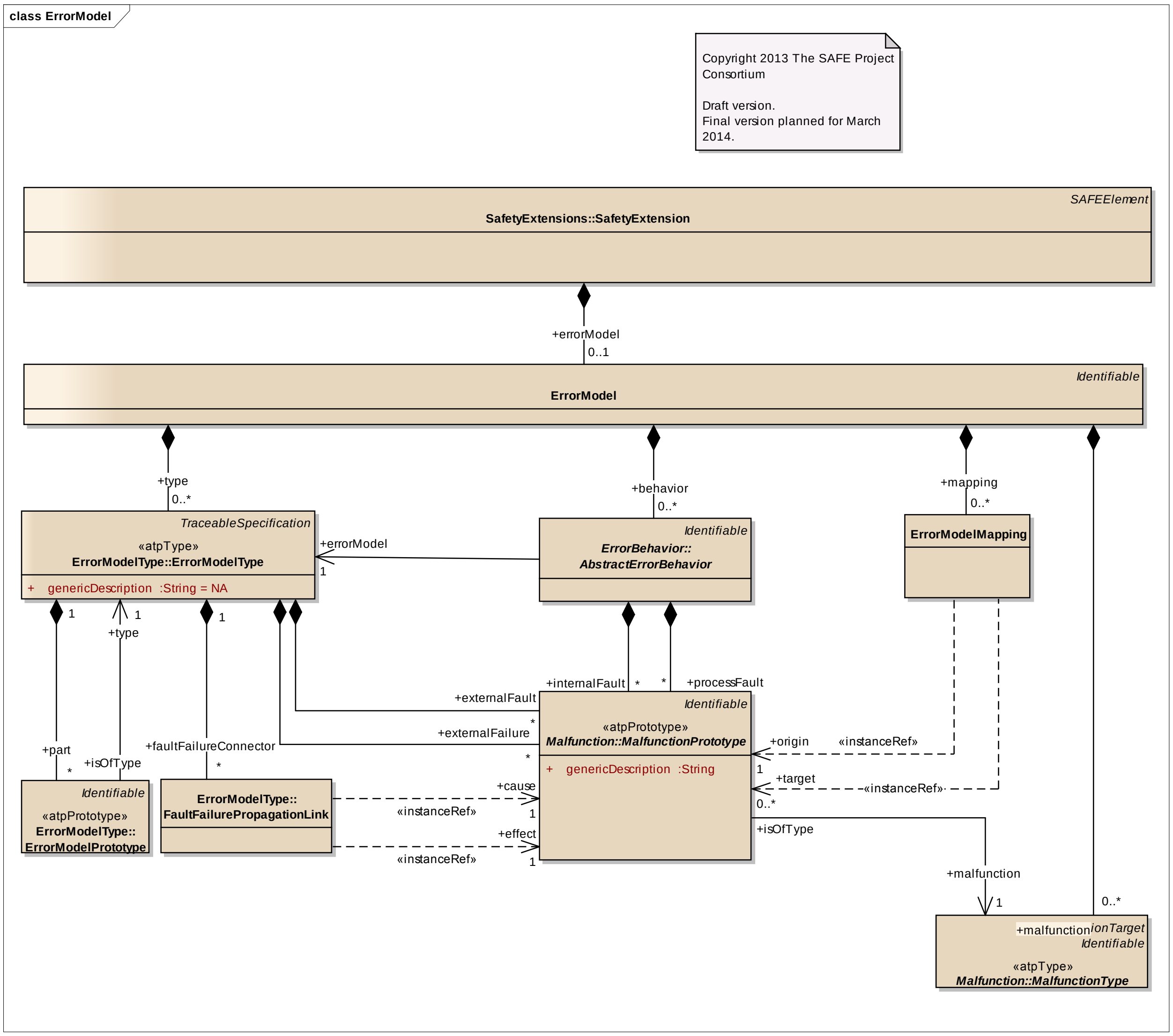


Figure: 13

Element "ErrorModel"

Parent Package: ErrorModel

Stereotype: ,

Notes:

The error model is a container for all artifacts, which are needed to describe the error model of an architectural element: malfunctions, error types and error behaviors

Relationships

Name	Source/Target	Notes
	Source: **0..*** ErrorModelMapping.mapping **Target:** ErrorModel.	
	Source: **0..*** AbstractErrorBehavior.behavior **Target:** ErrorModel.	an arbitrary number of error behaviors
	Source: ErrorModel. **Target:** Identifiable.	
	Source: **0..*** ErrorModelType.type **Target:** ErrorModel.	an arbitrary number of error model types
	Source: **0..*** MalfunctionType.malfunction **Target:** ErrorModel.	an arbitrary number of malfunction types
	Source: **0..1** ErrorModel.errorModel **Target:** SafetyExtension.	An error model associated with the respective safety extension. The error model is valid in the context of this safety extension.

Element "ErrorModelMapping"

Parent Package: ErrorModel

Stereotype: ,

Notes:

Via the class ErrorModelMapping it is possible to map malfunctions of one abstraction level (e.g. EAST ADL implementation level) to another level of abstraction (e.g. EAST ADL analysis level). This way the correlation between different levels of abstraction can be made explicit.

The ErrorModelMapping shall be attached to the "lower" level of abstraction. Example:

- a malfunction defined for a software component (implementation level) shall be related with a malfunction defined on design level
- in this case, (at least) two instances of SafetyExtensions (TechnicalSafetyExtension, AutosarSystemSafetyExtension) exist. In each of them an ErrorModelType is defined

containing the before mentioned malfunctions

- in the AutosarSystemSafetyExtension, we define an ErrorModelMapping and relate the software malfunction (in the role of "origin") to the malfunction defined within the TechnicalSafetyExtension (in the role "target")

Thus, the following rules shall be applied:

- the MalfunctionPrototype referenced via the "origin" relationship shall be defined within the same SafetyExtension as an instance of this class
- the MalfunctionPrototype referecnes via the "target" relationship shall be defined within a SafetyExtension which corresponds to a higher level of abstraction

Relationships

Name	Source/Target	Notes
	Source: ErrorModelMapping. **Target:** **0..*** MalfunctionPrototype.target	
	Source: **0..*** ErrorModelMapping.mapping **Target:** ErrorModel.	
	Source: **1** MalfunctionInstanceRef.origin **Target:** ErrorModelMapping.	
	Source: **1** MalfunctionInstanceRef.target **Target:** ErrorModelMapping.	
	Source: ErrorModelMapping. **Target:** **1** MalfunctionPrototype.origin	

Package "ErrorBehavior"

Type of Package: **Package**

Parent Package: ErrorModel

Notes:

Diagram "ErrorBehavior"

Notes:

Diagram for ErrorBehavior.

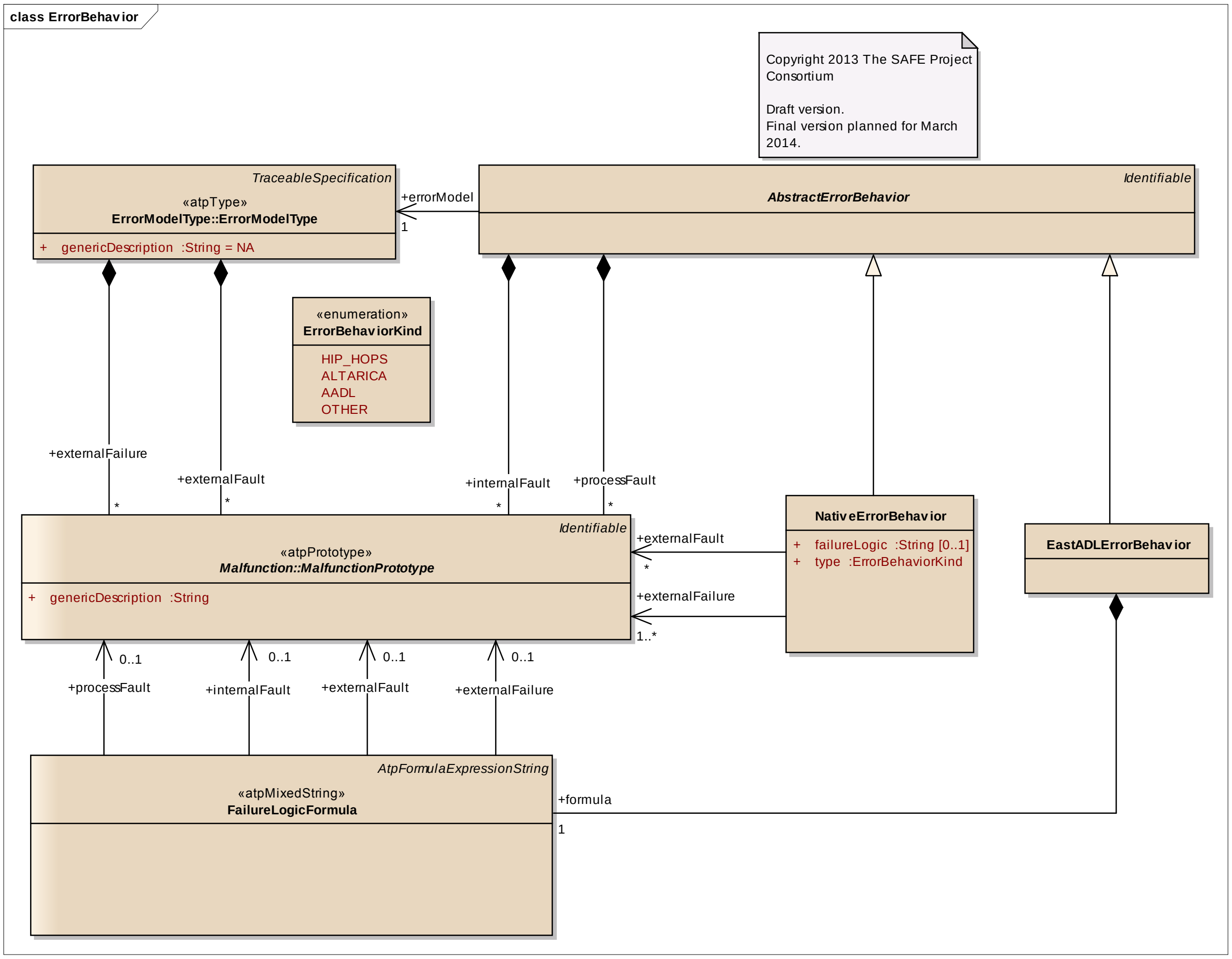


Figure: 14

Element "AbstractErrorBehavior"

Parent Package: ErrorBehavior

Stereotype: ,

Notes:

This class contains information about the error behavior independent of concrete behavior descriptions.

The AbstractErrorBehavior contains internalFaults, representing faults that are either propagated to externalFailures of the ErrorModelType or masked, according to the definition of its fault propagation.

A processFault represents a flaw introduced during design, and may lead to any of the failures represented by the ErrorModelType. A processFault therefore has a direct propagation to all externalFailures and cannot be masked.

Each error behavior description relates the occurrences of internal faults and incoming external faults to external failures. The faults and failures that the error behavior propagates to and from the target element are declared through the malfunction prototypes of the error model.

Semantics:

An error behavior describes the error propagation logic of its containing ErrorModelType.

The ErrorBehavior description represents the error propagation from internal faults or external faults to external failures. Faults are identified by the internalFault externalFault associations. The propagated external failures are identified by the externalFailure association.

Relationships

Name	Source/Target	Notes
	Source: EastADLErrorBehavior. **Target:** AbstractErrorBehavior.	
	Source: * MalfunctionPrototype.processFault **Target:** AbstractErrorBehavior.	processFaults that may affect the ErrorBehavior of the architectural element associated via the ErrorModelType
	Source: NativeErrorBehavior. **Target:** AbstractErrorBehavior.	
	Source: AbstractErrorBehavior. **Target:** 1 ErrorModelType.errorModel	
	Source: 0..* AbstractErrorBehavior.behavior **Target:** ErrorModel.	an arbitrary number of error behaviors
	Source: * MalfunctionPrototype.internalFault **Target:** AbstractErrorBehavior.	internalFaults that may affect the ErrorBehavior of the architectural element associated via the ErrorModelType
	Source: AbstractErrorBehavior. **Target:** Identifiable.	

Element "EastADLErrorBehavior"

Parent Package: ErrorBehavior

Stereotype: ,

Notes:

EASTADLErrorBehavior specifies a concrete failure logic description language, which describes the error propagation through the architectural element referenced by the containing ErrorModelType (e.g. function, hw component, sw component).

The failure logic is defined via a formula language called FailureLogicFormula (see "formula" association)

Relationships

Name	Source/Target	Notes
	Source: EastADLErrorBehavior. **Target:** AbstractErrorBehavior.	
	Source: 1 FailureLogicFormula.formula **Target:** EastADLErrorBehavior.	Failure logic used to describe the error propagation

Element "ErrorBehaviorKind"

Parent Package: ErrorBehavior

Stereotype: «enumeration»,

Notes:

The ErrorBehaviorKind metaclass represents an enumeration of literals describing various types of formalisms used for specifying error behavior.

Semantics:

ErrorBehaviorKind represents different formalisms for ErrorBehavior. The semantics is defined at each enumeration literal.

Extension:

Enumeration, no extension.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	HIP_HOPS				0	0	0		A specification of error behavior according to the external formalism HiP-HOPS.

	ALTARICA				0	0	0		A specification of error behavior according to the external formalism ALTARICA.
	AADL				0	0	0		A specification of error behavior according to the external formalism AADL.
	OTHER				0	0	0		A specification of error behavior according to other user defined formalism.

Element "FailureLogicFormula"

Parent Package: ErrorBehavior

Stereotype: «atpMixedString»,

Notes:

FailureLogicFormula is used to describe the error propagation through the architectural element associated with the containing ErrorModelType. The grammer of the FailureLogicFormula is defined in the respective specification document.

Relationships

Name	Source/Target	Notes
	Source: FailureLogicFormula. **Target:** **0..1** MalfunctionPrototype.externalFailure	external failures that may result from the ErrorBehavior
	Source: FailureLogicFormula. **Target:** **0..1** MalfunctionPrototype.processFault	processFaults that influence the errorBehavior
	Source: FailureLogicFormula. **Target:** **0..1** MalfunctionPrototype.internalFault	internalFaults that influence the errorBehavior
	Source: FailureLogicFormula.	

Name	Source/Target	Notes
	Target: AtpFormulaExpressionString.	
	Source: FailureLogicFormula. **Target:** **0..1** MalfunctionPrototype.externalFault	external(incoming) faults that influence the errorBehavior.
	Source: **1** FailureLogicFormula.formula **Target:** EastADLErrorBehavior.	Failure logic used to describe the error propagation

Element "NativeErrorBehavior"

Parent Package: ErrorBehavior

Stereotype: ,

Notes:

NativeErrorBehavior represents the descriptions of failure logics or semantics that the architectural element associated by the ErrorModelType exhibits.

Semantics:

The NativeErrorBehavior is defined in the failureLogic string, either directly or as a url referencing an external specification.

The failureLogic can be based on different formalisms, depending on the analysis techniques and tools available. This is indicated by its type:ErrorBehaviorKind attribute. The failureLogic attribute contains the actual failure propagation logic.

Extension:

UML:Behavior

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	failureLogic	String			0	0	0		The specification of error behavior based on an external formalism or the path to the file containing the external specification.
	type	ErrorBehaviorKind			0	0	0		The type of formalism applied for the error behavior

									description.

Relationships

Name	Source/Target	Notes
	Source: NativeErrorBehavior. **Target:** AbstractErrorBehavior.	
	Source: NativeErrorBehavior. **Target:** **1..*** MalfunctionPrototype.externalFailure	external failures that may result from the ErrorBehavior
	Source: NativeErrorBehavior. **Target:** ***** MalfunctionPrototype.externalFault	external(incoming) faults that the influence errorBehavior.

Package "ErrorModelType"

Type of Package: **Package**

Parent Package: ErrorModel

Notes:

Diagram "ErrorModelPrototype"

Notes:

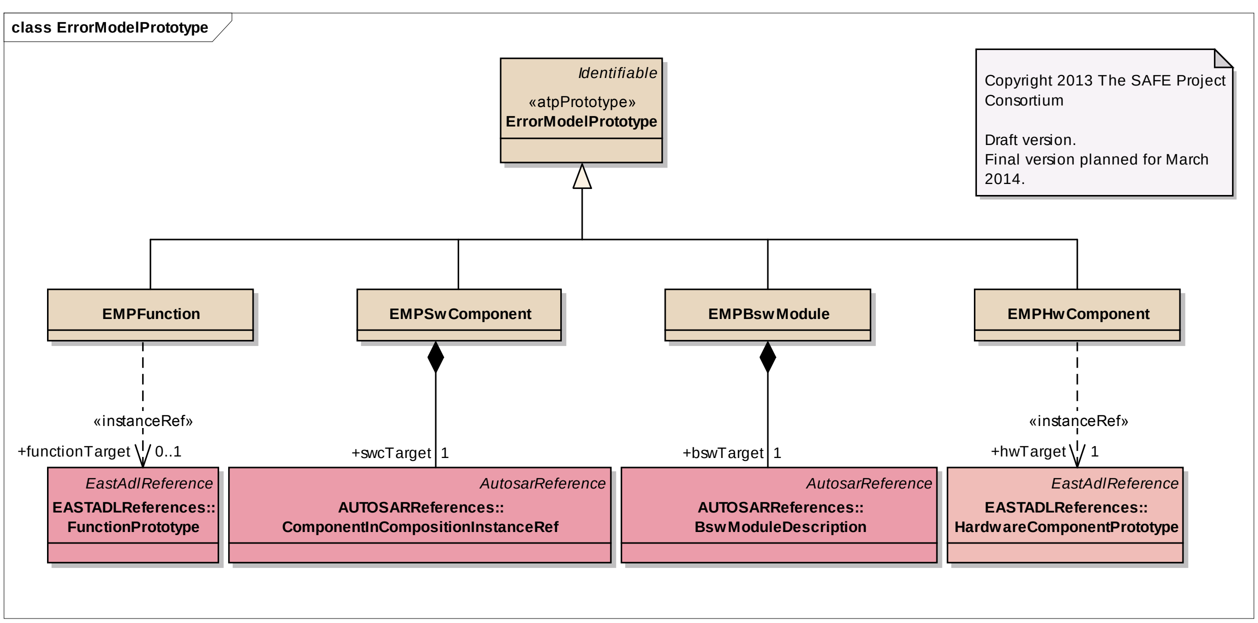


Figure: 15

Diagram "ErrorModelType"

Notes:

The EAST-ADL metaclasses for defining the error model structure.

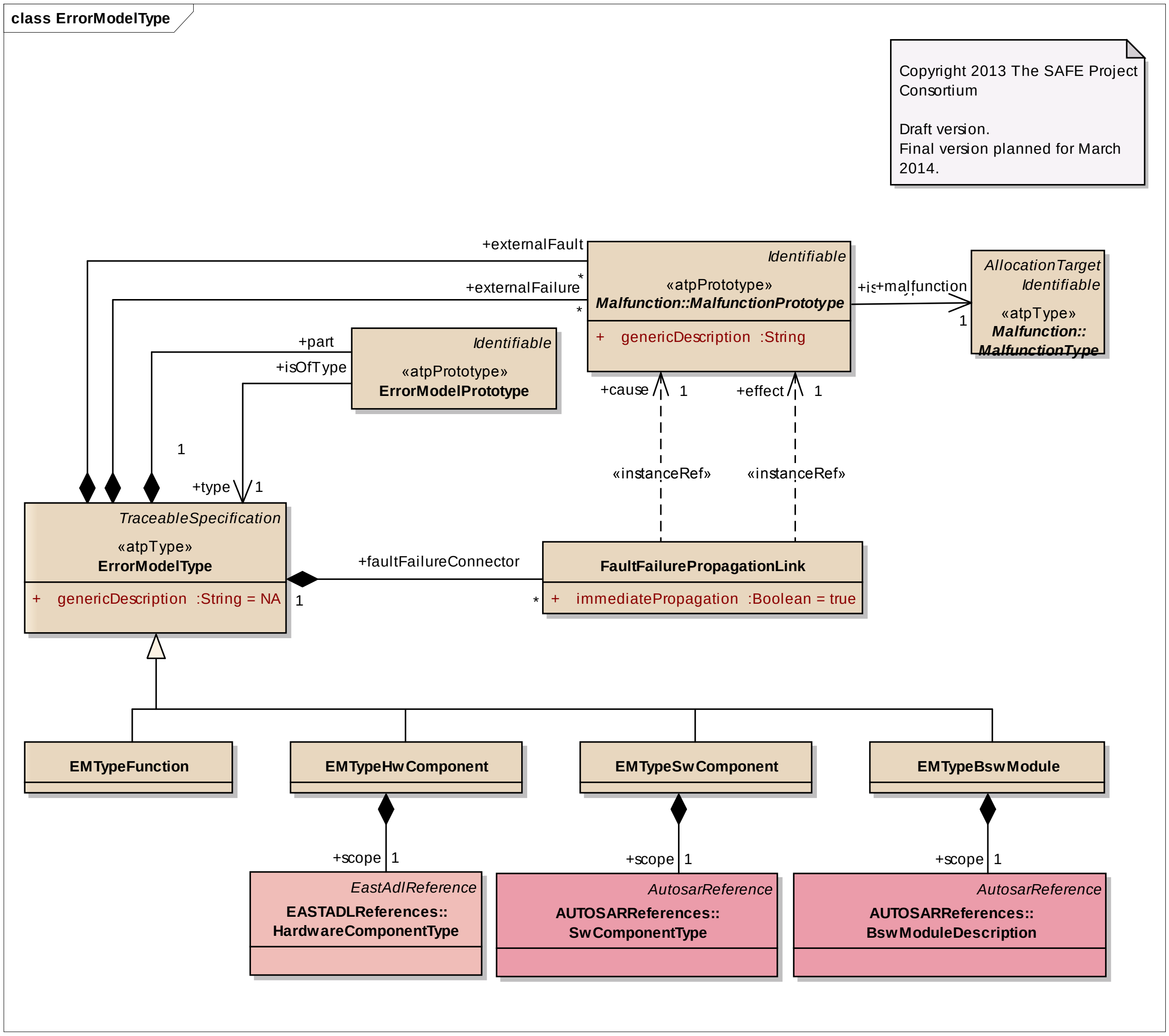


Figure: 16

Element "EMPBswModule"

Parent Package: ErrorModelType

Stereotype: ,

Notes:

Error model prototype specified for a concrete bsw software module.

Relationships

Name	Source/Target	Notes
	Source: **1** BswModuleDescription.bswTarget **Target:** EMPBswModule.	The target basic software module
	Source: EMPBswModule. **Target:** ErrorModelPrototype.	

Element "EMPFunction"

Parent Package: ErrorModelType

Stereotype: ,

Notes:

Error model prototype specified for a concrete function instance.

Relationships

Name	Source/Target	Notes
	Source: **0..1** ErrorModelPrototype_functionTarget.functionTarget **Target:** EMPFunction.	A nominal function instance as target of the related error model prototype.
	Source: EMPFunction. **Target:** **0..1** FunctionPrototype.functionTarget	the target function instance
	Source: EMPFunction. **Target:** ErrorModelPrototype.	

Element "EMPHwComponent"

Parent Package: ErrorModelType

Stereotype: ,

Notes:

Error model prototype specified for a concrete hardware component instance.

Relationships

Name	Source/Target	Notes
	Source: ***** ErrorModelPrototype_hwTarget.hwTarget	A nominal

Name	Source/Target	Notes
	Target: EMPHwComponent.	hardware component instance as target of the error model protoype.
	Source: EMPHwComponent. **Target:** **1** HardwareComponentPrototype.hwTarget	
	Source: EMPHwComponent. **Target:** ErrorModelPrototype.	

Element "EMPReference"

Parent Package: ErrorModelType

Stereotype: ,

Notes:

Element "EMPSwComponent"

Parent Package: ErrorModelType

Stereotype: ,

Notes:

Error model prototype specified for a concrete software component instance.

Relationships

Name	Source/Target	Notes
	Source: EMPSwComponent. **Target:** ErrorModelPrototype.	
	Source: **1** ComponentInCompositionInstanceRef.swcTarget **Target:** EMPSwComponent.	the target software component instance

Element "EMTypeBswModule"

Parent Package: ErrorModelType

Stereotype: ,

Notes:

Error model type specified for a concrete basic software module.

Relationships

Name	Source/Target	Notes
	Source: EMTypeBswModule. **Target:** ErrorModelType.	
	Source: 1 BswModuleDescription.scope **Target:** EMTypeBswModule.	the target basic software module

Element "EMTypeFunction"

Parent Package: ErrorModelType

Stereotype: ,

Notes:

Error model type specified for a concrete function.

Relationships

Name	Source/Target	Notes
	Source: EMTypeFunction. **Target:** ErrorModelType.	

Element "EMTypeHwComponent"

Parent Package: ErrorModelType

Stereotype: ,

Notes:

Error model type specified for a concrete hardware component

Relationships

Name	Source/Target	Notes
	Source: 1 HardwareComponentType.scope **Target:** EMTypeHwComponent.	the target hardware component
	Source: EMTypeHwComponent. **Target:** ErrorModelType.	

Element "EMTypeSwComponent"

Parent Package: ErrorModelType

Stereotype: ,

Notes:

Error model type specified for a concrete software component

Relationships

Name	Source/Target	Notes
	Source: 1 SwComponentType.scope **Target:** EMTypeSwComponent.	the target software component
	Source: EMTypeSwComponent. **Target:** ErrorModelType.	

Element "ErrorModelPrototype"

Parent Package: ErrorModelType

Stereotype: «atpPrototype»,

Notes:

The ErrorModelPrototype is used to define hierarchical error models allowing additional detail or structure to the error model of a particular target. A hierarchal structure can also be defined when several ErrorModels are integrated to a larger ErrorModel representing a system integrated from several targets.

There are diffent subtypes of ErrorModelPrototype specified, allowing to add additional information describe the context of the ErrorModelProtoype.

Semantics:

An ErrorModelPrototype represents an occurrence of the ErrorModelType that types it.

Extension:

(See ADLFunctionPrototype)

Relationships

Name	Source/Target	Notes
	Source: ErrorModelPrototype. **Target:** Identifiable.	
	Source: ErrorModelPrototype.isOfType	The ErrorModelType

Name	Source/Target	Notes
	Target: 1 ErrorModelType.type	that types the ErrorModelPrototype.
	Source: EMPSwComponent. **Target:** ErrorModelPrototype.	
	Source: EMPBswModule. **Target:** ErrorModelPrototype.	
	Source: EMPHwComponent. **Target:** ErrorModelPrototype.	
	Source: MalfunctionInstanceRef. **Target: 0..*** ErrorModelPrototype.errorModelPrototype	
	Source: 1 ErrorModelType. **Target: *** ErrorModelPrototype.part	The contained error models forming a hierarchy.
	Source: EMPFunction. **Target:** ErrorModelPrototype.	

Element "ErrorModelType"

Parent Package: ErrorModelType

Stereotype: «atpType»,

Notes:

ErrorModelType and ErrorModelPrototype support the hierarchical composition of error models based on the type-prototype pattern also adopted for the nominal architecture composition. The purpose of the error models is to represent information relating to the anomalies of a nominal model element.

Independent of the different subtypes of ErrorModelType, this class describes the external faults affecting the element, external failures caused by the element and fault propagations within the nominal element.

ErrorModelType inherits the abstract metaclass TraceableSpecification, allowing the ErrorModelType to be referenced from its design context in a similar way as requirements, test cases and other specifications.

Constraints:

For An ErrorModelType without part, a respective error behavior shall be defined in the safety model.

Semantics:

The ErrorModelType represents a specification of the faults and fault propagations of its target element.

Both types and prototypes may be targets, and the following cases are relevant:

- One nominal type:

The ErrorModelType represents the identified nominal type wherever this nominal type is instantiated.

- Several nominal types:

The ErrorModelType represents the identified nominal types individually, i.e. the same error model applies to all nominal types and is reused.

- One nominal prototype:

The ErrorModelType represents the identified nominal prototype whenever its context, i.e. its top-level composition is instantiated.

- Several nominal prototypes with instanceref:

The ErrorModelType represents the identified set of nominal prototypes (together) whenever their context, i.e. their top-level composition, is instantiated.

The fault propagation of an errorModelType is defined by its contained parts, the ErrorModelPrototypes and their connections. In case an error behavior is defined for this error model type, the fault propagation information, the error behavior and the parts of the error model shall be consistent.

FaultFailurePropagationLinks define valid propagation paths in the ErrorModelType. In case the contained external faults and external failures reference nominal ports, the connectivity of the nominal model may serve as a pattern for connecting malfunction prototypes in the ErrorModelType.

Extension:

(see ADLTraceableSpecfication)

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	genericDescription	String			0	0	0	NA	

Relationships

Name	Source/Target	Notes
	Source: **1** ErrorModelType. **Target:** ***** FaultFailurePropagationLink.faultFailureConnector	The contained links for internal propagation of faults/failures between the subordinate error models.
	Source: ErrorModelPrototype.isOfType **Target:** **1** ErrorModelType.type	The ErrorModelType that types the ErrorModelPrototype.
	Source: EMTypeBswModule. **Target:** ErrorModelType.	
	Source: EMTypeFunction. **Target:** ErrorModelType.	
	Source: AbstractErrorBehavior. **Target:** **1** ErrorModelType.errorModel	
	Source: EMTypeHwComponent. **Target:** ErrorModelType.	
	Source: ***** MalfunctionPrototype.externalFault **Target:** ErrorModelType.	The external faults affecting the proper execution of the architectural element associated with the error model type
	Source: ErrorModelType. **Target:** TraceableSpecification.	
	Source: ***** MalfunctionPrototype.externalFailure **Target:** ErrorModelType.	The external failures visible at the borders of the architectural

Name	Source/Target	Notes
		element.
	Source: EMTypeSwComponent. **Target:** ErrorModelType.	
	Source: **0..*** ErrorModelType.type **Target:** ErrorModel.	an arbitrary number of error model types
	Source: **1** ErrorModelType. **Target:** ***** ErrorModelPrototype.part	The contained error models forming a hierarchy.

Element "FaultFailurePropagationLink"

Parent Package: ErrorModelType

Stereotype: ,

Notes:

The FaultFailurePropagationLink metaclass represents the links for the propagations of faults/failures across system elements. In particular, it defines that one error model provides the faults/failures that another error model receives.

A fault/failure link can only be applied to compatible ports, either for fault/failure delegation within an error model or for fault/failure transmission across two error models. A FaultFailurePropagationLink can only connect fault/failures that have compatible types.

Constraints:

[1] Only compatible cause-effect pairs may be connected.

[2] Two fault/failure are compatible if the MalfunctionType of the cause represents a subset of the MalfunctionType set represented by the MalfunctionType of the effect.

Semantics:

The FaultFailurePropagationLink defines a Failure propagation path, from the cause on one error model to the effect of another error model.

Extension:

UML::Connector

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	immediatePropagation	Boolean			0	0	0	true	

Relationships

Name	Source/Target	Notes
	Source: **1** ErrorModelType. **Target:** ***** FaultFailurePropagationLink.faultFailureConnector	The contained links for internal propagation of faults/failures between the subordinate error models.
	Source: **1** MalfunctionInstanceRef.effect **Target:** FaultFailurePropagationLink.	
	Source: FaultFailurePropagationLink. **Target:** **1** MalfunctionInstanceRef.cause	
	Source: FaultFailurePropagationLink. **Target:** **1** MalfunctionPrototype.effect	
	Source: FaultFailurePropagationLink. **Target:** **1** MalfunctionPrototype.cause	

Package "Malfunction"

Type of Package: **Package**

Parent Package: ErrorModel

Notes:

Diagram "MalfunctionPrototype"

Notes:

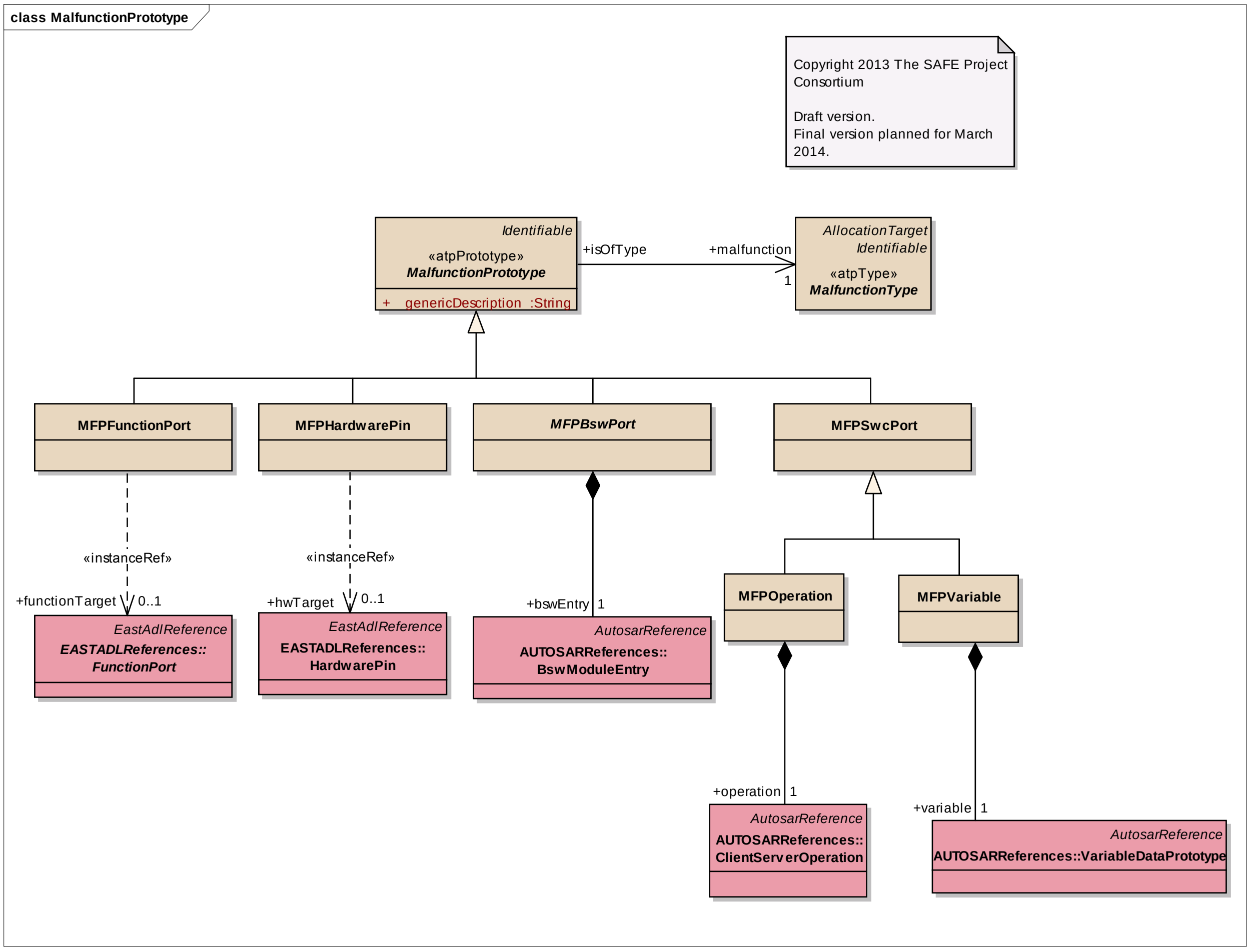


Figure: 17

Diagram "MalfunctionType"

Notes:

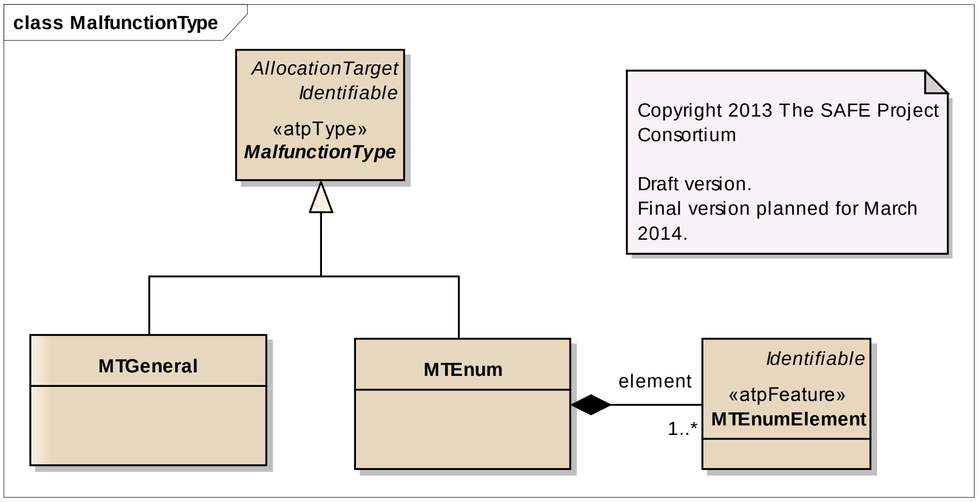


Figure: 18

Element "MFPBswPort"

Parent Package: Malfunction

Stereotype: ,

Notes:

The MalfunctionPrototype pointing to a basic software module entry

Relationships

Name	Source/Target	Notes
	Source: MFPBswPort. **Target:** MalfunctionPrototype.	
	Source: 1 BswModuleEntry.bswEntry **Target:** MFPBswPort.	the target bsw module entry

Element "MFPFunctionPort"

Parent Package: Malfunction

Stereotype: ,

Notes:

The MalfunctionPrototype pointing to a function port instance

Relationships

Name	Source/Target	Notes

Name	Source/Target	Notes
	Source: MFPFunctionPort. **Target:** **0..1** FunctionPort.functionTarget	
	Source: MFPFunctionPort. **Target:** MalfunctionPrototype.	
	Source: **0..1** FaultFailurePort_functionTarget.functionTarget **Target:** MFPFunctionPort.	A nominal function port instance as target of the malfunction prototype.

Element "MFPHardwarePin"

Parent Package: Malfunction

Stereotype: ,

Notes:

The MalfunctionPrototype pointing to a HardwarPin instance

Relationships

Name	Source/Target	Notes
	Source: ***** FaultFailurePort_hwTarget.hwTarget **Target:** MFPHardwarePin.	A nominal HW pin instance as target of the malfunction prototype.
	Source: MFPHardwarePin. **Target:** MalfunctionPrototype.	
	Source: MFPHardwarePin. **Target:** **0..1** HardwarePin.hwTarget	

Element "MFPOperation"

Parent Package: Malfunction

Stereotype: ,

Notes:

The MalfunctionPrototype pointing to an AUTOSAR operation instance

Relationships

Name	Source/Target	Notes
	Source: MFPOperation. **Target:** MFPSwcPort.	
	Source: **1** ClientServerOperation.operation **Target:** MFPOperation.	the target operation prototype instance

Element "MFPSwcPort"

Parent Package: Malfunction

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: MFPSwcPort. **Target:** MalfunctionPrototype.	
	Source: MFPOperation. **Target:** MFPSwcPort.	
	Source: MFPVariable. **Target:** MFPSwcPort.	

Element "MFPVariable"

Parent Package: Malfunction

Stereotype: ,

Notes:

The MalfunctionPrototype pointing to an AUTOSAR variable instance

Relationships

Name	Source/Target	Notes
	Source: **1** VariableDataPrototype.variable **Target:** MFPVariable.	the target variable data prototype instance
	Source: MFPVariable.	

Name	Source/Target	Notes
	Target: MFPSwcPort.	

Element "MTEnum"

Parent Package: Malfunction

Stereotype: ,

Notes:

This enumeration malfunction type allows to define the different ways, how the malfunction becomes visible. As a typical example, an enumeration could have the enumerations "comission" and "omission".

BrakeMalfunctionType:

- BrakePressureTooLow

Semantics="brake pressure is below 20% of requested value"

- Omission

Semantics="brake pressure is below 10% of maximal brake pressure"

- Comission

Semantics="brake pressure exceeds requested value with more than 10% of maximal brake pressure"

Semantics may also be a more formal expression defining in the type of the nominal datatype what value range is considered a fault. This depends on the user and tooling available.

Relationships

Name	Source/Target	Notes
	Source: MTEnum. **Target:** MalfunctionType.	
element	**Source:** **1..*** MTEnumElement. **Target:** MTEnum.	elements of the malfunction type enum

Element "MTEnumElement"

Parent Package: Malfunction

Stereotype: «atpFeature»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: MTEnumElement. **Target:** Identifiable.	
element	**Source:** **1..*** MTEnumElement. **Target:** MTEnum.	elements of the malfunction type enum

Element "MTGeneral"

Parent Package: Malfunction

Stereotype: ,

Notes:

General description of a malfunction. The description field of the derived Identifiable class shall be used to describe the malfunction.

Relationships

Name	Source/Target	Notes
	Source: MTGeneral. **Target:** MalfunctionType.	

Element "MalfunctionPrototype"

Parent Package: Malfunction

Stereotype: «atpPrototype»,

Notes:

A malfunction is a failure or unintended behavior of the item or element of the item that has the potential to propogate. The MalfunctionPrototype metaclass represents an error that may occur internally in an ErrorModel or be propagated to it, or a failure that is propagated out of an Error Model. The MalfunctionPrototype may represent different errors depending on its type (enumeration of generic description).

Semantics:

An malfunction prototype refers to a condition that deviates from expectations based on requirements specifications, design documents, user documents, standards, etc., or from someone's perceptions or experiences (ISO26262). The set of available faults or failures represented by the MalfunctionPrototype is defined by its type, typically an enumeration type like {omission, commission}. It is an abstract class further specialized with metaclasses for different types of fault/failure.

Extension:

(UML::Part)

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	genericDescription	String			0	0	0		A description of the MalfunctionPrototype

Relationships

Name	Source/Target	Notes
	Source: Hazard. **Target:** **1** MalfunctionPrototype.malfunction	
	Source: FailureLogicFormula. **Target:** **0..1** MalfunctionPrototype.externalFailure	external failures that may result from the ErrorBehavior
	Source: ***** MalfunctionPrototype.processFault **Target:** AbstractErrorBehavior.	processFaults that may affect the ErrorBehavior of the architectural element associated via the ErrorModelType
	Source: ErrorModelMapping. **Target:** **0..*** MalfunctionPrototype.target	
	Source: FailureLogicFormula. **Target:** **0..1** MalfunctionPrototype.processFault	processFaults that influence the errorBehavior
	Source: MFPFunctionPort. **Target:** MalfunctionPrototype.	
	Source: **0..1** HardwareFailureAnalysis. **Target:** **1** MalfunctionPrototype.malfunctionAnalysis	

Name	Source/Target	Notes
	Source: MFPSwcPort. **Target:** MalfunctionPrototype.	
	Source: MFPHardwarePin. **Target:** MalfunctionPrototype.	
	Source: MalfunctionPrototype. **Target:** Identifiable.	
	Source: **0..1** HWPartFailureAnalysis. **Target:** **1** MalfunctionPrototype.malfunctionPartAnalysis	
	Source: * MalfunctionPrototype.externalFault **Target:** ErrorModelType.	The external faults affecting the proper execution of the architectural element associated with the error model type
	Source: MFPBswPort. **Target:** MalfunctionPrototype.	
	Source: * MalfunctionPrototype.externalFailure **Target:** ErrorModelType.	The external failures visible at the borders of the architectural element.
	Source: FailureLogicFormula. **Target:** **0..1** MalfunctionPrototype.internalFault	internalFaults that influence the errorBehavior
	Source: MalfunctionInstanceRef. **Target:** **1** MalfunctionPrototype.malfunction	
	Source: * MalfunctionPrototype.internalFault **Target:** AbstractErrorBehavior.	internalFaults that may affect the ErrorBehavior of the architectural element associated via the ErrorModelType
	Source: NativeErrorBehavior. **Target:** **1..*** MalfunctionPrototype.externalFailure	external failures that may result from the ErrorBehavior

Name	Source/Target	Notes
	Source: MalfunctionPrototype.isOfType **Target:** **1** MalfunctionType.malfunction	The type of the malfunction prototype. It describes how the malfunction prototype becomes visible.
	Source: NativeErrorBehavior. **Target:** ***** MalfunctionPrototype.externalFault	external(incoming) faults that influence the errorBehavior.
	Source: FailureLogicFormula. **Target:** **0..1** MalfunctionPrototype.externalFault	external(incoming) faults that influence the errorBehavior.
	Source: FaultFailurePropagationLink. **Target:** **1** MalfunctionPrototype.effect	
	Source: FaultFailurePropagationLink. **Target:** **1** MalfunctionPrototype.cause	
	Source: ErrorModelMapping. **Target:** **1** MalfunctionPrototype.origin	

Element "MalfunctionType"

Parent Package: Malfunction

Stereotype: «atpType»,

Notes:

A MalfunctionType describes how a malfunction becomes visible. Currently, it can either be a generic description of a malfunction or an enumeration of different "appearance" possibilities.

Relationships

Name	Source/Target	Notes
	Source: MalfunctionType. **Target:** Identifiable.	
	Source: MalfunctionType. **Target:** AllocationTarget.	
	Source: MTEnum. **Target:** MalfunctionType.	

Name	Source/Target			Notes
	Source:		MTGeneral.	
	Target:		MalfunctionType.	
	Source:		MalfunctionPrototype.isOfType	The type of the malfunction prototype. It describes how the malfunction prototype becomes visible.
	Target:	**1**	MalfunctionType.malfunction	
	Source:		HWFailureMode.	
	Target:		MalfunctionType.	
	Source:	**0..***	MalfunctionType.malfunction	an arbitrary number of malfunction types
	Target:		ErrorModel.	

Package "_instanceRef"

Type of Package: **Package**

Parent Package: ErrorModel

Notes:

Diagram "EMPFunction_functionTarget"

Notes:

Diagram for ErrorModelPrototype.

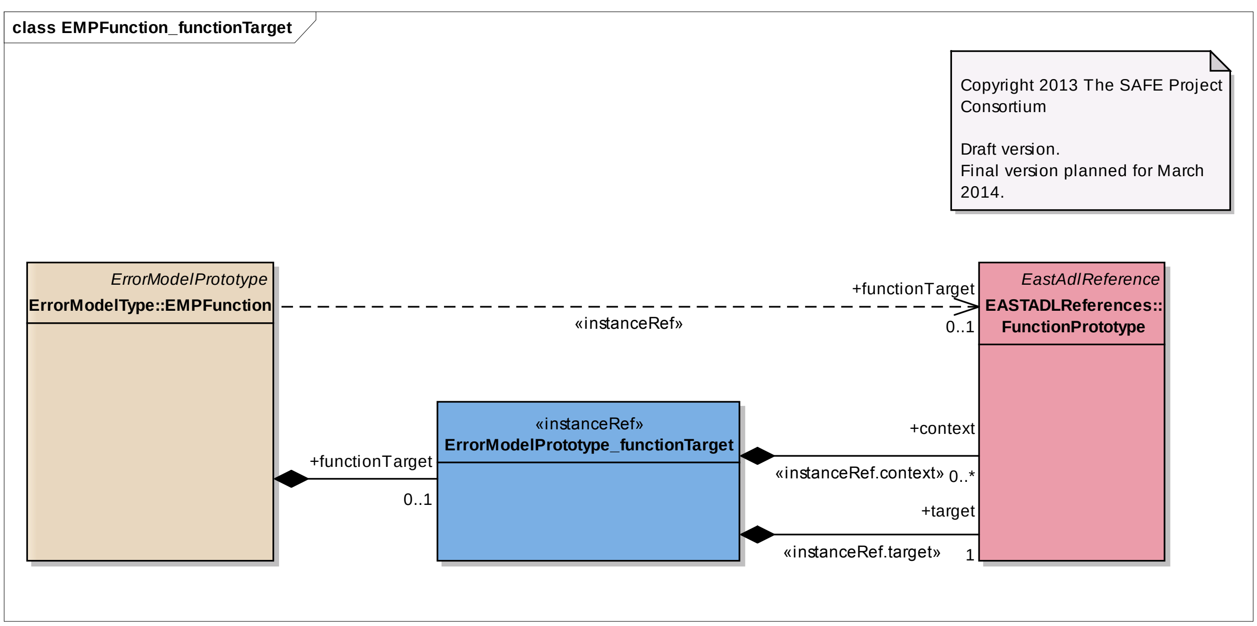


Figure: 19

Diagram "EMPHwComponent_hwTarget"

Notes:

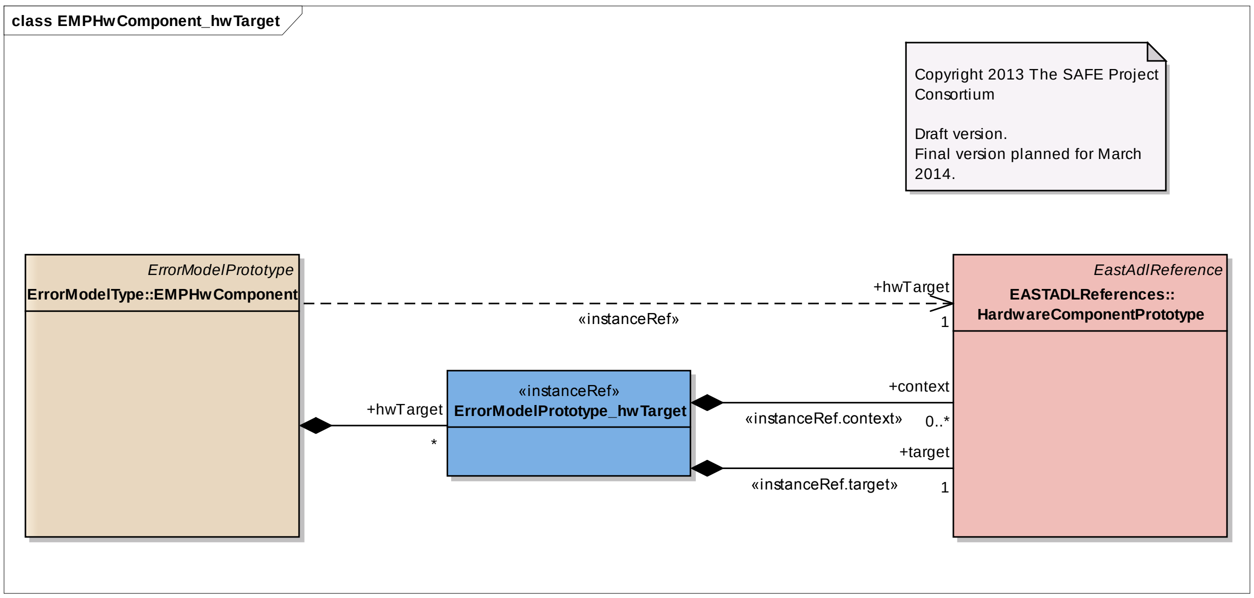


Figure: 20

Diagram "ErrorModelMapping"

Notes:

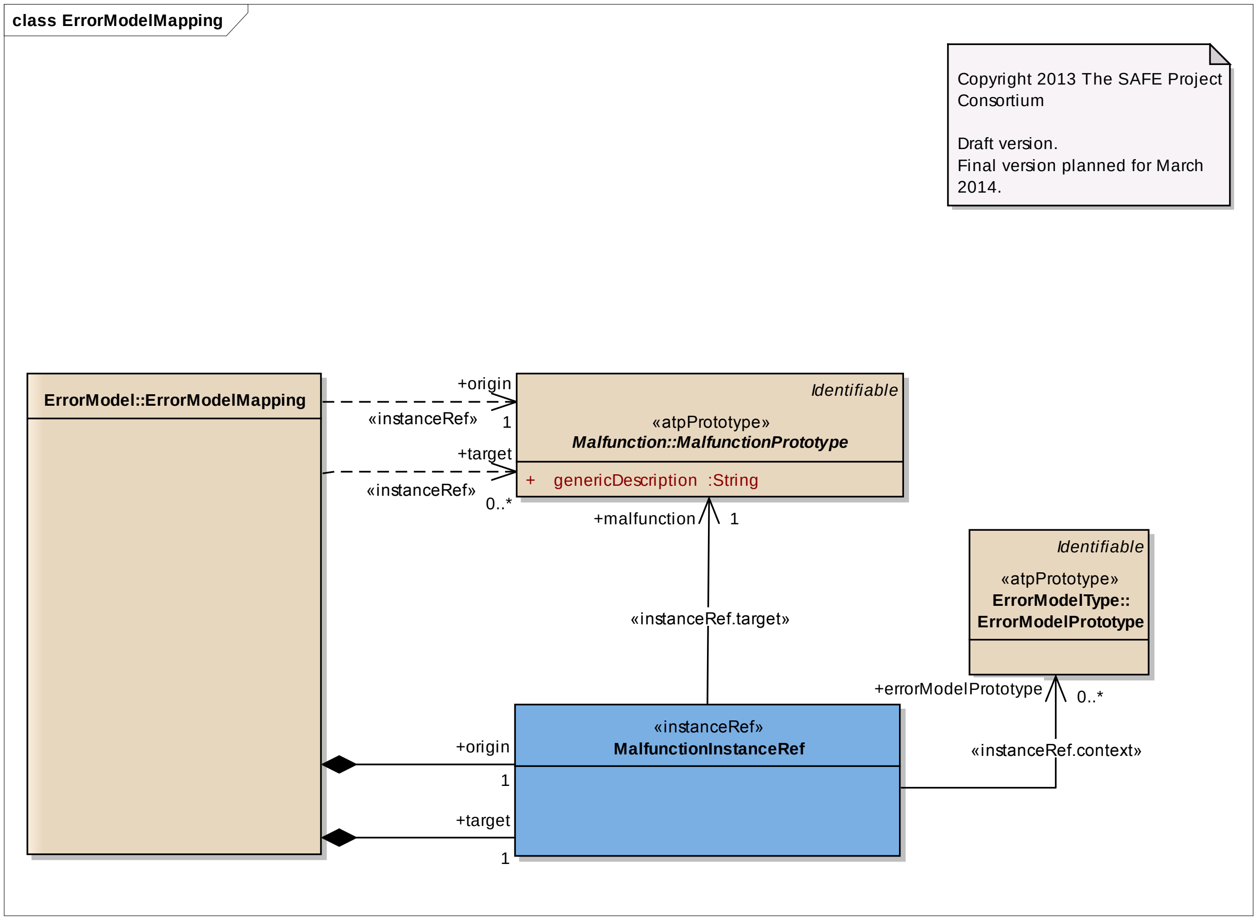


Figure: 21

Diagram "FaultFailurePropagationLink"

Notes:

Diagram for FaultFailurePropagationLink.

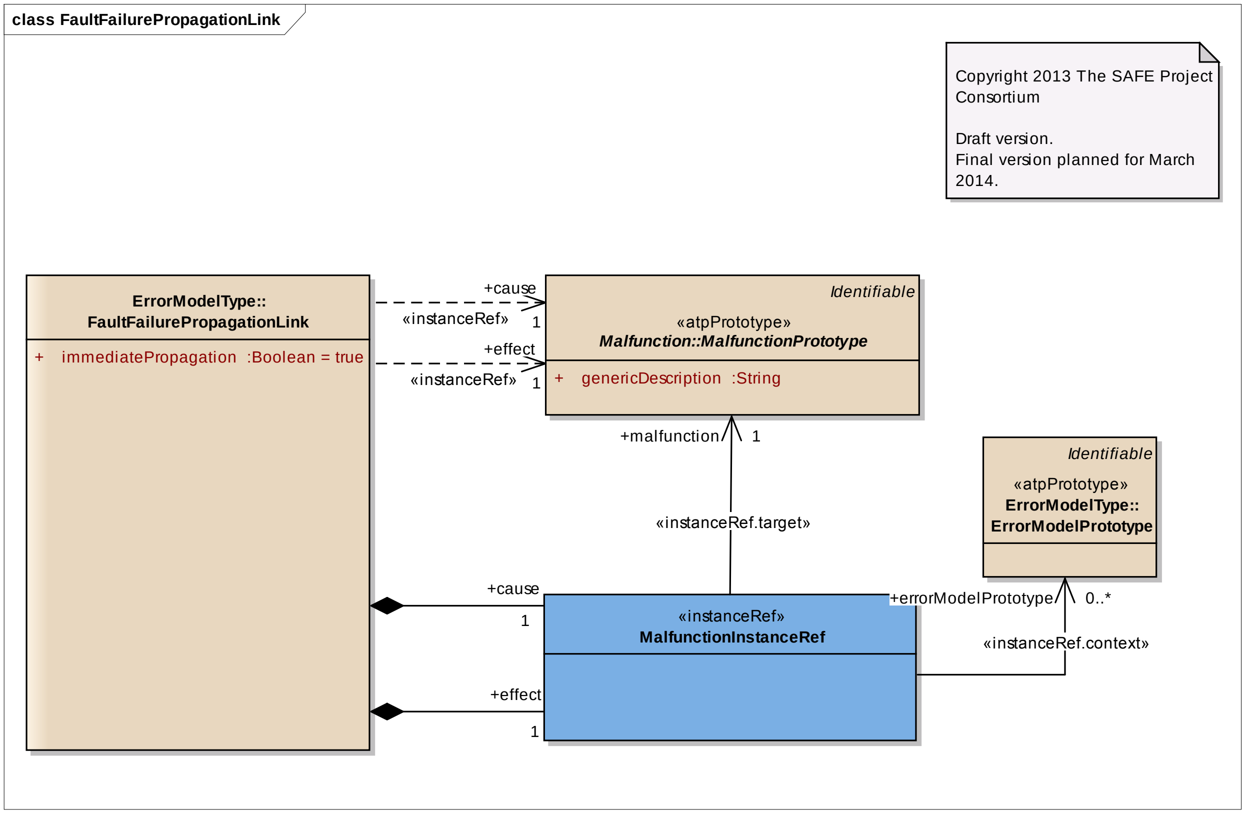


Figure: 22

Diagram "MFPFunctionPort_functionTarget"

Notes:

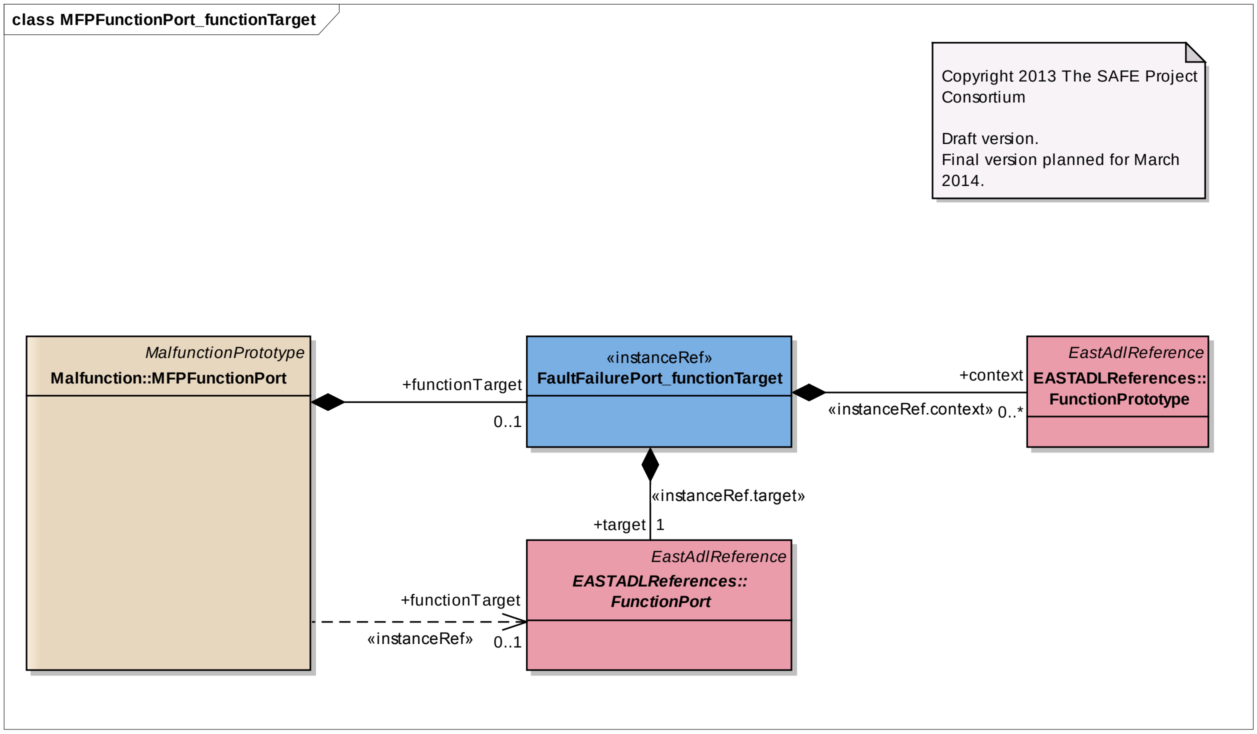


Figure: 23

Diagram "MFPHardwarePin_hwTarget"

Notes:

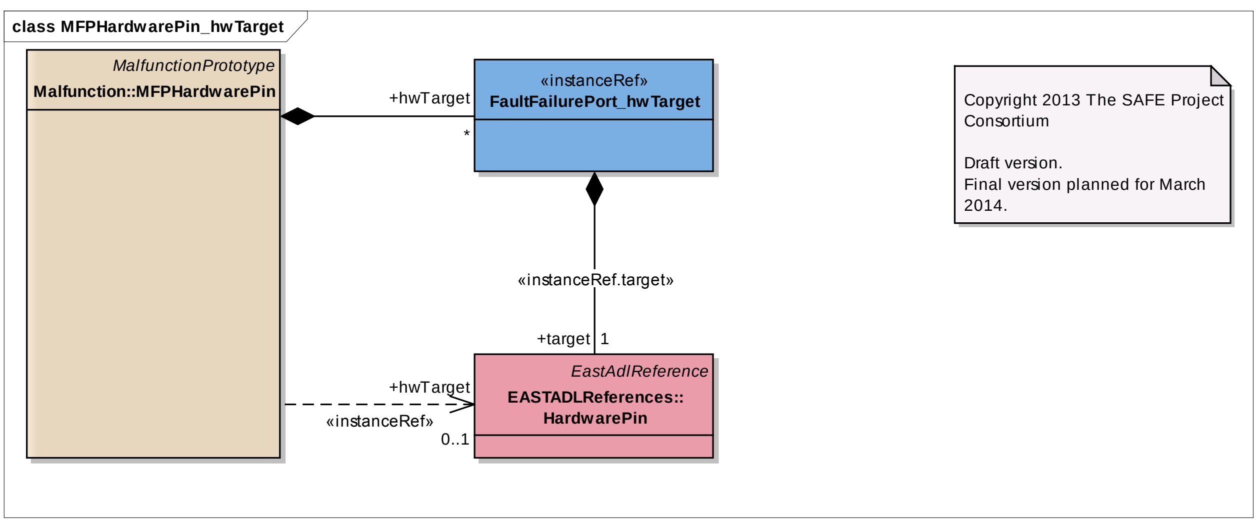


Figure: 24

Element "ErrorModelPrototype_functionTarget"

Parent Package: _instanceRef

Stereotype: «instanceRef»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: **0..1** ErrorModelPrototype_functionTarget.functionTarget **Target:** EMPFunction.	A nominal function instance as target of the related error model prototype.
	Source: **1** FunctionPrototype.target **Target:** ErrorModelPrototype_functionTarget.	
	Source: **0..*** FunctionPrototype.context **Target:** ErrorModelPrototype_functionTarget.	

Element "ErrorModelPrototype_hwTarget"

Parent Package: _instanceRef

Stereotype: «instanceRef»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: ***** ErrorModelPrototype_hwTarget.hwTarget **Target:** EMPHwComponent.	A nominal hardware component instance as target of the error model protoype.
	Source: **1** HardwareComponentPrototype.target **Target:** ErrorModelPrototype_hwTarget.	
	Source: **0..*** HardwareComponentPrototype.context **Target:** ErrorModelPrototype_hwTarget.	

Element "FaultFailurePort_functionTarget"

Parent Package: _instanceRef

Stereotype: «instanceRef»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: **0..*** FunctionPrototype.context **Target:** FaultFailurePort_functionTarget.	
	Source: **0..1** FaultFailurePort_functionTarget.functionTarget **Target:** MFPFunctionPort.	A nominal function port instance as target of the malfunction prototype.
	Source: **1** FunctionPort.target **Target:** FaultFailurePort_functionTarget.	

Element "FaultFailurePort_hwTarget"

Parent Package: _instanceRef

Stereotype: «instanceRef»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: **1** HardwarePin.target **Target:** FaultFailurePort_hwTarget.	
	Source: ***** FaultFailurePort_hwTarget.hwTarget **Target:** MFPHardwarePin.	A nominal HW pin instance as target of the malfunction prototype.

Element "MalfunctionInstanceRef"

Parent Package: _instanceRef

Stereotype: «instanceRef»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: **1** MalfunctionInstanceRef.effect **Target:** FaultFailurePropagationLink.	
	Source: FaultFailurePropagationLink. **Target:** **1** MalfunctionInstanceRef.cause	
	Source: **1** MalfunctionInstanceRef.origin **Target:** ErrorModelMapping.	
	Source: MalfunctionInstanceRef. **Target:** **1** MalfunctionPrototype.malfunction	
	Source: **1** MalfunctionInstanceRef.target **Target:** ErrorModelMapping.	
	Source: MalfunctionInstanceRef. **Target:** **0..*** ErrorModelPrototype.errorModelPrototype	

Package "Hardware"

Type of Package: **Package**

Parent Package: SAFE Meta-Model

Notes:

This package describes the top-level package for all meta model extension regarding hardware, developed in WT 3.2.2.

Package "FailureFormula"

Type of Package: **Package**

Parent Package: Hardware

Notes:

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

Diagram "FailureFormula"

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.

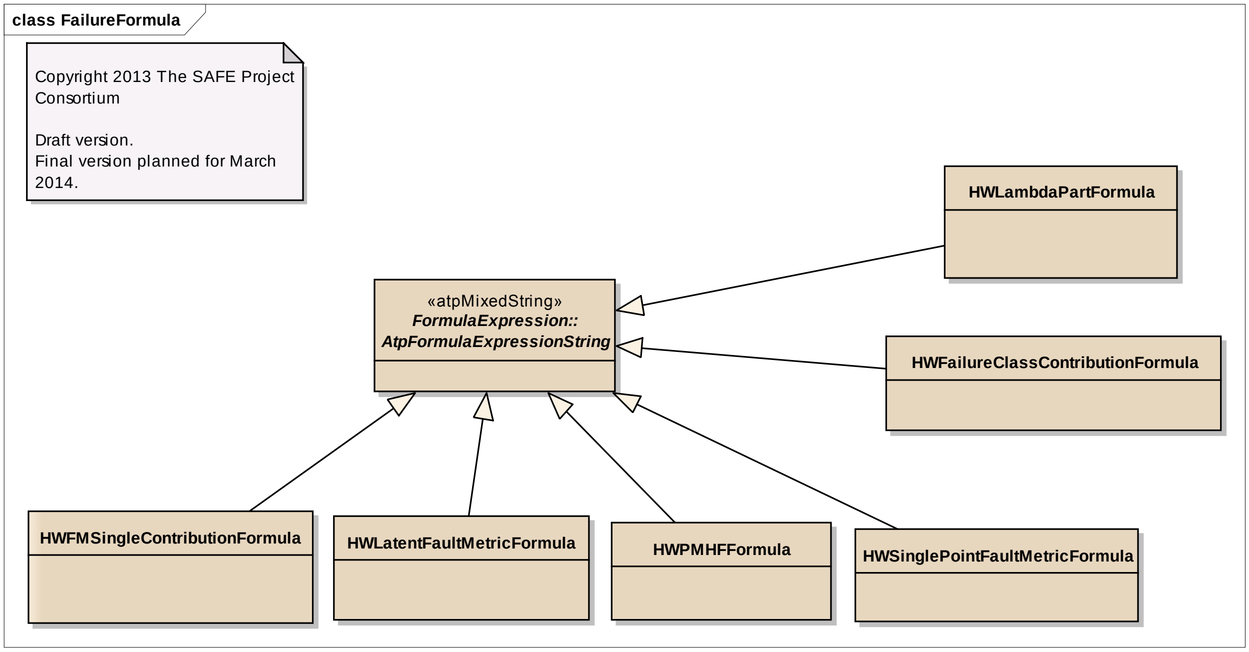


Figure: 25

Element "HWFMSingleContributionFormula"

Parent Package: FailureFormula

Stereotype: ,

Notes:

This class describes the individual contribution of an HWFailureMode of a 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).

SinglePointFault represents a first order fault, while DualPointFault

lambdaSafetyComponent = Value(HWFailureRate)

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

If (HWFault == SafeFault)

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

Else

lambdaSafeFault(HWFMSingleContribution) = 0

Endif

If (HWFault == SinglePointFault)

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

Else lambdaSinglePointFault(HWFMSingleContribution) = 0

Endif

If (HWFault == DualPointFault)

If (HWSafetyMechanism covers the FailureMode) //residual Fault as HWFailureMode.HWSafetyMechanism = null

lambdaResidualFault(HWFMSingleContribution) = [Value(HWFailureRate)*failureRateDistribution(HWFailureMode)] * [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)] * Other Component Failure Rate

Endif

Notes that value(HWFailureRate) and failureRateDistribution(HWFailureMode) are applied on the calculated value extracted from electronic design level to perform the final calculation and verification of the architectural hardware metrics and probabilistic evaluation of violation of the safety goal. The selection between allocated and calculated value is a tool feature that allow first a calculation for estimation based on allocation field of failure rate and distribution, and then verification based on HWComponentQuantifiedFMFromPartHWPart as extract from Part Analysis gives directly the lambda of the HWFailiure Mode as HWFailureRate*FailureRateDistribution thanks to the relation to HWFault.

Relationships

Name	Source/Target	Notes
	Source: HWFMSingleContributionFormula. **Target:** **1** HWFailureMode.failureRateDistribution	
	Source: **1** HWFMSingleContributionFormula.hwFMSCLambdaValue **Target:** HWFMSingleContribution.	
	Source: HWFMSingleContributionFormula. **Target:** **1** HWFault.hwFaultTypeValue	
	Source: HWFMSingleContributionFormula. **Target:** **1** HWFailureRate.hwFailureRateValue	
	Source: HWFMSingleContributionFormula. **Target:** AtpFormulaExpressionString.	

Element "HWFailureClassContributionFormula"

Parent Package: FailureFormula

Stereotype: ,

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).

Relationships

Name	Source/Target	Notes
	Source: HWFailureClassContributionFormula. **Target:** AtpFormulaExpressionString.	
	Source: **1** HWFailureClassContributionFormula.hwEFRChwElementValue **Target:** HWElementFailureRateClass.	
	Source: HWFailureClassContributionFormula. **Target:** * HWFMSingleContribution.lambdaValue	

Element "HWLambdaPartFormula"

Parent Package: FailureFormula

Stereotype: ,

Notes:

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

Relationships

Name	Source/Target	Notes
	Source: **1** HWLambdaPartFormula. **Target:** ***** HWPartFailureRate.hwPartFailureRateValue	
	Source: HWLambdaPartFormula. **Target:** AtpFormulaExpressionString.	
	Source: **1** HWLambdaPartFormula.hwCQFMlambdaFailureModeValue **Target:** HWComponentQuantifiedFMFromPart.	
	Source: **1** HWLambdaPartFormula. **Target:** ***** HWPartFailureMode.failureRateDistributionValue	

Element "HWLatentFaultMetricFormula"

Parent Package: FailureFormula

Stereotype: ,

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).

Relationships

Name	Source/Target	Notes
	Source: HWLatentFaultMetricFormula. **Target:** * HWFMSingleContribution.lambdaValue	
	Source: HWLatentFaultMetricFormula. **Target:** AtpFormulaExpressionString.	
	Source: 1 HWLatentFaultMetricFormula.lfmCalculatedValue **Target:** HWLatentFaultMetric.	

Element "HWPMHFFormula"

Parent Package: FailureFormula

Stereotype: ,

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 Component safety relevant are considered.

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

Relationships

Name	Source/Target	Notes
	Source: HWPMHFFormula. **Target:** * HWFMSingleContribution.lambdaValue	

Name	Source/Target	Notes
	Source: HWPMHFFormula. **Target:** AtpFormulaExpressionString.	
	Source: **1** HWPMHFFormula.hwPMHFCalculatedValue **Target:** HWPMHF.	

Element "HWSinglePointFaultMetricFormula"

Parent Package: FailureFormula

Stereotype: ,

Notes:

This class describes the 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).

Relationships

Name	Source/Target	Notes
	Source: **1** HWSinglePointFaultMetricFormula.spfmCalculatedValue **Target:** HWSinglePointFaultMetric.	
	Source: HWSinglePointFaultMetricFormula. **Target:** * HWFMSingleContribution.lambdaValue	

Name	Source/Target		Notes
	Source:	HWSinglePointFaultMetricFormula.	
	Target:	AtpFormulaExpressionString.	

Package "Failure"

Type of Package: **Package**

Parent Package: Hardware

Notes:

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

Diagram "FailureAnalysis"

Notes:

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

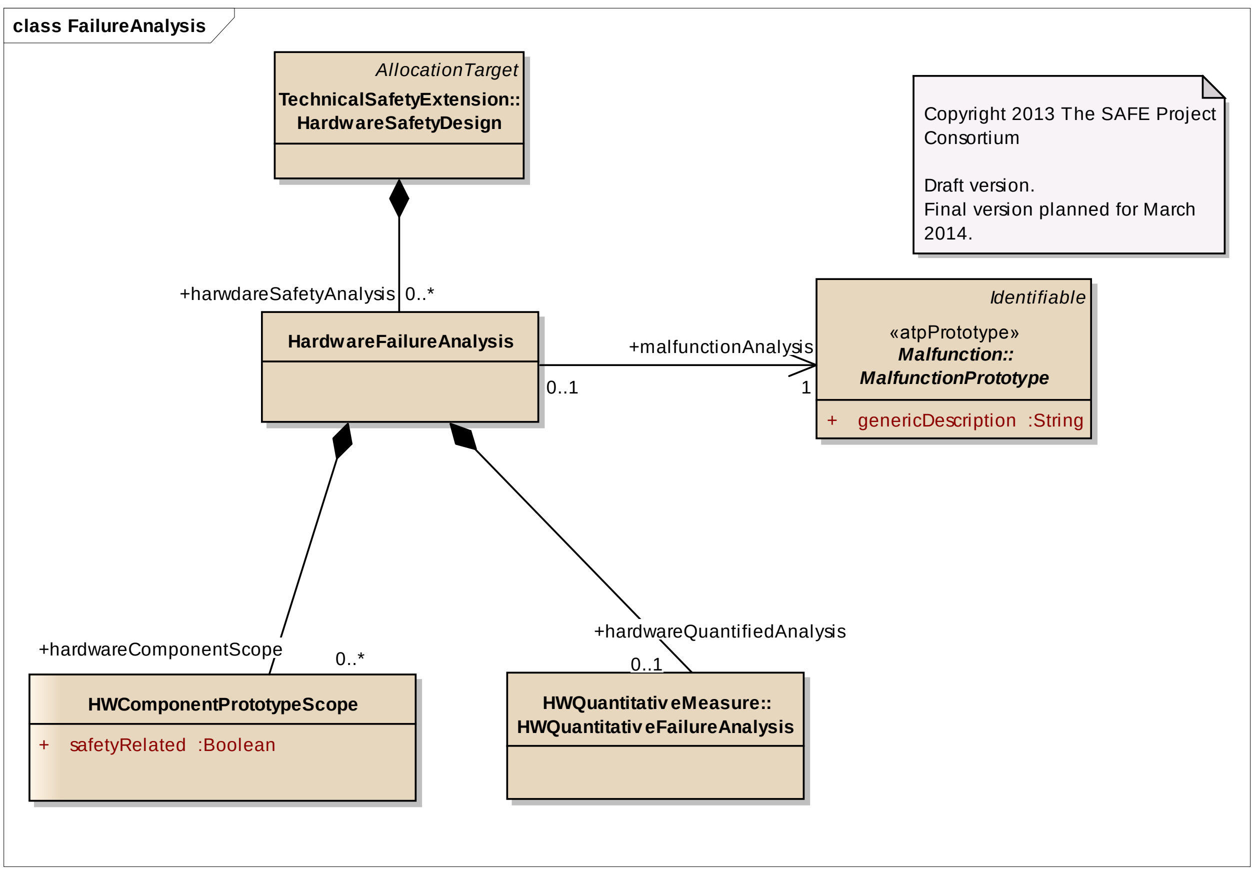


Figure: 26

Diagram "Failure"

Notes:

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

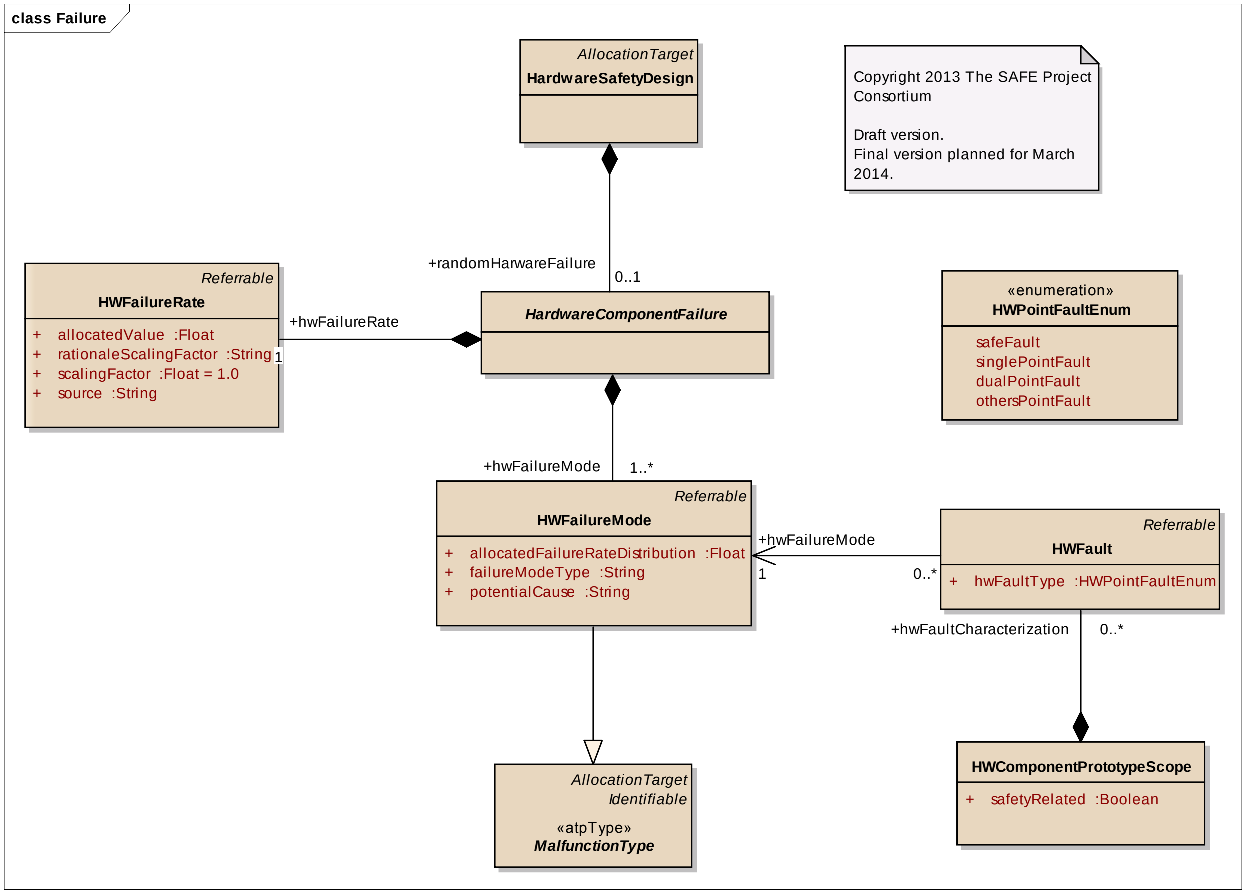


Figure: 27

Diagram "HwComponentScopePrototype"

Notes:

This diagram shows the context for HWComponent for an instanceRef between Type and Prototype

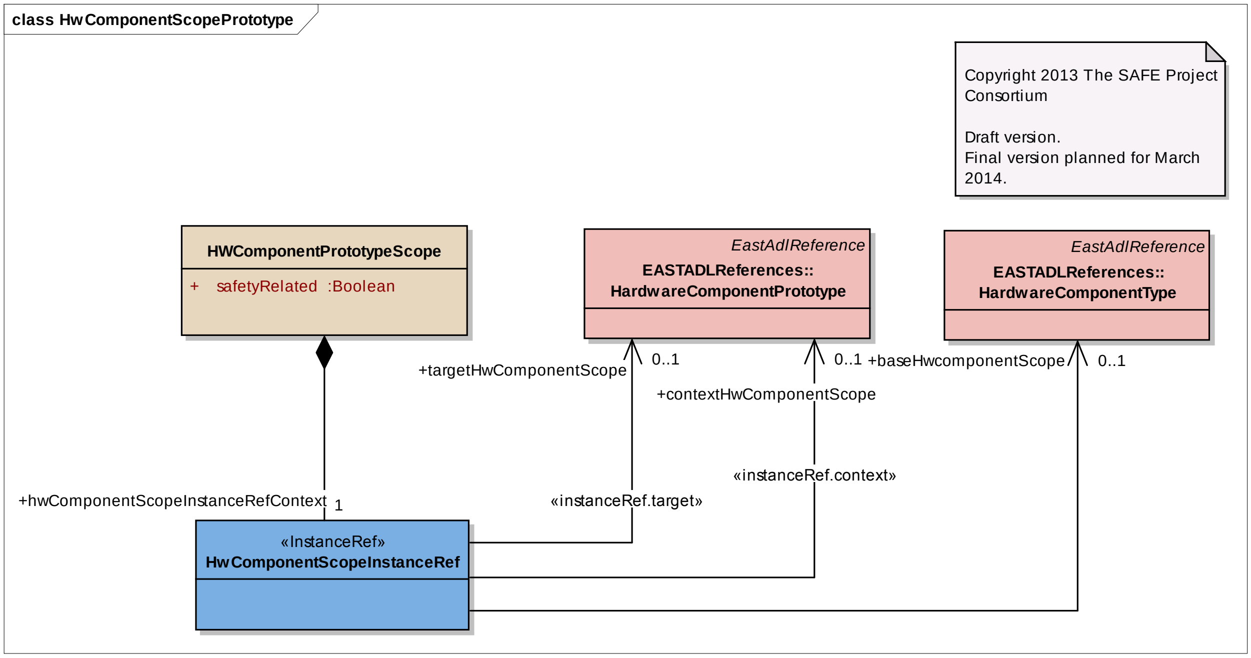


Figure: 28

Diagram "HWQuantitativeElement"

Notes:

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

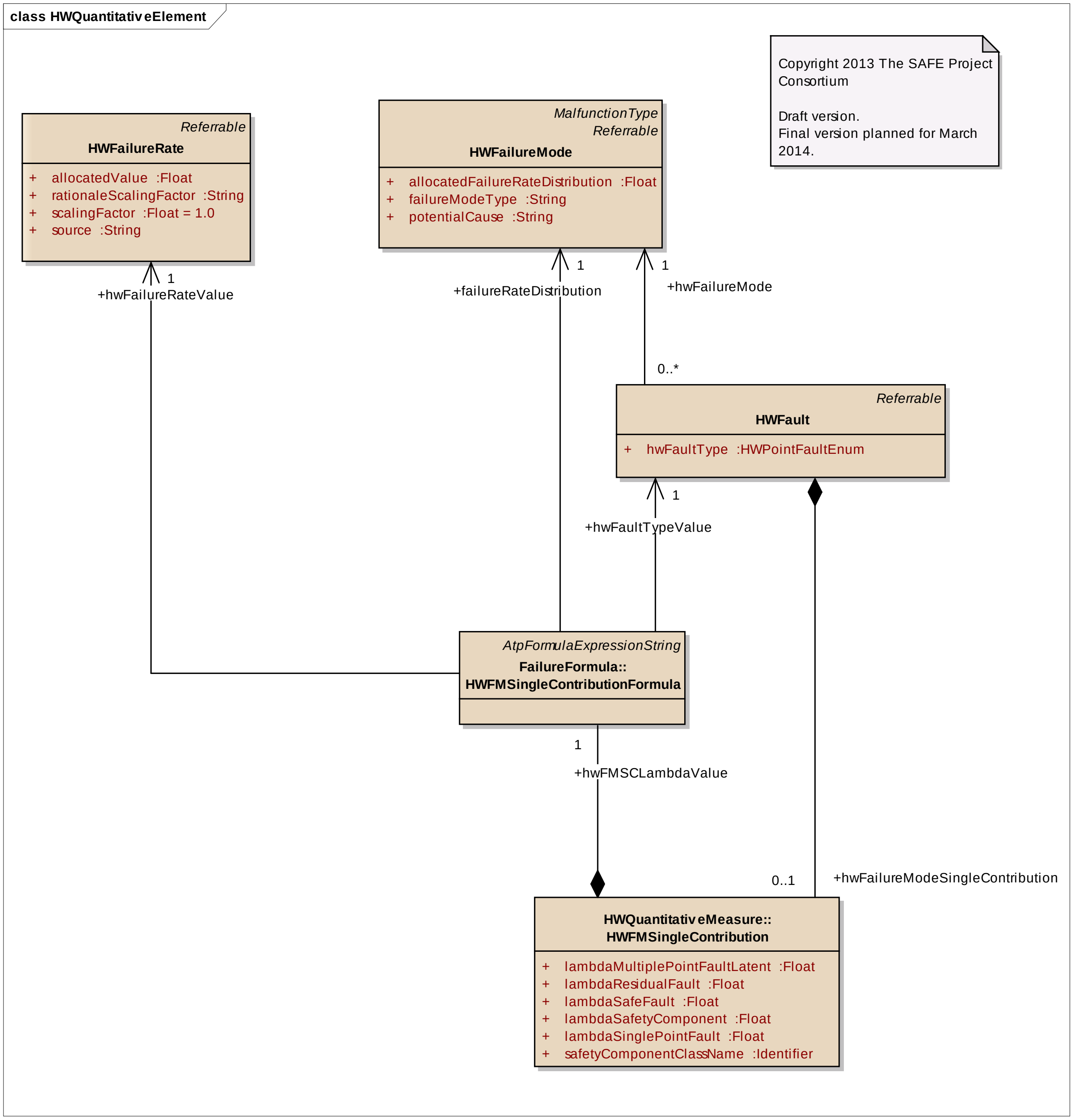


Figure: 29

Diagram "FailureCalculatedFromPart"

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.

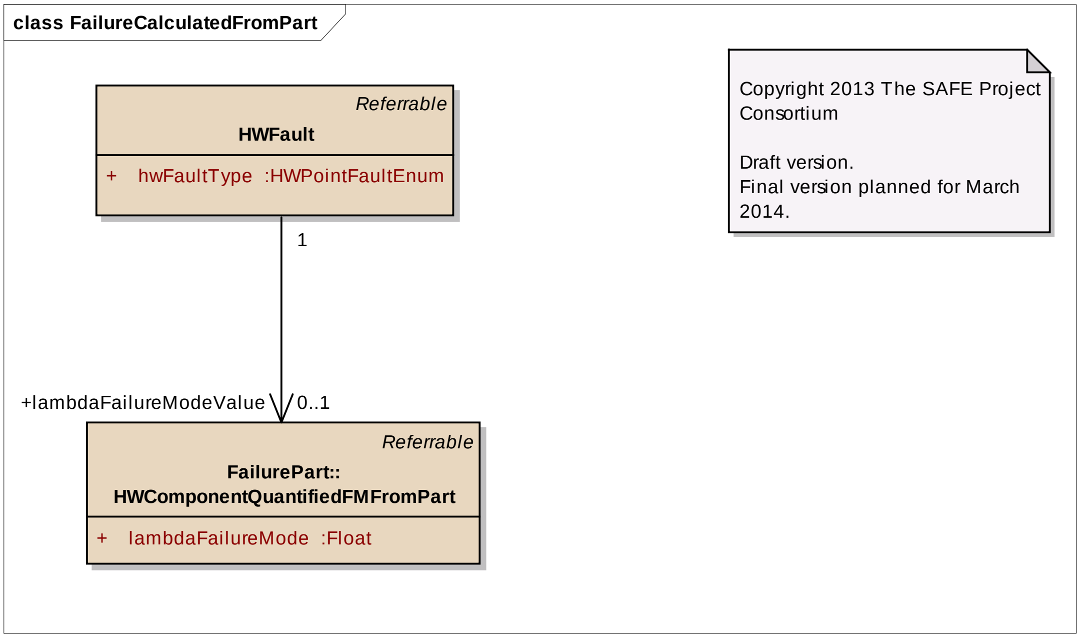


Figure: 30

Element "HardwareFailureAnalysis"

Parent Package: Failure

Stereotype: ,

Notes:

This class represent the container for all Hardware Failure Analysis.

Each safety goal (as malfunction) must lead to a safety analysis, so this class contains all the information related to the analysis as :

- the relation to the malfunction as the malfunctionPrototype for each analysis

- the HwComponentPrototypeScope to identify all hardware component specific to the context as HWComponentProtype inside a type composition

- the HWQuantifiedFailure Analysis performed on the level of the composition

Relationships

Name	Source/Target	Notes
	Source: 0..1 HardwareFailureAnalysis. **Target: 1** MalfunctionPrototype.malfunctionAnalysis	
	Source: 0..* HWComponentPrototypeScope.hardwareComponentScope **Target:** HardwareFailureAnalysis.	

Name	Source/Target	Notes
	Source: **0..1** HWQuantitativeFailureAnalysis.hardwareQuantifiedAnalysis **Target:** HardwareFailureAnalysis.	
	Source: **0..*** HardwareFailureAnalysis.harwdareSafetyAnalysis **Target:** HardwareSafetyDesign.	

Element "HardwareComponentFailure"

Parent Package: Failure

Stereotype: ,

Notes:

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

Relationships

Name	Source/Target	Notes
	Source: **0..1** HardwareComponentFailure.randomHarwareFailure **Target:** HardwareSafetyDesign.	
	Source: **1..*** HWFailureMode.hwFailureMode **Target:** HardwareComponentFailure.	
	Source: **1** HWFailureRate.hwFailureRate **Target:** HardwareComponentFailure.	

Element "HWFailureRate"

Parent Package: Failure

Stereotype: ,

Notes:

This class captures the HWFailureRate of a 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.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes

	allocatedValue	Float			0	0	0		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.
	rationaleScalingFactor	String			0	0	0		The rationaleScalingFactor shall provide a rationale, if a scaling factor different to 1.0 is applied.
	scalingFactor	Float			0	0	0	1.0	The scalingFactor allows potential scaling between different sources of failure rates as described in ISO Part 5 Annex F.
	source	String			0	0	0		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

Relationships

Name	**Source/Target**	**Notes**
	Source: HWFMSingleContributionFormula. **Target: 1** HWFailureRate.hwFailureRateValue	
	Source: HWFailureRate.	

Name	Source/Target	Notes
	Target: Referrable.	
	Source: 1 HWFailureRate.hwFailureRate **Target:** HardwareComponentFailure.	

Element "HWFailureMode"

Parent Package: Failure

Stereotype: ,

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).

The HWfailureMode as a specialization of A malfunction can be traced according Requirement tracing relation to a SafetyMechanismSpecification composed with QuantifiedHWDCProperty class to identify the Diagnosis Coverage value for Latent and/or Residual Fault to be able then to compute HW metrics.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	allocatedFailureRateDistribution	Float			0	0	0		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).
	failureModeType	String			0	0	0		This attribute textually describes the type of a failure mode of a HWComponent (e.g. "No value" for a sensor).
	potentialCause	String			0	0	0		This attribute allows the documentation of the potential cause of the HWComponent failure mode (e.g. high temperature).

Relationships

Name	Source/Target	Notes
	Source: HWFMSingleContributionFormula. **Target:** **1** HWFailureMode.failureRateDistribution	
	Source: **1..*** HWFailureMode.hwFailureMode **Target:** HardwareComponentFailure.	
	Source: HWFailureMode. **Target:** Referrable.	
	Source: **0..*** HWFault. **Target:** **1** HWFailureMode.hwFailureMode	
	Source: HWFailureMode. **Target:** MalfunctionType.	

Element "HWFault"

Parent Package: Failure

Stereotype: ,

Notes:

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

HardwareFault can only exist for HardwareComponentPrototype when HWComponent are used given by the HWComponentPrototypeScope.

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).

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	hwFaultType	HWPointFaultEnum			0	0	0		Characterization of the Failure Mode for a single point related malfunction (linked to violation of a Safety Goal). singlePoint Hw Fault Type can be either SPF or Residual 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)

Relationships

Name	Source/Target	Notes
	Source: **1** HWFault. **Target:** **0..1** HWComponentQuantifiedFMFromPart.lambdaFailureModeValue	
	Source: HWFMSingleContributionFormula. **Target:** **1** HWFault.hwFaultTypeValue	
	Source: **0..1** HWFMSingleContribution.hwFailureModeSingleContribution **Target:** HWFault.	
	Source: HWFault. **Target:** Referrable.	
	Source: HWComponentPrototypeScope. **Target:** **0..*** HWFault.hwFaultCharacterization	
	Source: **0..*** HWFault. **Target:** **1** HWFailureMode.hwFailureMode	

Element "HWComponentPrototypeScope"

Parent Package: Failure

Stereotype: ,

Notes:

This class describes the context for the definition of a hardware component. The attribute defines a results of the analysis if the Hardware Component is safety related means impacted by the contribution to the violation of the malfunction

During modeling a design rule must be ensured that HardwareComponentType used as BaseHwComponent for the instanceRef relation is the same as the root hierarchy on the HwComponentprototypeScope creation

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	safetyRelated	Boolean			0	0	0		This attribute stores

									the contribution of the HWComponent to a malfunction as a boolean information

Relationships

Name	Source/Target	Notes
	Source: 1 HwComponentScopeInstanceRef.hwComponentScopeInstanceRefContext **Target:** HWComponentPrototypeScope.	
	Source: 0..* HWComponentPrototypeScope.hardwareComponentScope **Target:** HardwareFailureAnalysis.	
	Source: HWComponentPrototypeScope. **Target:** 0..* HWFault.hwFaultCharacterization	

Element "HWMultiplePointFaultEnum"

Parent Package: Failure

Stereotype: «enumeration»,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	noMultiplePointFault								
	DualPointFault								
	TriplePointFault								
	othersOrderPointFault								

Element "HWPointFaultEnum"

Parent Package: Failure

Stereotype: «enumeration»,

Notes:

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

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

SinglePointFault represents a first order fault, while DualPointFault represents a second order fault, and OthersPointFault order greater then two. 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.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	safeFault				0	0	0		This literal describes the characterization as a safe fault.
	singlePointFault				0	0	0		This literal describes the characterization as a single-point of failure (direct violation).
	dualPointFault				0	0	0		This literal describes the characterization as a multiple-point fault (violation in combination with another independent fault).
	othersPointFault								

Element "HwComponentScopeInstanceRef"

Parent Package: Failure

Stereotype: «InstanceRef»,

Notes:

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

Relationships

Name	Source/Target	Notes

Name	Source/Target	Notes
	Source: **1** HwComponentScopeInstanceRef.hwComponentScopeInstanceRefContext **Target:** HWComponentPrototypeScope.	
	Source: HwComponentScopeInstanceRef. **Target:** **0..1** HardwareComponentType.baseHwcomponentScope	
	Source: HwComponentScopeInstanceRef. **Target:** **0..1** HardwareComponentPrototype.contextHwComponentScope	
	Source: HwComponentScopeInstanceRef. **Target:** **0..1** HardwareComponentPrototype.targetHwComponentScope	

Package "FailurePart"

Type of Package: **Package**

Parent Package: Hardware

Notes:

Diagram "FailurePartAnalysis"

Notes:

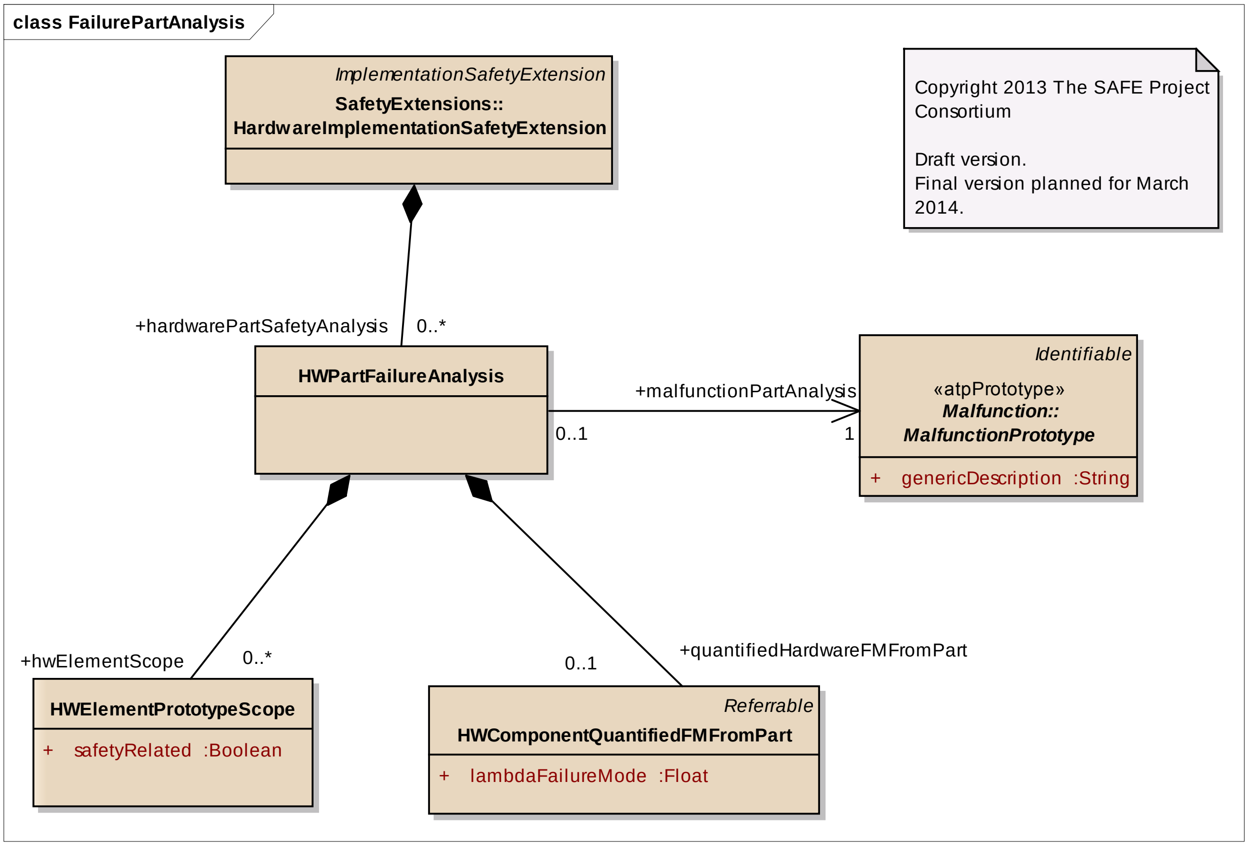


Figure: 31

Diagram "PartFailure"

Notes:

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

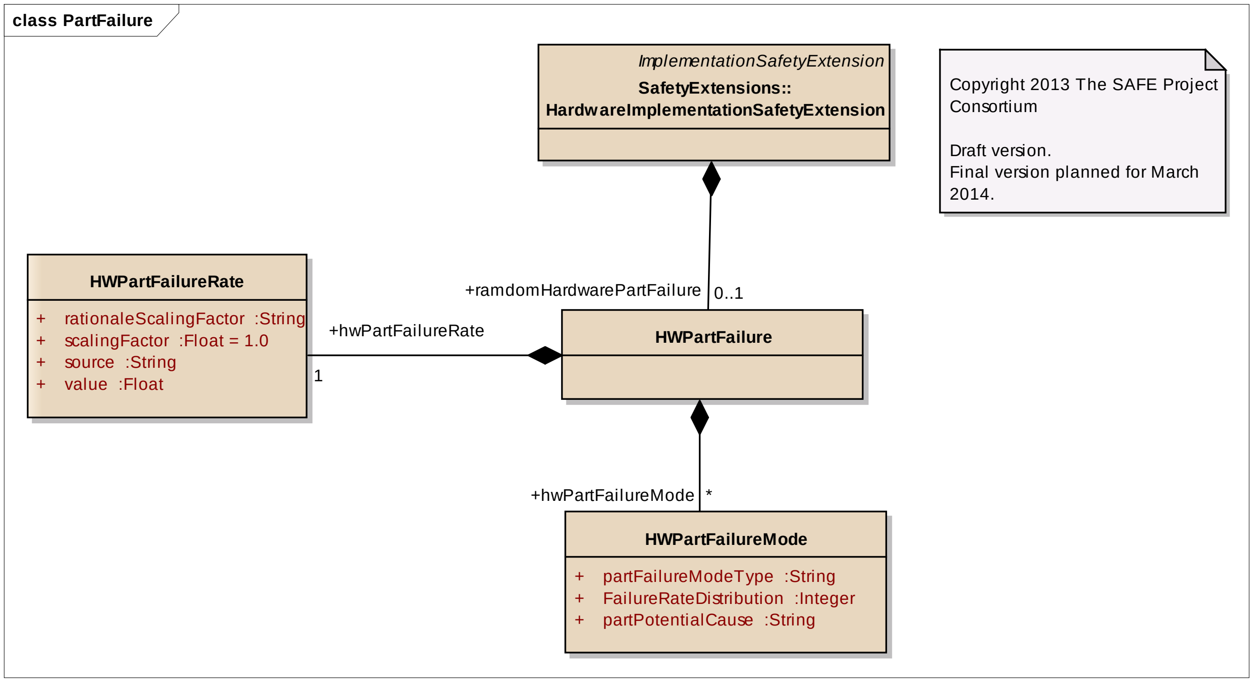


Figure: 32

Diagram "HwElementScopePrototype"

Notes:

This diagram shows the instance reference of a Hardware Component Quantified failure Value issued from Hardware Part.

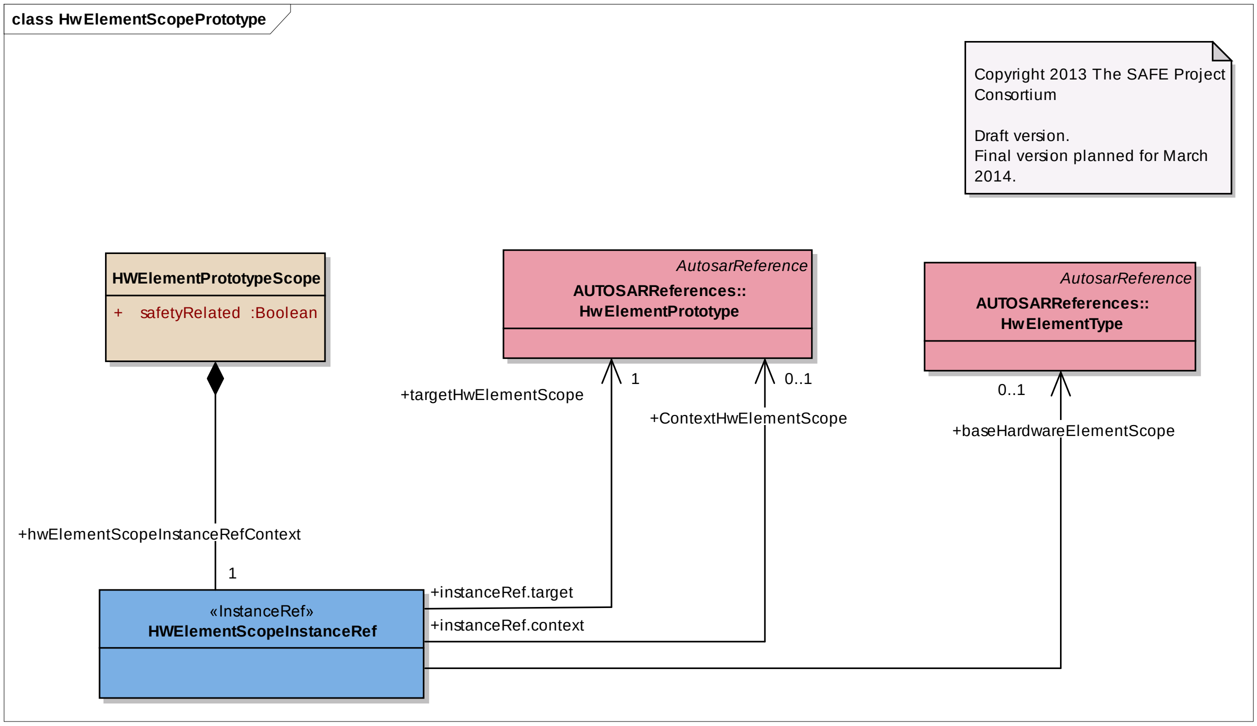


Figure: 33

Diagram "HwPartContributionToComponent"

Notes:

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

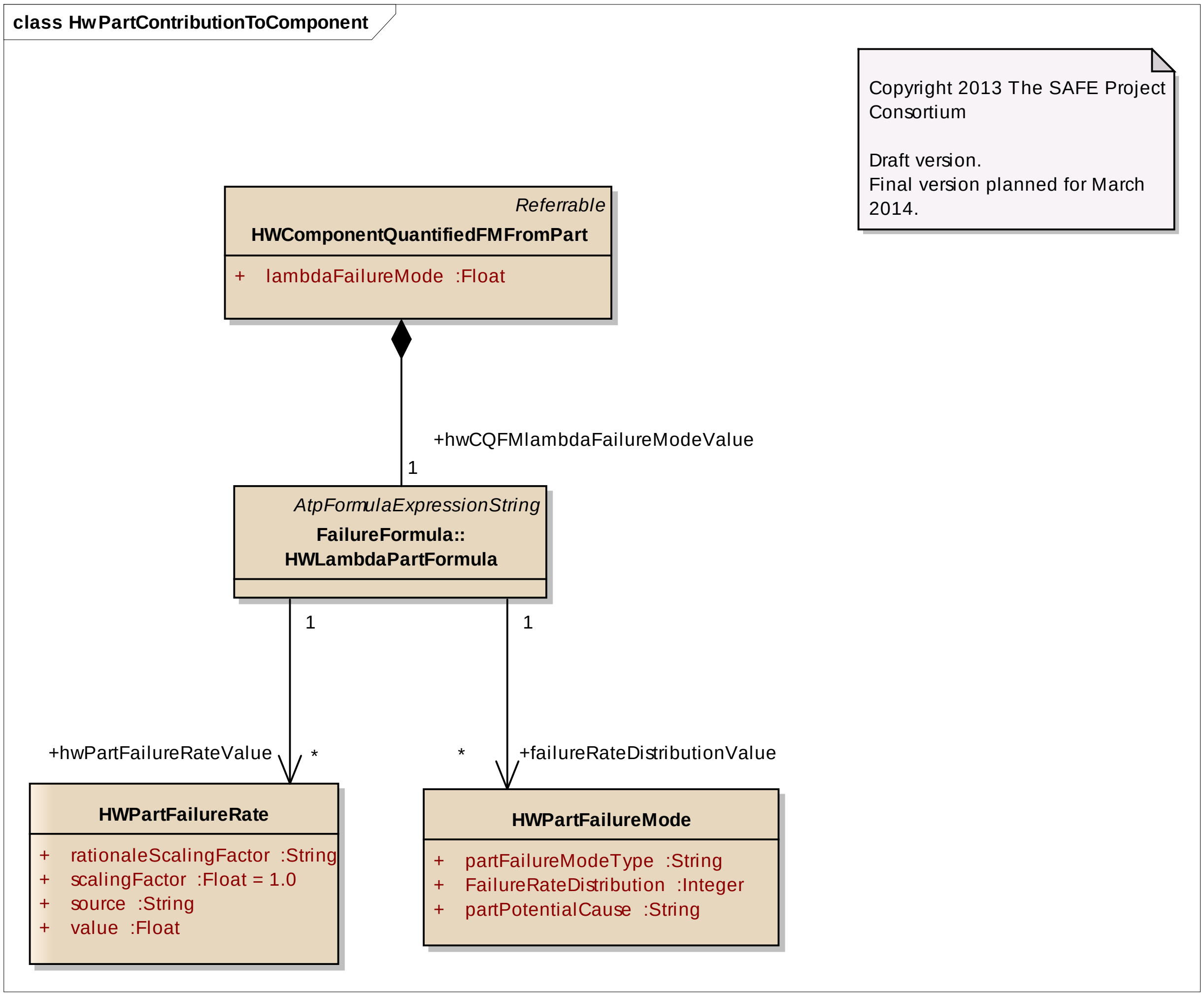


Figure: 34

Element "HWComponentQuantifiedFMFromPart"

Parent Package: FailurePart

Stereotype: ,

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.

During modeling a design rule must be ensured that AutosarHWElementType used as BaseHwElement for the instanceRef relation is the same as the root hierarchy on the HwElement creation

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	lambdaFailureMode	Float			0	0	0		This attribute contains the quantified failure rate for the corresponding failure mode of the hardware component.

Relationships

Name	Source/Target	Notes
	Source: **1** HWFault. **Target:** **0..1** HWComponentQuantifiedFMFromPart.lambdaFailureModeValue	
	Source: **1** HWLambdaPartFormula.hwCQFMlambdaFailureModeValue **Target:** HWComponentQuantifiedFMFromPart.	
	Source: HWPartFailureAnalysis. **Target:** **0..1** HWComponentQuantifiedFMFromPart.quantifiedHardwareFMFromPart	
	Source: HWComponentQuantifiedFMFromPart. **Target:** Referrable.	

Element "HWElementPrototypeScope"

Parent Package: FailurePart

Stereotype: ,

Notes:

This class describes the context for the definition of a hardware Element. The attribute defines a results of the analysis if the Hardware Component is safety related means impacted by the contribution to the violation of the malfunction

During modeling a design rule must be ensured that HWElementType used as BaseHwComponent for the instanceRef relation is the same as the root hierarchy on the HwElementPrototypeScope creation

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	safetyRelated	Boolean							This attribute stores the contribution of the HWElement to a malfunction as a boolean information

Relationships

Name	Source/Target	Notes
	Source: **0..*** HWElementPrototypeScope.hwElementScope **Target:** HWPartFailureAnalysis.	
	Source: HWElementPrototypeScope. **Target:** **1** HWElementScopeInstanceRef.hwElementScopeInstanceRefContext	

Element "HWPartFailure"

Parent Package: FailurePart

Stereotype: ,

Notes:

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

Relationships

Name	Source/Target	Notes
	Source: **1** HWPartFailureRate.hwPartFailureRate **Target:** HWPartFailure.	
	Source: **0..1** HWPartFailure.ramdomHardwarePartFailure **Target:** HardwareImplementationSafetyExtension.	
	Source: * HWPartFailureMode.hwPartFailureMode **Target:** HWPartFailure.	

Element "HWPartFailureAnalysis"

Parent Package: FailurePart

Stereotype: ,

Notes:

This class represent the container for all Hardware Part Failure Analysis (Autosar HWElement) for a malfunction.

Each malfunction (as Hardware Failure Mode) must lead to a safety part analysis, so this class contains all the information related to the analysis as :

- the relation to the malfunction as the malfunctionPrototype for each analysis

- the HwElementPrototypeScope to identify all hardware Autosar Element specific to the context as HWComponentPrototype inside a type composition

- the HWComponentQuantifiedFMFromPart Analysis performed on the level of the composition of hardware part for contribution to the malfunction

Relationships

Name	Source/Target	Notes
	Source: **0..*** HWElementPrototypeScope.hwElementScope **Target:** HWPartFailureAnalysis.	
	Source: **0..1** HWPartFailureAnalysis. **Target:** **1** MalfunctionPrototype.malfunctionPartAnalysis	
	Source: HardwareImplementationSafetyExtension. **Target:** **0..*** HWPartFailureAnalysis.hardwarePartSafetyAnalysis	
	Source: HWPartFailureAnalysis. **Target:** **0..1** HWComponentQuantifiedFMFromPart.quantifiedHardwareFMFromPart	

Element "HWPartFailureMode"

Parent Package: FailurePart

Stereotype: ,

Notes:

This class describes HWPartFailureModes of a HWPart as AUTOSAR HWElement. It aso captures the potential cause for a 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.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	partFailureModeType	String			0	0	0		This attribute textually describes the type of a failure mode of a HWPart element (e.g. "ShortCircuit" for a resistor).
	FailureRateDistribution	Integer			0	0	0		This attribute describes the distribution of the failure rate of the HWPart element for the specific hardware part failure mode in percentage.
	partPotentialCause	String			0	0	0		This attribute allows the documentation of the potential cause of the HWPart failure mode (e.g. high temperature).

Relationships

Name	Source/Target	Notes
	Source: * HWPartFailureMode.hwPartFailureMode **Target:** HWPartFailure.	
	Source: 1 HWLambdaPartFormula. **Target:** * HWPartFailureMode.failureRateDistributionValue	

Element "HWPartFailureRate"

Parent Package: FailurePart

Stereotype: ,

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.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	rationaleScalingFactor	String			0	0	0		The rationaleScalingFactor shall provide a rationale, if a scaling factor different to 1.0 is applied.
	scalingFactor	Float			0	0	0	1.0	The scalingFactor allows potential scaling between different sources of failure rates as described in ISO Part 5 Annex F.
	source	String			0	0	0		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			0	0	0		FIT rate for the hardware part element. It shall be expressed in FIT.

Relationships

Name	Source/Target	Notes
	Source: **1** HWLambdaPartFormula. **Target:** ***** HWPartFailureRate.hwPartFailureRateValue	
	Source: **1** HWPartFailureRate.hwPartFailureRate **Target:** HWPartFailure.	

Element "HWElementScopeInstanceRef"

Parent Package: FailurePart

Stereotype: «InstanceRef»,

Notes:

This "instanceRef" meta-class is the container for holding the relation of HWElementScope in context of AutosarHWElementPrototype

Relationships

Name	Source/Target	Notes
	Source: HWElementScopeInstanceRef.instanceRef.context **Target:** **0..1** HwElementPrototype.ContextHwElementScope	
	Source: HWElementScopeInstanceRef. **Target:** **0..1** HwElementType.baseHardwareElementScope	
	Source: HWElementScopeInstanceRef.instanceRef.target **Target:** **1** HwElementPrototype.targetHwElementScope	
	Source: HWElementPrototypeScope. **Target:** **1** HWElementScopeInstanceRef.hwElementScopeInstanceRefContext	

Package "HWQuantitativeMeasure"

Type of Package: **Package**

Parent Package: Hardware

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.

Diagram "HWQuantitativeMeasure"

Notes:

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

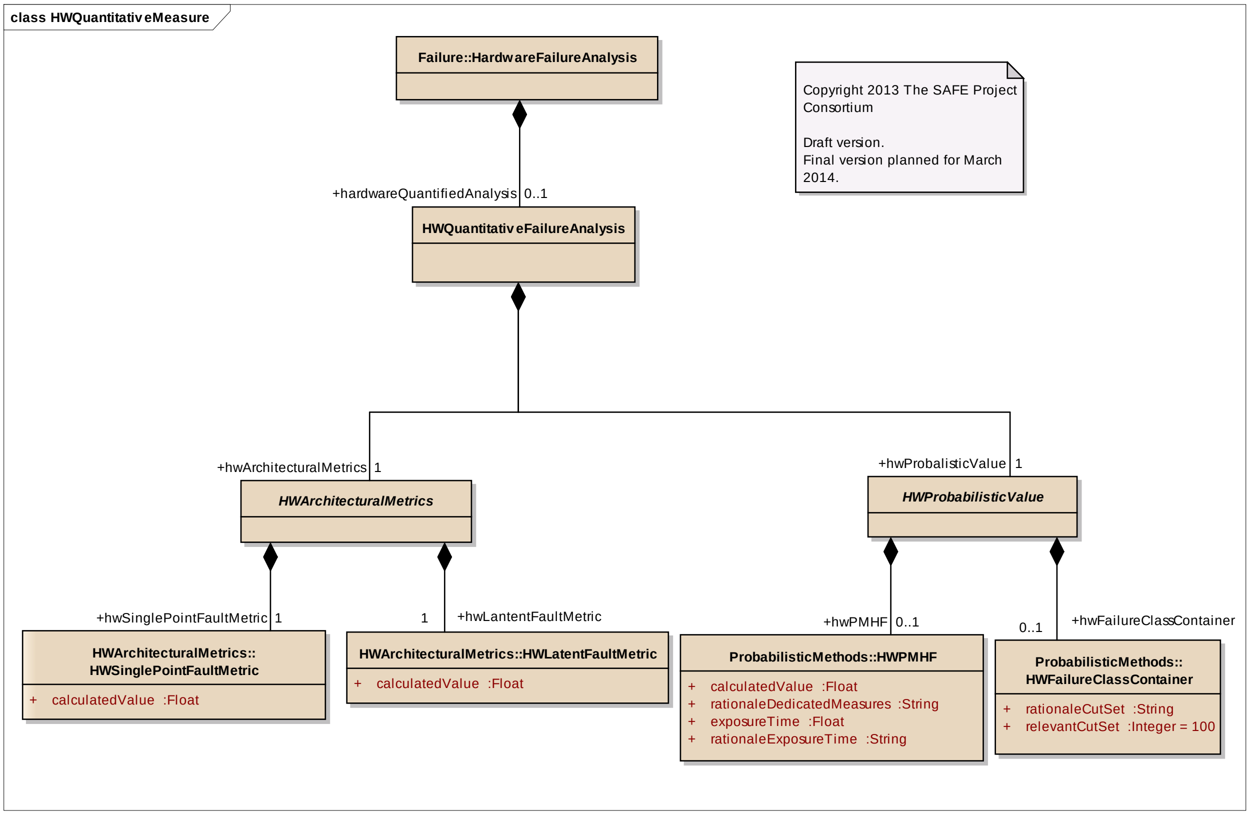


Figure: 35

Element "HWArchitecturalMetrics"

Parent Package: HWQuantitativeMeasure

Stereotype: ,

Notes:

This class represents an abstract definition of all quantified failure analysis required by the ISO Part 5 Clause 8. This class allows to map 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.

Relationships

Name	Source/Target	Notes
	Source: HWSinglePointFaultMetric.hwSinglePointFaultMetric **1** **Target:** HWArchitecturalMetrics.	

Name	Source/Target	Notes
	Source: **1** HWArchitecturalMetrics.hwArchitecturalMetrics **Target:** HWQuantitativeFailureAnalysis.	
	Source: **1** HWLatentFaultMetric.hwLantentFaultMetric **Target:** HWArchitecturalMetrics.	

Element "HWFMSingleContribution"

Parent Package: HWQuantitativeMeasure

Stereotype: ,

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

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	lambdaMultiplePointFaultLatent	Float			0	0	0		This attribute stores the specific failure rate for single failure mode contribution as multiple-point latent, lambda(MPF,L).
	lambdaResidualFault	Float			0	0	0		This attribute stores the specific failure rate for single failure mode contribution as residual fault, lambda(RF).
	lambdaSafeFault	Float			0	0	0		This attribute stores the specific failure rate for single failure mode contribution as safe fault, lambda(SF).
	lambdaSafetyComponent	Float			0	0	0		This attribute stores the sum of specific failure rates for the hardware component

									for verification.
	lambdaSinglePoint Fault	Float			0	0	0		This attribute stores the specific failure rate for single failure mode contribution as single-point fault, lambda(SPF).
	safetyComponent ClassName	Identifie r			0	0	0		This attribute stores the name of the hardware component class.

Relationships

Name	Source/Target	Notes
	Source: HWPMHFFormula. **Target:** * HWFMSingleContribution.lambdaValue	
	Source: HWLatentFaultMetricFormula. **Target:** * HWFMSingleContribution.lambdaValue	
	Source: **1** HWFMSingleContributionFormula.hwFMSCLambdaValue **Target:** HWFMSingleContribution.	
	Source: **0..1** HWFMSingleContribution.hwFailureModeSingleContribution **Target:** HWFault.	
	Source: HWSinglePointFaultMetricFormula. **Target:** * HWFMSingleContribution.lambdaValue	
	Source: HWFailureClassContributionFormula. **Target:** * HWFMSingleContribution.lambdaValue	

Element "HWProbabilisticValue"

Parent Package: HWQuantitativeMeasure

Stereotype: ,

Notes:

This class represents an abstract definition of all failure analysis required by the ISO Part 5 Clause 9. This class allows to map 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.

Relationships

Name	Source/Target	Notes
	Source: **1** HWProbabilisticValue.hwProbalisticValue **Target:** HWQuantitativeFailureAnalysis.	
	Source: **0..1** HWPMHF.hwPMHF **Target:** HWProbabilisticValue.	
	Source: **0..1** HWFailureClassContainer.hwFailureClassContainer **Target:** HWProbabilisticValue.	

Element "HWQuantitativeFailureAnalysis"

Parent Package: HWQuantitativeMeasure

Stereotype: ,

Notes:

This class represent the container for all quantified failure analysis required by the ISO 26262 Part 5 for a dedicated SafetyGoal. This class allows to cluster 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.

Relationships

Name	Source/Target	Notes
	Source: **1** HWProbabilisticValue.hwProbalisticValue **Target:** HWQuantitativeFailureAnalysis.	
	Source: **0..1** HWQuantitativeFailureAnalysis.hardwareQuantifiedAnalysis **Target:** HardwareFailureAnalysis.	
	Source: **1** HWArchitecturalMetrics.hwArchitecturalMetrics **Target:** HWQuantitativeFailureAnalysis.	

Package "HWArchitecturalMetrics"

Type of Package: **Package**

Parent Package: Hardware

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.

Diagram "HWArchitecturalMetrics"

Notes:

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

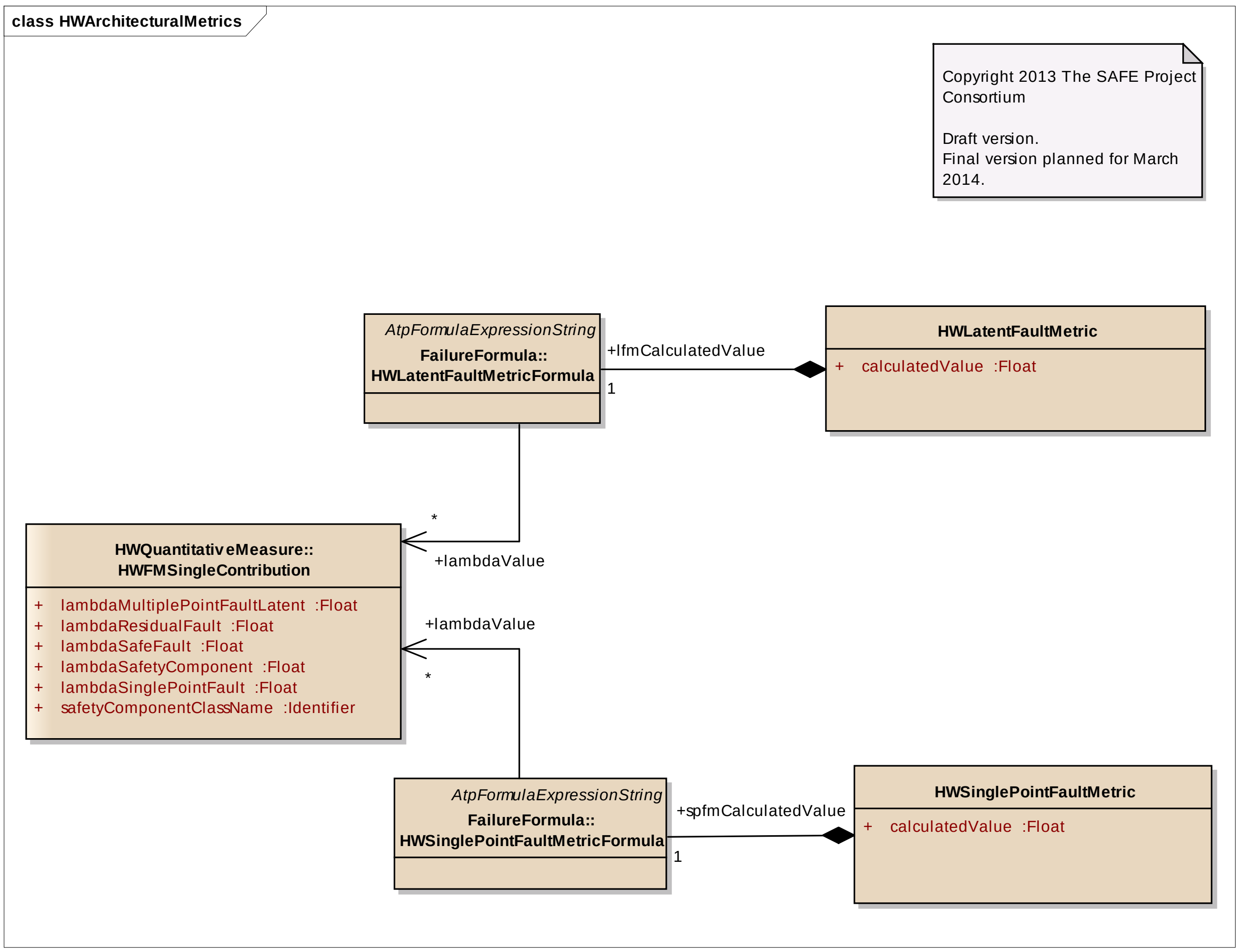


Figure: 36

Element "HWLatentFaultMetric"

Parent Package: HWArchitecturalMetrics

Stereotype: ,

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.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	calculatedValue	Float			0	0	0		The calculatedValue is the result of the calculation of the latent fault metric (in %).

Relationships

Name	Source/Target	Notes
	Source: 1 HWLatentFaultMetric.hwLantentFaultMetric **Target:** HWArchitecturalMetrics.	
	Source: 1 HWLatentFaultMetricFormula.lfmCalculatedValue **Target:** HWLatentFaultMetric.	

Element "HWSinglePointFaultMetric"

Parent Package: HWArchitecturalMetrics

Stereotype: ,

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.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	calculatedValue	Float			0	0	0		The calculatedValue is the result of the calculation of the single-point fault metric (in %).

Relationships

Name	Source/Target	Notes
	Source: HWSinglePointFaultMetricFormula.spfmCalculatedValue **1** **Target:** HWSinglePointFaultMetric.	
	Source: HWSinglePointFaultMetric.hwSinglePointFaultMetric **1** **Target:** HWArchitecturalMetrics.	

Element "TargetValuesLFMetricEnum"

Parent Package: HWArchitecturalMetrics

Stereotype: «enumeration»,

Notes:

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

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	ASIL_D	Float			0	0	0	90.0	This literal contains the target value for latent-fault metric for ASIL-D.
	ASIL_C	Float			0	0	0	80.0	This literal contains the target value for latent-fault metric for ASIL-C.
	ASIL_B	Float			0	0	0	60.0	This literal contains the target value for latent-fault metric for ASIL-B.

Element "TargetValuesSPFMetricEnum"

Parent Package: HWArchitecturalMetrics

Stereotype: «enumeration»,

Notes:

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

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes

	ASIL_D	Float			0	0	0	99.0	This literal contains the target value for single-point fault metric for ASIL-D.
	ASIL_C	Float			0	0	0	97.0	This literal contains the target value for single-point fault metric for ASIL-C.
	ASIL_B	Float			0	0	0	90.0	This literal contains the target value for single-point fault metric for ASIL-B.

Package "ProbabilisticMethods"

Type of Package: **Package**

Parent Package: Hardware

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).

Diagram "ProbabilisticMethods"

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.

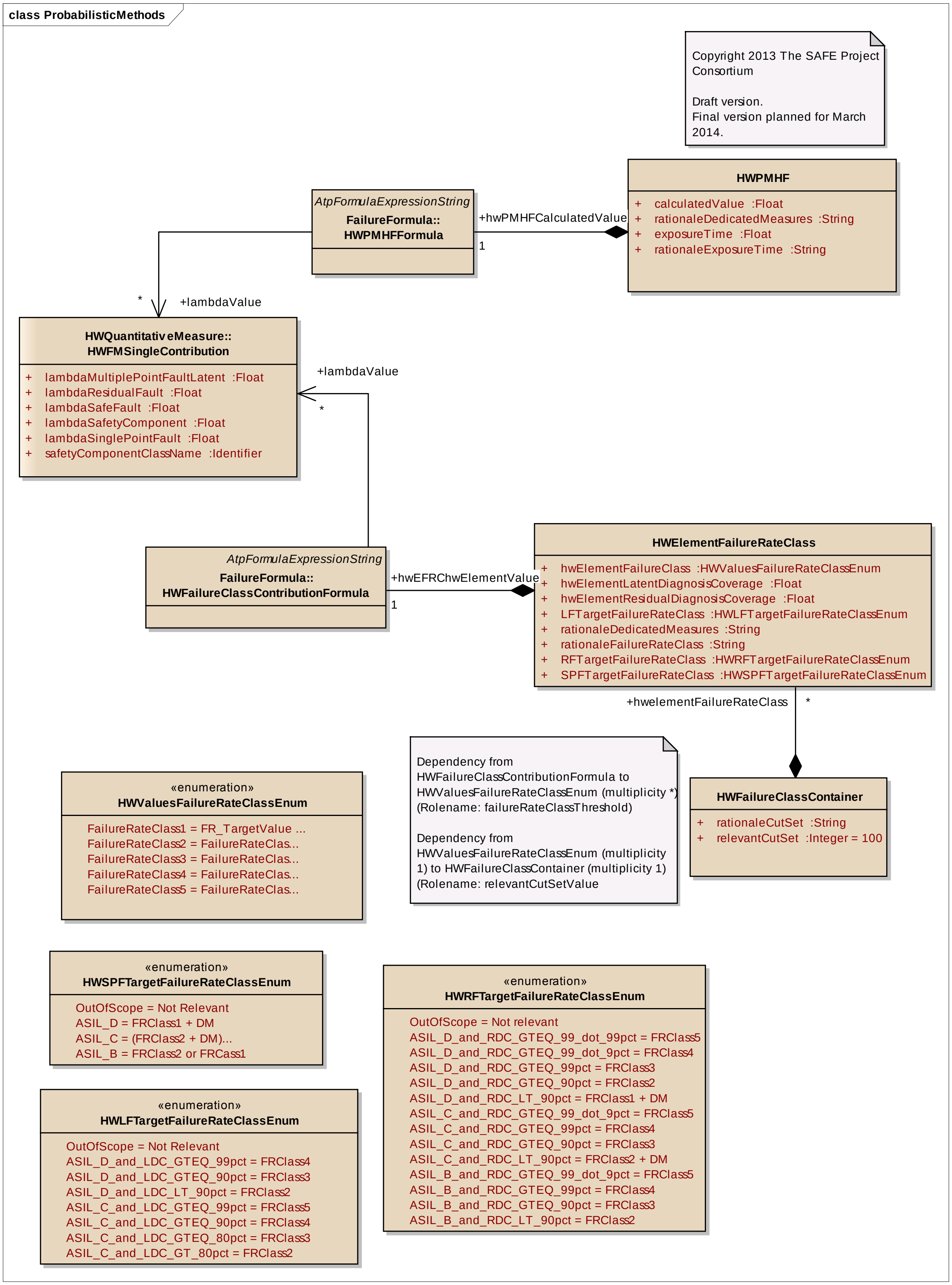

class ProbabilisticMethods
Copyright 2013 The SAFE Project Consortium
Draft version.
Final version planned for March 2014.
HWPMHF
+ calculatedValue :Float
+ rationaleDedicatedMeasures :String
+ exposureTime :Float
+ rationaleExposureTime :String
AtpFormulaExpressionString
FailureFormula::
HWPMHFFormula
+hwPMHFCalculatedValue
1
*
+lambdaValue
HWQuantitativeMeasure::
HWFMSingleContribution
+ lambdaMultiplePointFaultLatent :Float
+ lambdaResidualFault :Float
+ lambdaSafeFault :Float
+ lambdaSafetyComponent :Float
+ lambdaSinglePointFault :Float
+ safetyComponentClassName :Identifier
+lambdaValue
*
AtpFormulaExpressionString
FailureFormula::
HWFailureClassContributionFormula
+hwEFRChwElementValue
1
HWElementFailureRateClass
+ hwElementFailureClass :HWValuesFailureRateClassEnum
+ hwElementLatentDiagnosisCoverage :Float
+ hwElementResidualDiagnosisCoverage :Float
+ LFTargetFailureRateClass :HWLFTargetFailureRateClassEnum
+ rationaleDedicatedMeasures :String
+ rationaleFailureRateClass :String
+ RFTargetFailureRateClass :HWRFTargetFailureRateClassEnum
+ SPFTargetFailureRateClass :HWSPFTargetFailureRateClassEnum
+hwelementFailureRateClass
*
Dependency from HWFailureClassContributionFormula to HWValuesFailureRateClassEnum (multiplicity *) (Rolename: failureRateClassThreshold)
Dependency from HWValuesFailureRateClassEnum (multiplicity 1) to HWFailureClassContainer (multiplicity 1) (Rolename: relevantCutSetValue
HWFailureClassContainer
+ rationaleCutSet :String
+ relevantCutSet :Integer = 100
«enumeration»
HWValuesFailureRateClassEnum
FailureRateClass1 = FR_TargetValue ...
FailureRateClass2 = FailureRateClas...
FailureRateClass3 = FailureRateClas...
FailureRateClass4 = FailureRateClas...
FailureRateClass5 = FailureRateClas...
«enumeration»
HWSPFTargetFailureRateClassEnum
OutOfScope = Not Relevant
ASIL_D = FRClass1 + DM
ASIL_C = (FRClass2 + DM)...
ASIL_B = FRClass2 or FRCass1
«enumeration»
HWRFTargetFailureRateClassEnum
OutOfScope = Not relevant
ASIL_D_and_RDC_GTEQ_99_dot_99pct = FRClass5
ASIL_D_and_RDC_GTEQ_99_dot_9pct = FRClass4
ASIL_D_and_RDC_GTEQ_99pct = FRClass3
ASIL_D_and_RDC_GTEQ_90pct = FRClass2
ASIL_D_and_RDC_LT_90pct = FRClass1 + DM
ASIL_C_and_RDC_GTEQ_99_dot_9pct = FRClass5
ASIL_C_and_RDC_GTEQ_99pct = FRClass4
ASIL_C_and_RDC_GTEQ_90pct = FRClass3
ASIL_C_and_RDC_LT_90pct = FRClass2 + DM
ASIL_B_and_RDC_GTEQ_99_dot_9pct = FRClass5
ASIL_B_and_RDC_GTEQ_99pct = FRClass4
ASIL_B_and_RDC_GTEQ_90pct = FRClass3
ASIL_B_and_RDC_LT_90pct = FRClass2
«enumeration»
HWLFTargetFailureRateClassEnum
OutOfScope = Not Relevant
ASIL_D_and_LDC_GTEQ_99pct = FRClass4
ASIL_D_and_LDC_GTEQ_90pct = FRClass3
ASIL_D_and_LDC_LT_90pct = FRClass2
ASIL_C_and_LDC_GTEQ_99pct = FRClass5
ASIL_C_and_LDC_GTEQ_90pct = FRClass4
ASIL_C_and_LDC_GTEQ_80pct = FRClass3
ASIL_C_and_LDC_GT_80pct = FRClass2

Figure: 37

Element "HWElementFailureRateClass"

Parent Package: ProbabilisticMethods

Stereotype: ,

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.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	hwElementFailureClass	HWValuesFailureRateClassEnum			0	0	0		Failure Rate Class taken from HWValuesRateClassEnum based on the failure rate of the hardware component.
	hwElementLatentDiagnosisCoverage	Float			0	0	0		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			0	0	0		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			0	0	0		Target Failure Rate Class for multiple-point latent faults, taken from HWLFTargetFailureRateClassEnum.
	rationaleDedicatedMeasures	String			0	0	0		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 (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			0	0	0		Rationale for matching criteria on Failure Rate Class.
	RFTargetFailureRateClass	HWRFTargetFailureRateClassEnum			0	0	0		Target Failure Rate Class for residual faults, taken from HWRFTargetFailureRateClassEnum.
	SPFTargetFailureRateClass	HWSPFTargetFailureRateClassEnum			0	0	0		Target Failure Rate Class for single-point faults, taken from HWSPFTargetFailureRateClassEnum.

Relationships

Name	Source/Target	Notes
	Source: **1** HWFailureClassContributionFormula.hwEFRChwElementValue **Target:** HWElementFailureRateClass.	
	Source: ***** HWElementFailureRateClass.hwelementFailureRateClass **Target:** HWFailureClassContainer.	

Element "HWFailureClassContainer"

Parent Package: ProbabilisticMethods

Stereotype: ,

Notes:

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

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	rationaleCutSet	String			0	0	0		This attribute provides a textual rationale for the number of relevant

									cut-sets.
	relevantCutSet	Integer			0	0	0	100	This attributes stores the number of relevant cut sets.

Relationships

Name	Source/Target	Notes
	Source: HWFailureClassContainer.hwFailureClassContainer **0..1** **Target:** HWProbabilisticValue.	
	Source: HWElementFailureRateClass.hwelementFailureRateClass ***** **Target:** HWFailureClassContainer.	

Element "HWLFTargetFailureRateClassEnum"

Parent Package: ProbabilisticMethods

Stereotype: «enumeration»,

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.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	OutOfScope	String			0	0	0	Not Relevant	This literal describes values which are out of scope for the analysis.
	ASIL_D_and_LDC_GTEQ_99pct	String			0	0	0	FRClass4	This literal describes a single cell with value target failure rate class 4 in the table for ASIL-D and latent diagnostic coverage >= 99%.

	ASIL_D_and_LDC_GTEQ_90pct	String			0	0	0	FRClass3	This literal describes a single cell with value target failure rate class 3 in the table for ASIL-D and latent diagnostic coverage >= 90%.
	ASIL_D_and_LDC_LT_90pct	String			0	0	0	FRClass2	This literal describes a single cell with value target failure rate class 2 in the table for ASIL-D and latent diagnostic coverage < 90%.
	ASIL_C_and_LDC_GTEQ_99pct	String			0	0	0	FRClass5	This literal describes a single cell with value target failure rate class 5 in the table for ASIL-C and latent diagnostic coverage >= 99%.
	ASIL_C_and_LDC_GTEQ_90pct	String			0	0	0	FRClass4	This literal describes a single cell with value target failure rate class 4 in the table for ASIL-C and latent diagnostic coverage >= 90%.
	ASIL_C_and_LDC_GTEQ_80pct	String			0	0	0	FRClass3	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.
	ASIL_C_and_LDC_GT_80pct	String			0	0	0	FRClass2	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.

Element "HWPMHF"

Parent Package: ProbabilisticMethods

Stereotype: ,

Notes:

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

A simplified alculation is included in the class HWPMHFFormula.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	calculatedValue	Float			0	0	0		The calculatedValue is the result of the calculation of the PMHF (in FIT).
	rationaleDedicatedMeasures	String			0	0	0		The attribute rationaleDedicatedMeasures shall allow to define a rationale for applied dedicated measures in the design.
	exposureTime	Float			0	0	0		The exposure time is the duration of exposure, shall be expressed in h.
	rationaleExposureTime	String			0	0	0		The attribute rationaleExposureTime is for Documentation of rationale for Exposure Time.

Relationships

Name	Source/Target	Notes
	Source: 0..1 HWPMHF.hwPMHF	

Name	Source/Target	Notes
	Target: HWProbabilisticValue.	
	Source: 1 HWPMHFFormula.hwPMHFCalculatedValue **Target:** HWPMHF.	

Element "HWRFTargetFailureRateClassEnum"

Parent Package: ProbabilisticMethods

Stereotype: «enumeration»,

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.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	OutOfScope	String			0	0	0	Not relevant	This literal describes values which are out of scope for the analysis.
	ASIL_D_and_RDC_GTEQ_99_dot_99pct	String			0	0	0	FRClass5	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.
	ASIL_D_and_RDC_GTEQ_99_dot_9pct	String			0	0	0	FRClass4	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%.
	ASIL_D_and_RDC_GTEQ_99pct	String			0	0	0	FRClass3	This literal describes a single cell with value target failure rate class 3 in the table for ASIL-D and residual fault diagnostic coverage >= 99%.
	ASIL_D_and_RDC_GTEQ_90pct	String			0	0	0	FRClass2	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%.
	ASIL_D_and_RDC_LT_90pct	String			0	0	0	FRClass1 + DM	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%.
	ASIL_C_and_RDC_GTEQ_99_dot_9pct	String			0	0	0	FRClass5	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%.
	ASIL_C_and_RDC_GTEQ_99pct	String			0	0	0	FRClass4	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%.
	ASIL_C_and_RDC_GTEQ_90pct	String			0	0	0	FRClass3	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%.
	ASIL_C_and_RDC_LT_90pct	String			0	0	0	FRClass2 + DM	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%.
	ASIL_B_and_RDC_GTEQ_99_dot_9pct	String			0	0	0	FRClass5	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%.
	ASIL_B_and_RDC_GTEQ_99pct	String			0	0	0	FRClass4	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%.
	ASIL_B_and_RDC_GTEQ_90pct	String			0	0	0	FRClass3	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%.
	ASIL_B_and_RDC_LT_90pct	String			0	0	0	FRClass2	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%.

Element "HWSPFTargetFailureRateClassEnum"

Parent Package: ProbabilisticMethods

Stereotype: «enumeration»,

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.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	OutOfScope	String			0	0	0	Not Relevant	This literal describes values which are out of scope for the analysis.
	ASIL_D	String			0	0	0	FRClass1 + DM	This literal describes a single cell with value target failure rate class 1 + dedicated measures in the table for ASIL-D.
	ASIL_C	String			0	0	0	(FRClass2 + DM) or FRClass1	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.
	ASIL_B	String			0	0	0	FRClass2 or FRCass1	This literal describes a single cell with value target failure rate class 2 or failure rate class 1 in the

									table for ASIL-B.

Element "HWTargetValuesPMHFEnum"

Parent Package: ProbabilisticMethods

Stereotype: «enumeration»,

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

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	ASIL_D	Float			0	0	0	10.0	This attributes stores target value for PMHF for ASIL-D.
	ASIL_C	Float			0	0	0	100.0	This attributes stores target value for PMHF for ASIL-C.
	ASIL_B	Float			0	0	0	100.0	This attributes stores target value for PMHF for ASIL-B.

Element "HWValuesFailureRateClassEnum"

Parent Package: ProbabilisticMethods

Stereotype: «enumeration»,

Notes:

FailureRateClass value correspond 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 are based on the number of relevant cutset.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	FailureRateClass1	Float			0	0	0	FR_TargetV	This attribute contains the

								alue / relevantCutSet	maximum value for failure rate class 1 ranking (in FIT). It is computed from the allocated FIT rate to the analysis (as targetValue of HWFailureClassContainer) derided by the number of relevant cut-set of the analysis (as relevantCutSet of HWFailureClassContainer)
	FailureRateClass2	Float			0	0	0	FailureRateClass1 * 10	This attribute contains the maximum value for failure rate class 2 ranking (in FIT).
	FailureRateClass3	Float			0	0	0	FailureRateClass2 * 10	This attribute contains the maximum value for failure rate class 3 ranking (in FIT).
	FailureRateClass4	Float			0	0	0	FailureRateClass3 * 10	This attribute contains the maximum value for failure rate class 4 ranking (in FIT).
	FailureRateClass5	Float			0	0	0	FailureRateClass4 * 10	This attribute contains the maximum value for failure rate class 5 ranking (in FIT).

Package "Hazards"

Type of Package: **Package**

Parent Package: SAFE Meta-Model

Notes:

Diagram "Hazards"

Notes:

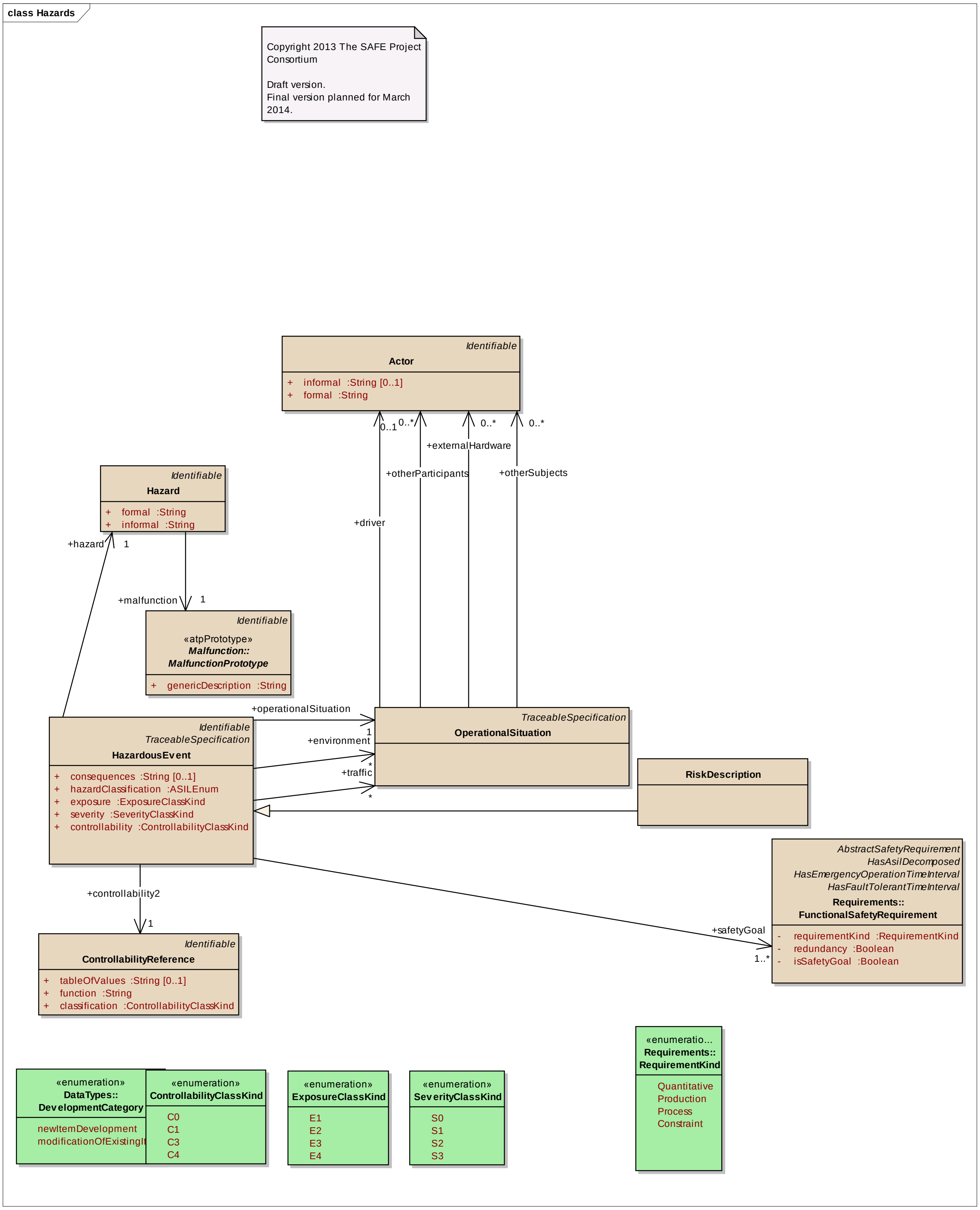

class Hazards
Copyright 2013 The SAFE Project Consortium
Draft version.
Final version planned for March 2014.
Identifiable
Actor
+ informal :String [0..1]
+ formal :String
0..1
0..*
0..*
0..*
+externalHardware
+otherParticipants
+otherSubjects
+driver
Identifiable
Hazard
+ formal :String
+ informal :String
+hazard
1
+malfunction
1
Identifiable
«atpPrototype»
Malfunction::
MalfunctionPrototype
+ genericDescription :String
+operationalSituation
1
+environment
*
+traffic
*
TraceableSpecification
OperationalSituation
Identifiable
TraceableSpecification
HazardousEvent
+ consequences :String [0..1]
+ hazardClassification :ASILEnum
+ exposure :ExposureClassKind
+ severity :SeverityClassKind
+ controllability :ControllabilityClassKind
RiskDescription
AbstractSafetyRequirement
HasAsilDecomposed
HasEmergencyOperationTimeInterval
HasFaultTolerantTimeInterval
Requirements::
FunctionalSafetyRequirement
- requirementKind :RequirementKind
- redundancy :Boolean
- isSafetyGoal :Boolean
+safetyGoal
1..*
+controllability2
1
Identifiable
ControllabilityReference
+ tableOfValues :String [0..1]
+ function :String
+ classification :ControllabilityClassKind
«enumeration»
DataTypes::
DevelopmentCategory
newItemDevelopment
modificationOfExistingIt
«enumeration»
ControllabilityClassKind
C0
C1
C2
C3
C4
«enumeration»
ExposureClassKind
E1
E2
E3
E4
«enumeration»
SeverityClassKind
S0
S1
S2
S3
«enumeratio...
Requirements::
RequirementKind
Quantitative
Production
Process
Constraint

Figure: 38

Diagram "HazardandRiskModel"

Notes:

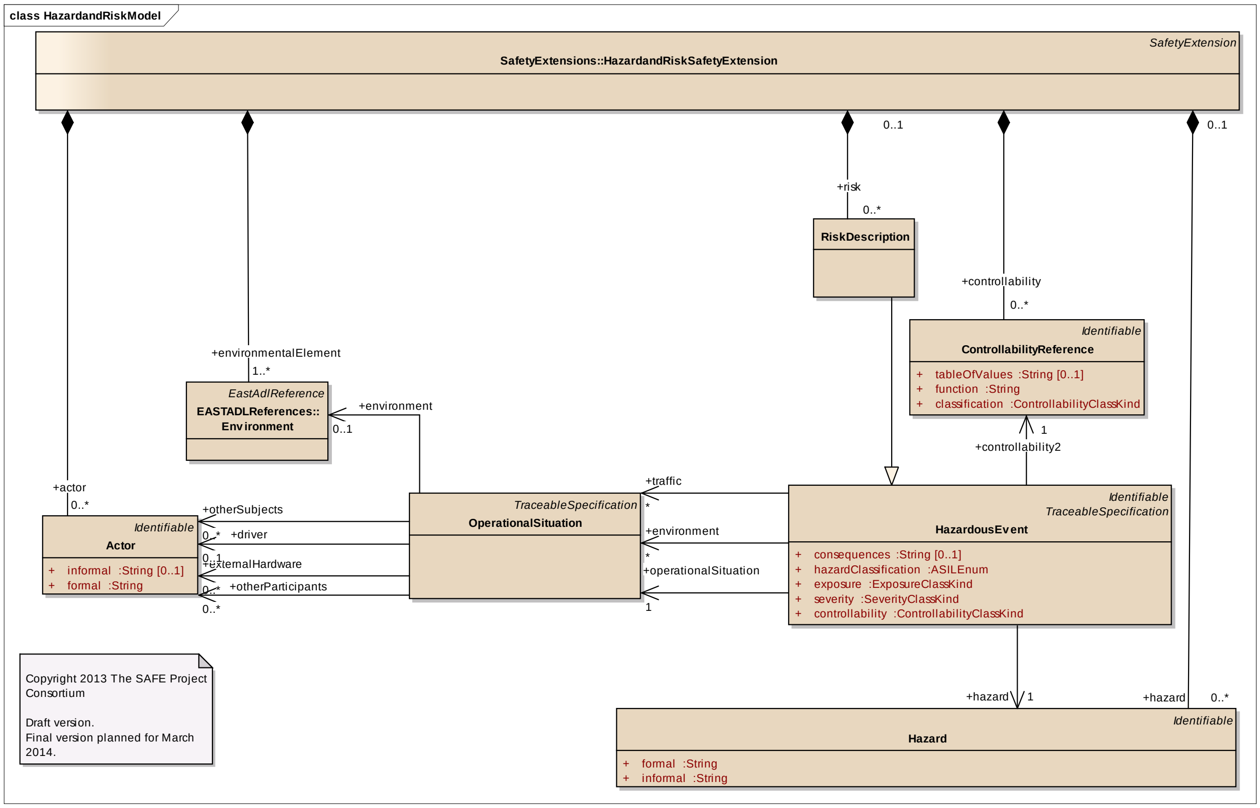


Figure: 39

Diagram "Hazards_elements"

Notes:

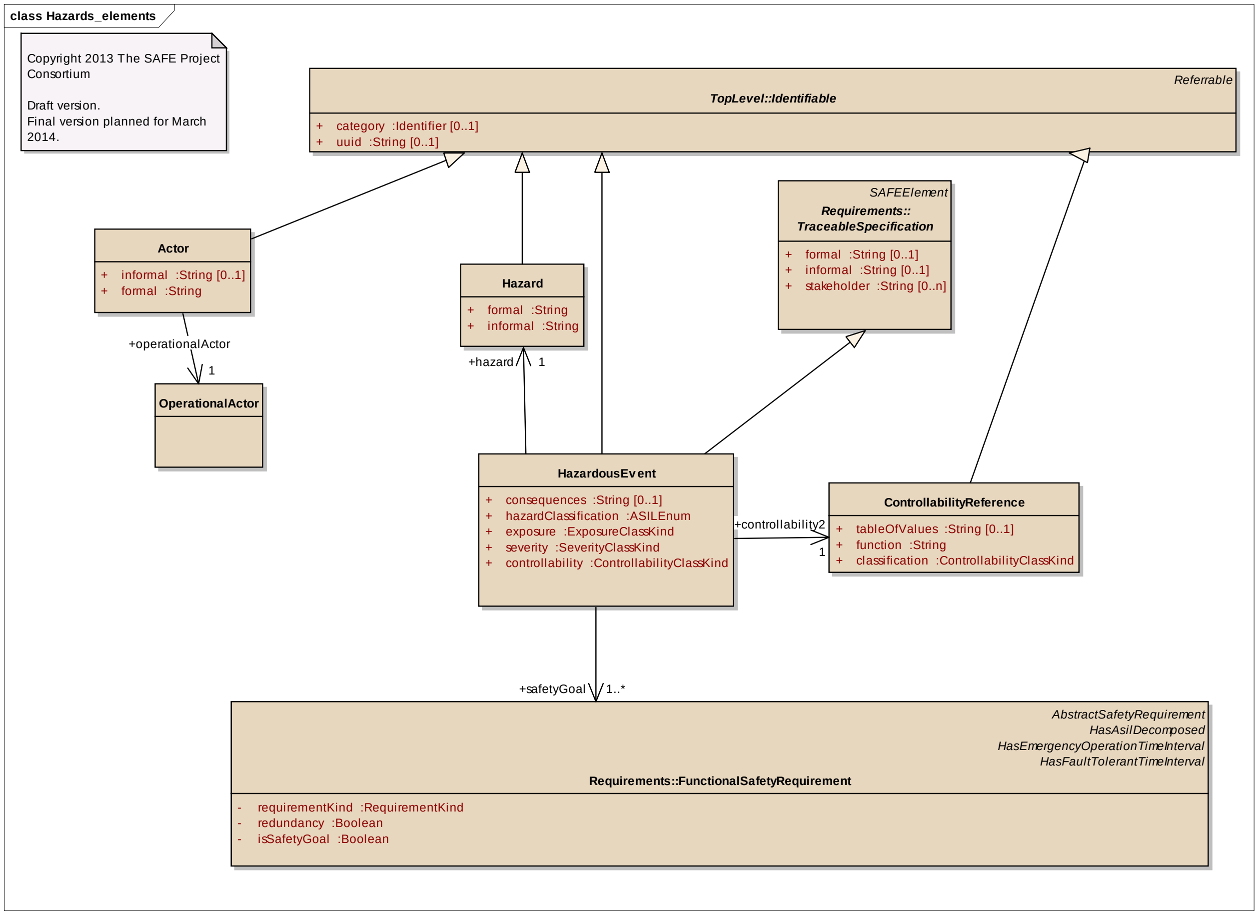


Figure: 40

Diagram "HazardsWithOperationalCondition"

Notes:

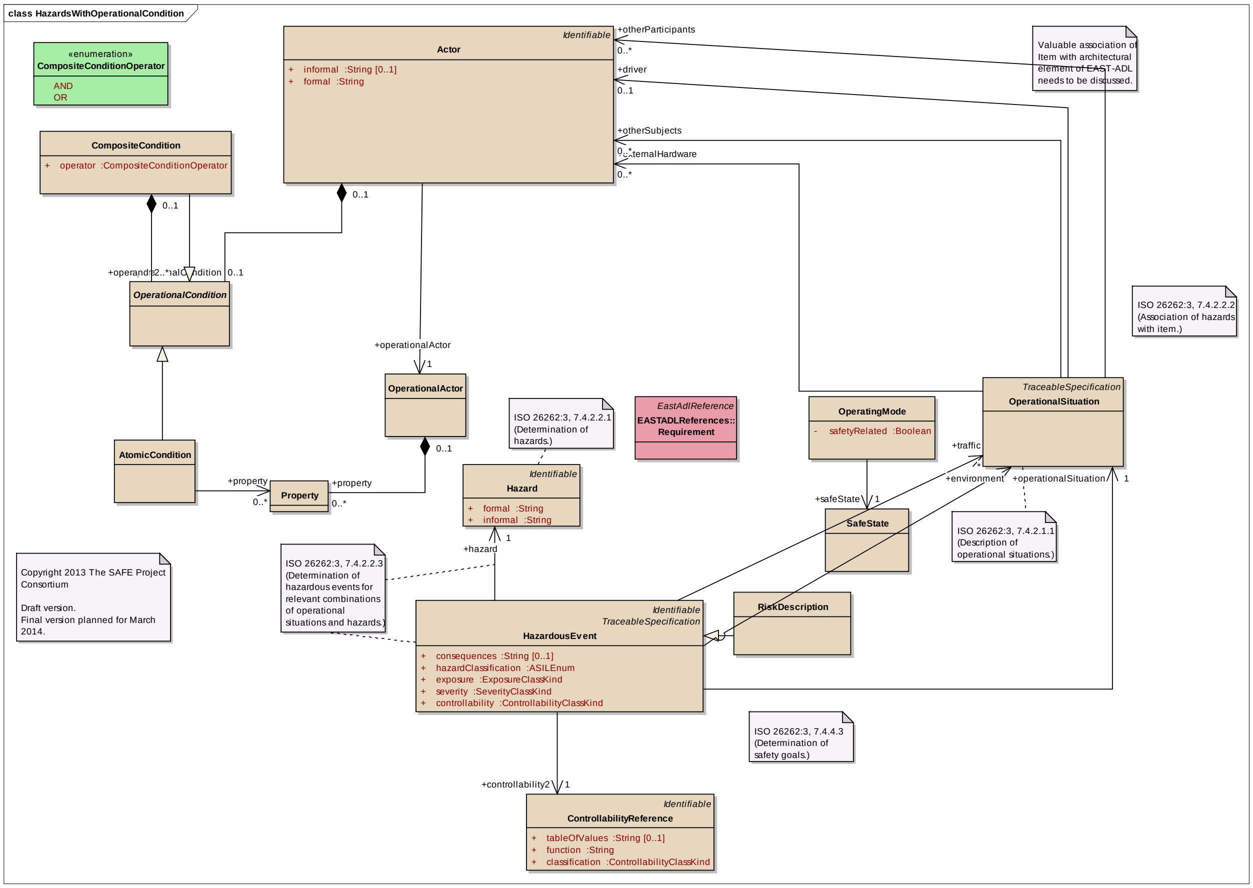


Figure: 41

Element "Actor"

Parent Package: Hazards

Stereotype: ,

Notes:

The “Actor” class is an abstract class which provides the structure that is needed for the contributions of different actors to the operational situation and the hazard.

Note:

The risk assessment of hazardous events focuses on the harm to each person potentially at risk – including the driver or the passengers of the vehicle causing the hazardous event, and other persons potentially at risk such as cyclists, pedestrians or occupants of other vehicles.

Reference:

ISO26262-3-7.4.3.2

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	informal	String			0	0	0		Provides the possibility to capture the informal description of the contribution of a particular actor
	formal	String			0	0	0		Provides the possibility to capture the formal description of the contribution of a particular actor

Relationships

Name	Source/Target	Notes
	Source: OperationalSituation. **Target:** **0..*** Actor.otherSubjects	
	Source: Actor. **Target:** Identifiable.	
	Source: OperationalSituation. **Target:** **0..1** Actor.driver	
	Source: OperationalSituation. **Target:** **0..*** Actor.externalHardware	
	Source: **0..1** OperationalCondition.operationalCondition **Target:** **0..1** Actor.	
	Source: **0..*** Actor.actor **Target:** HazardandRiskSafetyExtension.	
	Source: OperationalSituation. **Target:** **0..*** Actor.otherParticipants	
	Source: Actor. **Target:** **1** OperationalActor.operationalActor	

Element "AtomicCondition"

Parent Package: Hazards

Stereotype: ,

Notes:

Relationships

Name	Source/Target			Notes
	Source:		AtomicCondition.	
	Target:		OperationalCondition.	
	Source:		AtomicCondition.	
	Target:	**0..***	Property.property	

Element "CompositeCondition"

Parent Package: Hazards

Stereotype: ,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	operator	CompositeConditionOperator			0	0	0		

Relationships

Name	Source/Target			Notes
	Source:	**2..***	OperationalCondition.operand	
	Target:	**0..1**	CompositeCondition.	
	Source:		CompositeCondition.	
	Target:		OperationalCondition.	

Element "CompositeConditionOperator"

Parent Package: Hazards

Stereotype: «enumeration»,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	AND				0	0	0		
	OR				0	0	0		

Element "ControllabilityClassKind"

Parent Package: Hazards

Stereotype: «enumeration»,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	C0				0	0	0		
	C1				0	0	0		
	C3				0	0	0		
	C4				0	0	0		

Element "ControllabilityReference"

Parent Package: Hazards

Stereotype: ,

Notes:

The class "ControllabilityReference" is introduced to provide the possibility to capture diagrams. These diagrams are based on road tests and enable a determination of the controllability parameter of the hazardous event.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	tableOfValues	String			0	0	0		Provides the possibility to capture the diagrams in form of tables of values.
	function	String			0	0	0		Provides the possibility to capture the diagram in form of functions.
	classification	ControllabilityCl							

		assKind							

Relationships

Name	Source/Target	Notes
	Source: HazardousEvent. **Target:** **1** ControllabilityReference.controllability2	
	Source: ControllabilityReference. **Target:** Identifiable.	
	Source: **0..*** ControllabilityReference.controllability **Target:** HazardandRiskSafetyExtension.	

Element "ExposureClassKind"

Parent Package: Hazards

Stereotype: «enumeration»,

Notes:

The number of vehicles equipped with the item shall not be considered when estimating the probability of exposure.

Reference:

ISO 26262-3-7.4.3.5

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	E1								
	E2								
	E3								
	E4								

Element "Hazard"

Parent Package: Hazards

Stereotype: ,

Notes:

A hazard describes a potential source of harm. Important is that it is formulated in terms of behavior that can be observed on vehicle level.

Hazards shall be defined in terms of the conditions or behavior that can be observed at the vehicle level.

The hazards shall be determined systematically by using adequate techniques, such as brainstorming, checklists, quality history, FMEA and field studies.

Reference:

ISO 26262-3-7.4.2.2.2

ISO 26262-3-7.4.2.2.1

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	formal	String			0	0	0		Provides the possibility to capture the formal description of the hazard.
	informal	String			0	0	0		Provides the possibility to capture the informal description of the hazard.

Relationships

Name	Source/Target	Notes
	Source: Hazard. **Target:** **1** MalfunctionPrototype.malfunction	
	Source: **0..*** Hazard.hazard **Target:** **0..1** HazardandRiskSafetyExtension.	
	Source: HazardousEvent. **Target:** **1** Hazard.hazard	
	Source: Hazard. **Target:** Identifiable.	
	Source: <anonymous>. **Target:** Hazard.	

Element "HazardousEvent"

Parent Package: Hazards

Stereotype: ,

Notes:

The hazardous event describes a relevant outcome of combinations of a hazard and an operational situation.

An ASIL shall be determined for each hazardous event using the parameters "severity", "probability of exposure" and "controllability" in accordance with Table 4.

Semantic:

The ASIL shall be calculated automatically by using the information given in ISO26262-3-Table.4

Reference:

ISO 26262-3-7.4.2.2.3

ISO 26262-3-7.4.4.1

ISO 26262-3-Table.4

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	consequences	String			0	0	0		Provides the possibility to capture the consequences of a hazardous event. *Reference:* ISO 26262-3-7.4.2.2.4 (The consequences of hazardous events shall be identified.)
	hazardClassification	ASILEnum							Based on Table 4 defined in ISO26262-3 the ASIL shall be allocated to the hazardous event based on the classification of the attributes • Controllability • Severity • Exposure
	exposure	ExposureClassK							**Exposure** is defined as the state of being

		ind							in an **operational situation** that can be **hazardous** if coincident with the **failure mode** under analysis *Reference:* ISO 26262-1-1.37 ISO 26262-1-1.83 ISO 26262-1-1.57 ISO 26262-1-1.40
	severity	SeverityClassKind							**Severity** is defined as the estimation of the extent of **harm** to one or more individuals that can occur in a potentially **hazardous** situation *Reference:* ISO 26262-1-1.120 ISO 26262-1-1.56 ISO 26262-1-1.57
	controllability	ControllabilityClassKind							**Controllability** is defined as the ability to avoid a specified **harm** or damage through the timely reactions of the persons involved, possibly with support from **external measures** *NOTE:* 1. Persons involved can include the driver, passengers or persons in the vicinity of the vehicle's

									exterior. 2. The parameter C **in hazard analysis and risk assessment** represents the potential for controllability. *Reference:* ISO 26262-1.19 ISO 26262-1.56 ISO 26262-1.38 ISO 26262-1.58

Relationships

Name	Source/Target	Notes
	Source: HazardousEvent. **Target:** **1..*** FunctionalSafetyRequirement.safetyGoal	
	Source: HazardousEvent. **Target:** Identifiable.	
	Source: <anonymous>. **Target:** HazardousEvent.	
	Source: HazardousEvent. **Target:** ***** OperationalSituation.environment	
	Source: HazardousEvent. **Target:** **1** OperationalSituation.operationalSituation	
	Source: HazardousEvent. **Target:** ***** OperationalSituation.traffic	
	Source: HazardousEvent. **Target:** **1** ControllabilityReference.controllability2	
	Source: RiskDescription. **Target:** HazardousEvent.	
	Source: HazardousEvent. **Target:** TraceableSpecification.	
	Source: HazardousEvent. **Target:** **1** Hazard.hazard	

Element "OperatingMode"

Parent Package: Hazards

Stereotype: ,

Notes:

The Operating Mode is, according to ISO 26262, a “perceivable functional state of an item or element”. Therefore, it is associated with the item. Moreover, it is associated with the risk description since it describes a state of the item.

Reference:

ISO26262-1-1-81

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	safetyRelated	Boolean							

Relationships

Name	Source/Target	Notes
	Source: OperatingMode. **Target:** 1 SafeState.safeState	

Element "OperationalActor"

Parent Package: Hazards

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: Actor. **Target:** 1 OperationalActor.operationalActor	
	Source: 0..* Property.property **Target:** 0..1 OperationalActor.	

Element "OperationalCondition"

Parent Package: Hazards

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: AtomicCondition. **Target:** OperationalCondition.	
	Source: **2..*** OperationalCondition.operand **Target:** **0..1** CompositeCondition.	
	Source: CompositeCondition. **Target:** OperationalCondition.	
	Source: **0..1** OperationalCondition.operationalCondition **Target:** **0..1** Actor.	

Element "OperationalSituation"

Parent Package: Hazards

Stereotype: ,

Notes:

An operational situation is a scenario that may occur during a vehicles lifetime. Operational situations are formed by contributions of different actors, namely the driver (input of the driver via steering wheel, gas pedal, etc), the environment (e.g. road and lighting conditions), and other participants (pedestrians, other vehicles, etc).

The operational situations and operating modes in which an item's malfunctioning behavior will result in a hazardous event shall be described, both for cases when the vehicle is correctly used and when it is incorrectly used in a foreseeable way

Reference:

ISO 26262-3-7.4.2.1.1

ISO 26262-3-Annex.B-TableB.3.

Relationships

Name	Source/Target	Notes
	Source: OperationalSituation.	

Name	Source/Target	Notes
	Target: **0..*** Actor.otherSubjects	
	Source: OperationalSituation. **Target:** **0..1** Environment.environment	
	Source: HazardousEvent. **Target:** ***** OperationalSituation.environment	
	Source: HazardousEvent. **Target:** **1** OperationalSituation.operationalSituation	
	Source: HazardousEvent. **Target:** ***** OperationalSituation.traffic	
	Source: <anonymous>. **Target:** OperationalSituation.	
	Source: OperationalSituation. **Target:** **0..1** Actor.driver	
	Source: OperationalSituation. **Target:** TraceableSpecification.	
	Source: OperationalSituation. **Target:** **0..*** Actor.externalHardware	
	Source: OperationalSituation. **Target:** **0..*** Actor.otherParticipants	

Element "Property"

Parent Package: Hazards

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: AtomicCondition. **Target:** **0..*** Property.property	
	Source: **0..*** Property.property **Target:** **0..1** OperationalActor.	

Element "RiskDescription"

Parent Package: Hazards

Stereotype: ,

Notes:

The risk description is the counterpart formulated on item level to the hazardous event which is formulated on vehicle level. It describes the endangerment in terms that can be observed at the item boundary in combination with the operational situation.

Relationships

Name	Source/Target	Notes
	Source: RiskDescription. **Target:** HazardousEvent.	
	Source: **0..*** RiskDescription.risk **Target:** **0..1** HazardandRiskSafetyExtension.	

Element "SafeState"

Parent Package: Hazards

Stereotype: ,

Notes:

The safe state is defined as an operating mode of an item without an unreasonable level of risk.

The safe state shall be reached within the allocated FaultTolerantTimeInterval.

In case that the safe state is defiened for a emergency operation it shall be reached within the EmergencyOperationTimeInterval.

Reference:

ISO 26262-1-1.102

Relationships

Name	Source/Target	Notes
	Source: OperatingMode. **Target:** **1** SafeState.safeState	

Element "SeverityClassKind"

Parent Package: Hazards

Stereotype: «enumeration»,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	S0				0	0	0		
	S1				0	0	0		
	S2				0	0	0		
	S3				0	0	0		

Package "Requirements"

Type of Package: **Package**

Parent Package: SAFE Meta-Model

Notes:

Diagram "**SafetyRequirements**"

Notes:

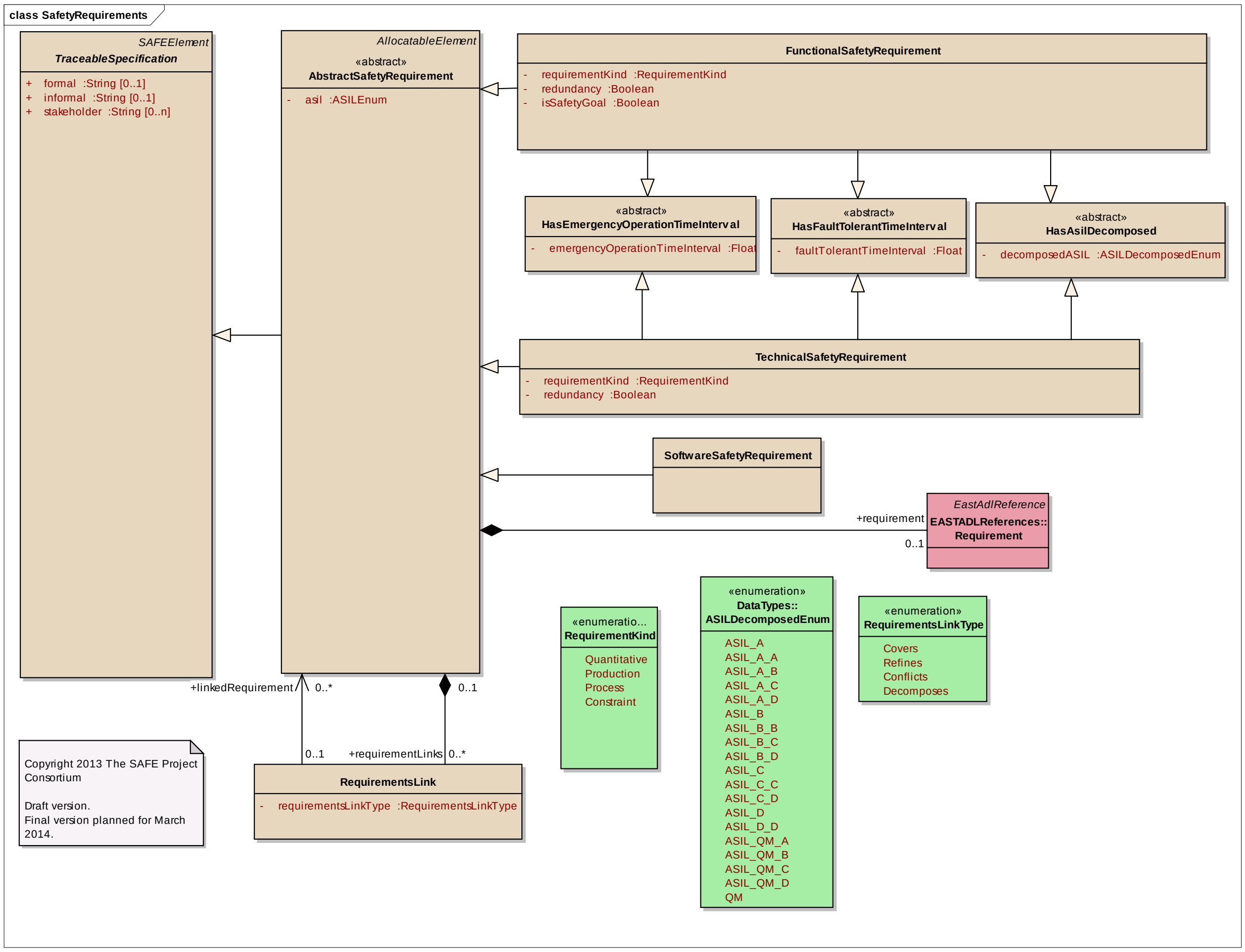


Figure: 42

Diagram "SatisfySafetyRequirements"

Notes:

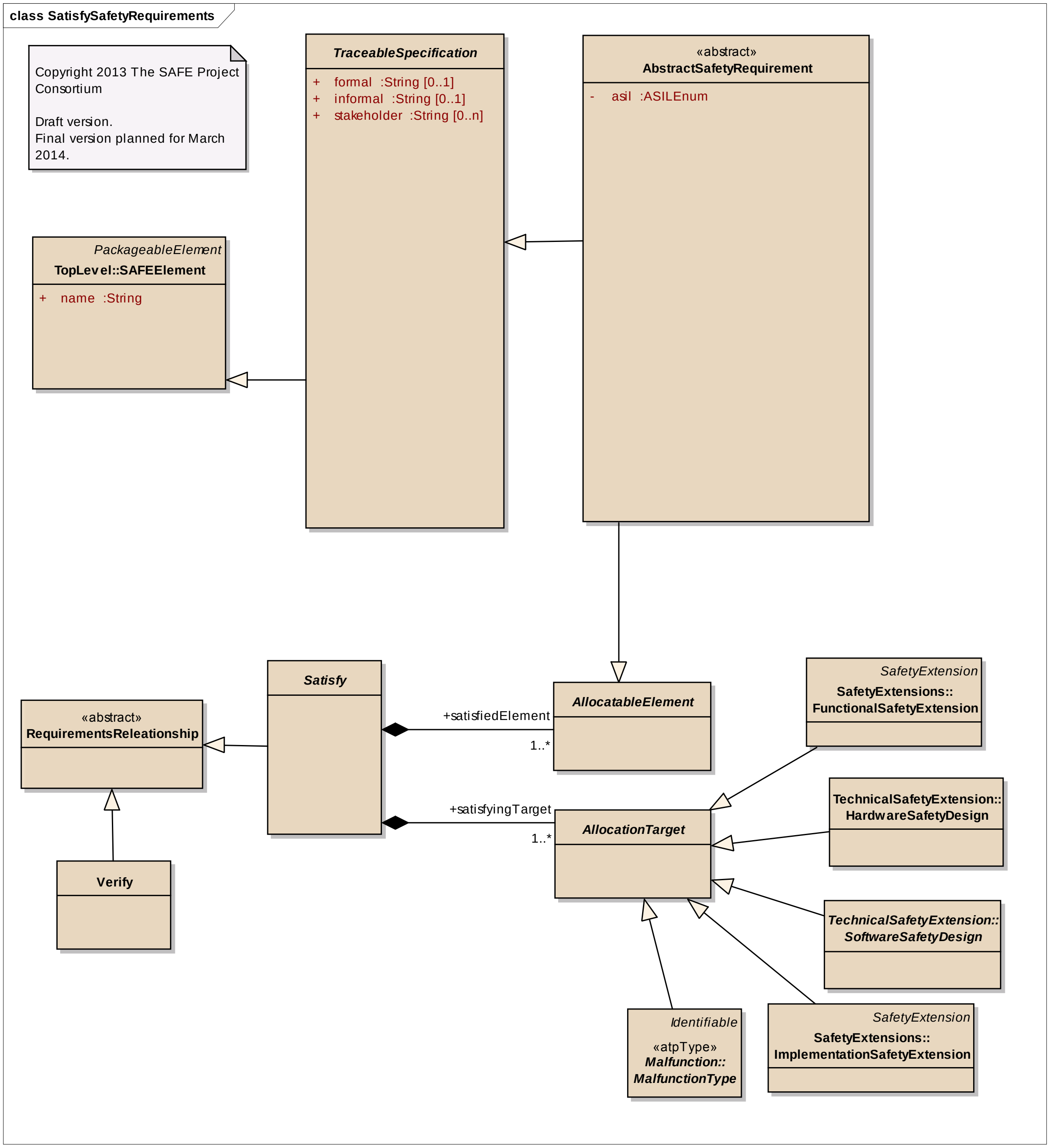


Figure: 43

Element "AbstractSafetyRequirement"

Parent Package: Requirements

Stereotype: «abstract»,

Notes:

In the SAFE meta model AbstratcSafetyRequirement is used as the as the abstract superclass for

- SafetyGoals,
- FunctionalSafetyRequirements and
- TechnicalSafetyRequirements.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	asil	ASILEnum			0	0	0		ASIL which has been assigned via decomposition

Relationships

Name	Source/Target	Notes
	Source: SoftwareSafetyRequirement. **Target:** AbstractSafetyRequirement.	
	Source: **0..1** RequirementsLink. **Target:** **0..*** AbstractSafetyRequirement.linkedRequirement	
	Source: FunctionalSafetyRequirement. **Target:** AbstractSafetyRequirement.	
	Source: TechnicalSafetyRequirement. **Target:** AbstractSafetyRequirement.	
	Source: **0..*** AbstractSafetyRequirement.safetyRequirement **Target:** RequirementsSafetyExtension.	
	Source: AbstractSafetyRequirement. **Target:** TraceableSpecification.	
	Source: AbstractSafetyRequirement. **Target:** AllocatableElement.	
	Source: AbstractSafetyRequirement. **Target:** **0..1** Requirement.requirement	
	Source: **0..*** RequirementsLink.requirementLinks **Target:** **0..1** AbstractSafetyRequirement.	

Element "AllocatableElement"

Parent Package: Requirements

Stereotype: ,

Notes:

Elements which can be allocated.

Relationships

Name	Source/Target	Notes
	Source: 1..* AllocatableElement.satisfiedElement **Target:** Satisfy.	
	Source: AbstractSafetyRequirement. **Target:** AllocatableElement.	

Element "AllocationTarget"

Parent Package: Requirements

Stereotype: ,

Notes:

Elements to which AllocatableElements can be allocated.

Relationships

Name	Source/Target	Notes
	Source: FunctionalSafetyExtension. **Target:** AllocationTarget.	
	Source: HardwareSafetyDesign. **Target:** AllocationTarget.	
	Source: SoftwareSafetyDesign. **Target:** AllocationTarget.	
	Source: MalfunctionType. **Target:** AllocationTarget.	
	Source: 1..* AllocationTarget.satisfyingTarget **Target:** Satisfy.	
	Source: ImplementationSafetyExtension. **Target:** AllocationTarget.	

Element "FunctionalSafetyRequirement"

Parent Package: Requirements

Stereotype: ,

Notes:

FunctionalSafetyRequirements are used to specify

- implementation-independent safety behavior
- implementation-independent safety measures.

FunctionalSafetyRequirements shall contain safety-related attributes.

At least one functional safety requirement shall be specified for each safety goal.

FunctionalSafetyRequirements are allocated to the FunctionComponent of the item.

A safety goal shall be determined for each hazardous event with an ASIL evaluated in the hazard analysis. If similar safety goals are determined, these may be combined into one safety goal.

Constraint:

HasAsilDecomposed shall only be set if attribute redundancy is set true.

Reference:

ISO 26262-1-1.53

ISO 26262-3: 8.4.2.3

ISO 26262-3-7.4.4.3

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	requirementKind	RequirementKind							
	redundancy	Boolean							
	isSafetyGoal	Boolean							

Relationships

Name	Source/Target	Notes
	Source: FunctionalSafetyRequirement. **Target:** HasAsilDecomposed.	
	Source: HazardousEvent. **Target:** **1..*** FunctionalSafetyRequirement.safetyGoal	
	Source: FunctionalSafetyRequirement. **Target:** AbstractSafetyRequirement.	
	Source: FunctionalSafetyRequirement. **Target:** HasEmergencyOperationTimeInterval.	

Name	Source/Target		Notes
	Source:	FunctionalSafetyRequirement.	
	Target:	HasFaultTolerantTimeInterval.	

Element "HasAsilDecomposed"

Parent Package: Requirements

Stereotype: «abstract»,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	decomposedASIL	ASILDecomposedEnum							

Relationships

Name	Source/Target		Notes
	Source:	FunctionalSafetyRequirement.	
	Target:	HasAsilDecomposed.	
	Source:	TechnicalSafetyRequirement.	
	Target:	HasAsilDecomposed.	

Element "HasEmergencyOperationTimeInterval"

Parent Package: Requirements

Stereotype: «abstract»,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	emergencyOperationTimeInterval	Float							

Relationships

Name	Source/Target	Notes

Name	Source/Target		Notes
	Source:	FunctionalSafetyRequirement.	
	Target:	HasEmergencyOperationTimeInterval.	
	Source:	TechnicalSafetyRequirement.	
	Target:	HasEmergencyOperationTimeInterval.	

Element "HasFaultTolerantTimeInterval"

Parent Package: Requirements

Stereotype: «abstract»,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	faultTolerantTime Interval	Float							

Relationships

Name	Source/Target		Notes
	Source:	TechnicalSafetyRequirement.	
	Target:	HasFaultTolerantTimeInterval.	
	Source:	FunctionalSafetyRequirement.	
	Target:	HasFaultTolerantTimeInterval.	

Element "RequirementKind"

Parent Package: Requirements

Stereotype: «enumeration»,

Notes:

This enumeration specifies the kind of safety requirement

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	Quantitative								
	Production								
	Process								

	Constraint								

Element "RequirementsLink"

Parent Package: Requirements

Stereotype: ,

Notes:

RequirementsLink is owned by requirement. It is used to establish links between requirements. The type of the link is determined by the RequirementsLinkType.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	requirementsLinkType	RequirementsLinkType			0	0	0		The type of the link between requirements is determined by the RequirementsLinkType.

Relationships

Name	Source/Target	Notes
	Source: **0..1** RequirementsLink. **Target:** **0..*** AbstractSafetyRequirement.linkedRequirement	
	Source: **0..*** RequirementsLink.requirementLinks **Target:** **0..1** AbstractSafetyRequirement.	

Element "RequirementsLinkType"

Parent Package: Requirements

Stereotype: «enumeration»,

Notes:

RequirementsLinkType defines the type of the link between requirements.

Covers: The requirement which owns a link of this type covers the linked requirements

Refines: The requirement which owns a link of this type refines the linked requirements

Conflicts: The requirement which owns a link of this type conflicts with the linked requirements

Decomposes: The requirement which owns a link of this type decomposes the linked requirements

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes

	Covers				0	0	0		
	Refines				0	0	0		
	Conflicts								
	Decomposes								

Element "RequirementsReleationship"

Parent Package: Requirements

Stereotype: «abstract»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: Satisfy. **Target:** RequirementsReleationship.	
	Source: **0..*** RequirementsReleationship.requirementsRelationship **Target:** RequirementsSafetyExtension.	
	Source: Verify. **Target:** RequirementsReleationship.	

Element "Satisfy"

Parent Package: Requirements

Stereotype: ,

Notes:

Satisfy allows allocating requirements to elements of the preliminary architecture as well as HW-Elements and SW-Elements

Relationships

Name	Source/Target	Notes
	Source: **1..*** AllocatableElement.satisfiedElement **Target:** Satisfy.	
	Source: Satisfy. **Target:** RequirementsReleationship.	

Name	Source/Target	Notes
	Source: 1..* AllocationTarget.satisfyingTarget **Target:** Satisfy.	

Element "SoftwareSafetyRequirement"

Parent Package: Requirements

Stereotype: ,

Notes:

Represents the refinement of the technical safety requirements namely the software safety requirements (SSR). Every SSR is realized in the form of a mechanism (partially automatically generated) which traces back to its originating SSR. The SSR element refers to the TSR (technical safety requirement) refined via its specification.

Relationships

Name	Source/Target	Notes
	Source: SoftwareSafetyRequirement. **Target:** AbstractSafetyRequirement.	
	Source: HeartbeatSender. **Target:** SoftwareSafetyRequirement.	
	Source: ErrorHandlerSSR. **Target:** SoftwareSafetyRequirement.	
	Source: HeartbeatReceiver. **Target:** SoftwareSafetyRequirement.	
	Source: CodeGenerationConfiguration. **Target: 1** SoftwareSafetyRequirement.fromRequirement	The mechanism configured by the SSMConfiguration element
	Source: HealthMonitor. **Target:** SoftwareSafetyRequirement.	
	Source: Voting. **Target:** SoftwareSafetyRequirement.	
	Source: ErrorDetectionSSR. **Target:** SoftwareSafetyRequirement.	

Element "TechnicalSafetyRequirement"

Parent Package: Requirements

Stereotype: ,

Notes:

TechnicalSafetyRequirements are used to specify the implementation of the associated FunctionComponent . They are derived by the allocated FunctionalSafetyRequirements.

TechnicalSafetyRequirements include the specification of safety-related failure mitigation.

The technical safety requirements specification refines the functional safety concept, considering both the functional concept and the preliminary architectural assumptions

The technical safety requirements are allocated to hardware and software, and, if applicable, on other technologies

Reference:

ISO26262-1-1.133

ISO26262-4- 6.4.9

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	requirementKind	RequirementKind							
	redundancy	Boolean							

Relationships

Name	Source/Target	Notes
	Source: TechnicalSafetyRequirement. **Target:** HasFaultTolerantTimeInterval.	
	Source: AutosarSafetyExtension. **Target:** TechnicalSafetyRequirement.	
	Source: ChromosomeSafeExtension. **Target:** TechnicalSafetyRequirement.	
	Source: TechnicalSafetyRequirement. **Target:** AbstractSafetyRequirement.	
	Source: TechnicalSafetyRequirement. **Target:** HasEmergencyOperationTimeInterval.	
	Source: TechnicalSafetyRequirement.	

Name	Source/Target	Notes
	Target: HasAsilDecomposed.	
	Source: CodeGenerationConfiguration.codegenconfiguration **0..*** **Target:** TechnicalSafetyRequirement.	

Element "TraceableSpecification"

Parent Package: Requirements

Stereotype: ,

Notes:

Abstract superclass for elements which can be traces (e.g. Requirements)

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	formal	String			0	0	0		
	informal	String			0	0	0		Informal description of the element
	stakeholder	String			0	0	0		Stakeholders for the element

Relationships

Name	Source/Target	Notes
	Source: TraceableSpecification. **Target:** SAFEElement.	
	Source: ErrorModelType. **Target:** TraceableSpecification.	
	Source: AbstractSafetyRequirement. **Target:** TraceableSpecification.	
	Source: OperationalSituation. **Target:** TraceableSpecification.	
	Source: HazardousEvent. **Target:** TraceableSpecification.	

Element "Verify"

Parent Package: Requirements

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: Verify. **Target:** RequirementsReleationship.	

Package "SoftwareSafetyRequirements"

Type of Package: **Package**

Parent Package: Requirements

Notes:

Diagram "SoftwareSafetyRequirements"

Notes:

Abstract structure for the specification of software safety requirements in a SAFE model. The software safety requirements trace back to the originating technical safety requirements and can be linked to realizations via the EAST-ADL Satisfy mechanism.

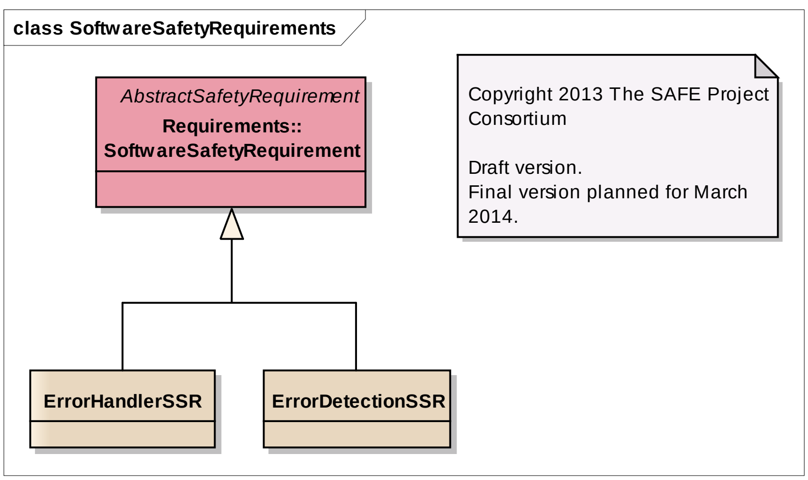


Figure: 44

Element "ErrorDetectionSSR"

Parent Package: SoftwareSafetyRequirements

Stereotype: ,

Notes:

Detection software safety requirements (SSR) define the requirements on detection software safety mechanisms (SSM) in order to detect errors which might lead to violation of safety goals. These mechanisms specify for each given error it is able to detect a reaction to be taken. This reaction can be considered as the handling of the error. Starting from an detection SSR it is possible to build the error reaction strategy modeled by the engineer.

Relationships

Name	Source/Target	Notes
	Source: CpuSelfTest. **Target:** ErrorDetectionSSR.	
	Source: ActuatorMonitor. **Target:** ErrorDetectionSSR.	
	Source: Heartbeat. **Target:** ErrorDetectionSSR.	
	Source: GradientCheck. **Target:** ErrorDetectionSSR.	
	Source: MemorySelfTest. **Target:** ErrorDetectionSSR.	
	Source: CRC. **Target:** ErrorDetectionSSR.	
	Source: Comparison. **Target:** ErrorDetectionSSR.	
	Source: ErrorDetectionSSR. **Target:** SoftwareSafetyRequirement.	
	Source: ContextRangeCheck. **Target:** ErrorDetectionSSR.	
	Source: AlivenessMonitor. **Target:** ErrorDetectionSSR.	

Element "ErrorHandlerSSR"

Parent Package: SoftwareSafetyRequirements

Stereotype: ,

Notes:

This element serves as the base for all handler SSR.

Relationships

Name	Source/Target	Notes
	Source: AlivenessMonitor. **Target:** ***** ErrorHandlerSSR.tooEarly	
	Source: GradientCheck. **Target:** **0..*** ErrorHandlerSSR.gradientTooHigh	The error handling SSR specification in case the gradient is too high
	Source: **1** ErrorCondition. **Target:** **1** ErrorHandlerSSR.onTrigger	Error handling: executed when triggered by condition in Health Monitor.
	Source: Comparison. **Target:** ***** ErrorHandlerSSR.comparisonFalse	The error handling SSR when a comparison results in a false value.
	Source: AlivenessMonitor. **Target:** ***** ErrorHandlerSSR.tooLate	
	Source: GradientCheck. **Target:** **0..*** ErrorHandlerSSR.gradientTooLow	The error handling SSR specification in case the gradient is too low
	Source: ActuatorMonitor. **Target:** ***** ErrorHandlerSSR.actuatorError	The error handling SSR specification in case an actuator monitor error occurs.
	Source: Filter. **Target:** ErrorHandlerSSR.	
	Source: CRC. **Target:** ***** ErrorHandlerSSR.crcFailed	The error handling SSR specification in case a CRC error occurs.
	Source: ChromosomeErrorHandler. **Target:** ErrorHandlerSSR.	
	Source: ContextRangeCheck.	

Name	Source/Target	Notes
	Target: ***** ErrorHandlerSSR.belowRange	
	Source: Comparison. **Target:** ***** ErrorHandlerSSR.comparisonTrue	The error handling SSR when a comparison results in a true value.
	Source: ErrorHandlerSSR. **Target:** SoftwareSafetyRequirement.	
	Source: CHROMOSOMEHealthMonitorNotification. **Target:** ErrorHandlerSSR.	
	Source: **1** MemorySelfTest. **Target:** **1** ErrorHandlerSSR.testFailed	The error handling SSR specification in case a RAM error occurs during the test execution.
	Source: **1** HeartbeatReceiver. **Target:** **1** ErrorHandlerSSR.deadlineMissed	The error handling SSR specification in case a deadline has been missed and no input received at receiver side.
	Source: Filter. **Target:** **1..*** ErrorHandlerSSR.filterError	The error handling SSR specification in case a filtering error occurs
	Source: ContextRangeCheck. **Target:** ***** ErrorHandlerSSR.aboveRange	
	Source: AlivenessMonitor. **Target:** ***** ErrorHandlerSSR.tooOften	
	Source: **1** Voting. **Target:** **1** ErrorHandlerSSR.noConsensus	The error handling SSR specification in case one (or more) values demonstrate mismatch, and the consensus could

Name	Source/Target	Notes
		not be found.
	Source: **1** Voting. **Target:** **1** ErrorHandlerSSR.valueMismatch	The error handling SSR specification in case one (or more) values demonstrate large mismatch, but the consensus still can be found.
	Source: **1** CpuSelfTest. **Target:** **1** ErrorHandlerSSR.testFailed	The error handling SSR specification in case a CPU error occurs during the test execution.

Package "AlivenessMonitor"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

This package groups the elements related to the software safety mechanism Aliveness Monitor

Diagram "AlivenessMonitor"

Notes:

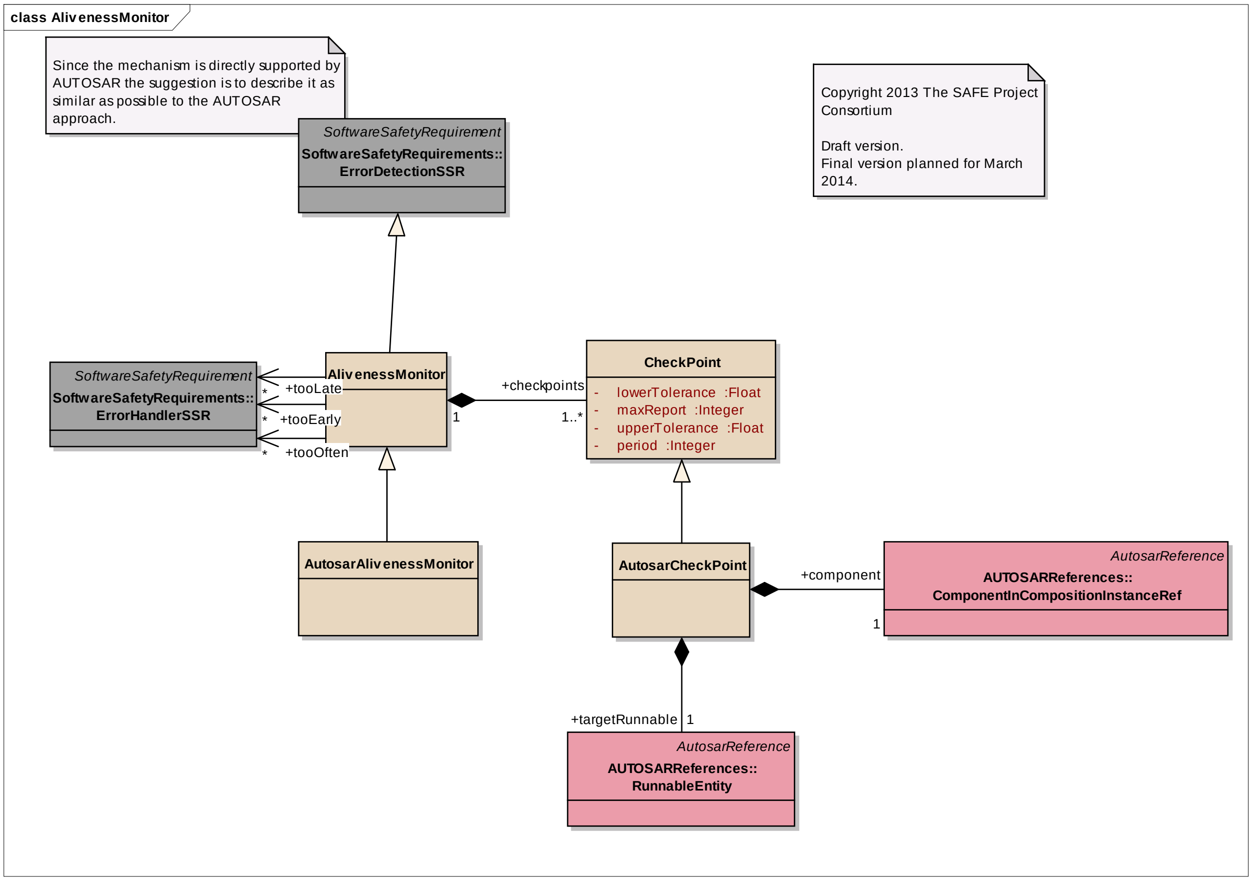


Figure: 45

Element "AlivenessMonitor"

Parent Package: AlivenessMonitor

Stereotype: ,

Notes:

The aliveness monitor contains the set of mapped checkpoint definitions to executable entities. The set of checkpoints contained in the mechanism defines the scope of a specific mechanism instance.

Relationships

Name	Source/Target	Notes
	Source: AlivenessMonitor. **Target:** * ErrorHandlerSSR.tooEarly	
	Source: AlivenessMonitor. **Target:** * ErrorHandlerSSR.tooLate	
	Source: AutosarAlivenessMonitor.	

Name	Source/Target	Notes
	Target: AlivenessMonitor.	
	Source: **1..*** CheckPoint.checkpoints **Target:** **1** AlivenessMonitor.	The set of checkpoints used to monitor aliveness of a runnable component.
	Source: AlivenessMonitor. **Target:** ***** ErrorHandlerSSR.tooOften	
	Source: AlivenessMonitor. **Target:** ErrorDetectionSSR.	

Element "AutosarAlivenessMonitor"

Parent Package: AlivenessMonitor

Stereotype: ,

Notes:

AUTOSAR specific aliveness monitor meta class

Relationships

Name	Source/Target	Notes
	Source: AutosarAlivenessMonitor. **Target:** AlivenessMonitor.	

Element "AutosarCheckPoint"

Parent Package: AlivenessMonitor

Stereotype: ,

Notes:

AUTOSAR specific checkpoint

Relationships

Name	Source/Target	Notes
	Source: **1** RunnableEntity.targetRunnable **Target:** AutosarCheckPoint.	The target runnable entity to be observed
	Source: **1** ComponentInCompositionInstanceRef.component	The target component

Name	Source/Target	Notes
	Target: AutosarCheckPoint.	containing the internal behavior whose runnable entity is monitored
	Source: AutosarCheckPoint. **Target:** CheckPoint.	

Element "CheckPoint"

Parent Package: AlivenessMonitor

Stereotype: ,

Notes:

A checkpoint provides the attributes necessary for checking if a given executable unity is being triggered correctly by the system.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	lowerTolerance	Float			0	0	0		
	maxReport	Integer			0	0	0		
	upperTolerance	Float			0	0	0		
	period	Integer							

Relationships

Name	Source/Target	Notes
	Source: AutosarCheckPoint. **Target:** CheckPoint.	
	Source: **1..*** CheckPoint.checkpoints **Target:** **1** AlivenessMonitor.	The set of checkpoints used to monitor aliveness of a runnable component.

Package "ActuatorMonitor"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

This package groups the elements related to the software safety mechanism Actuator Monitor

Diagram "ActuatorMonitor"

Notes:

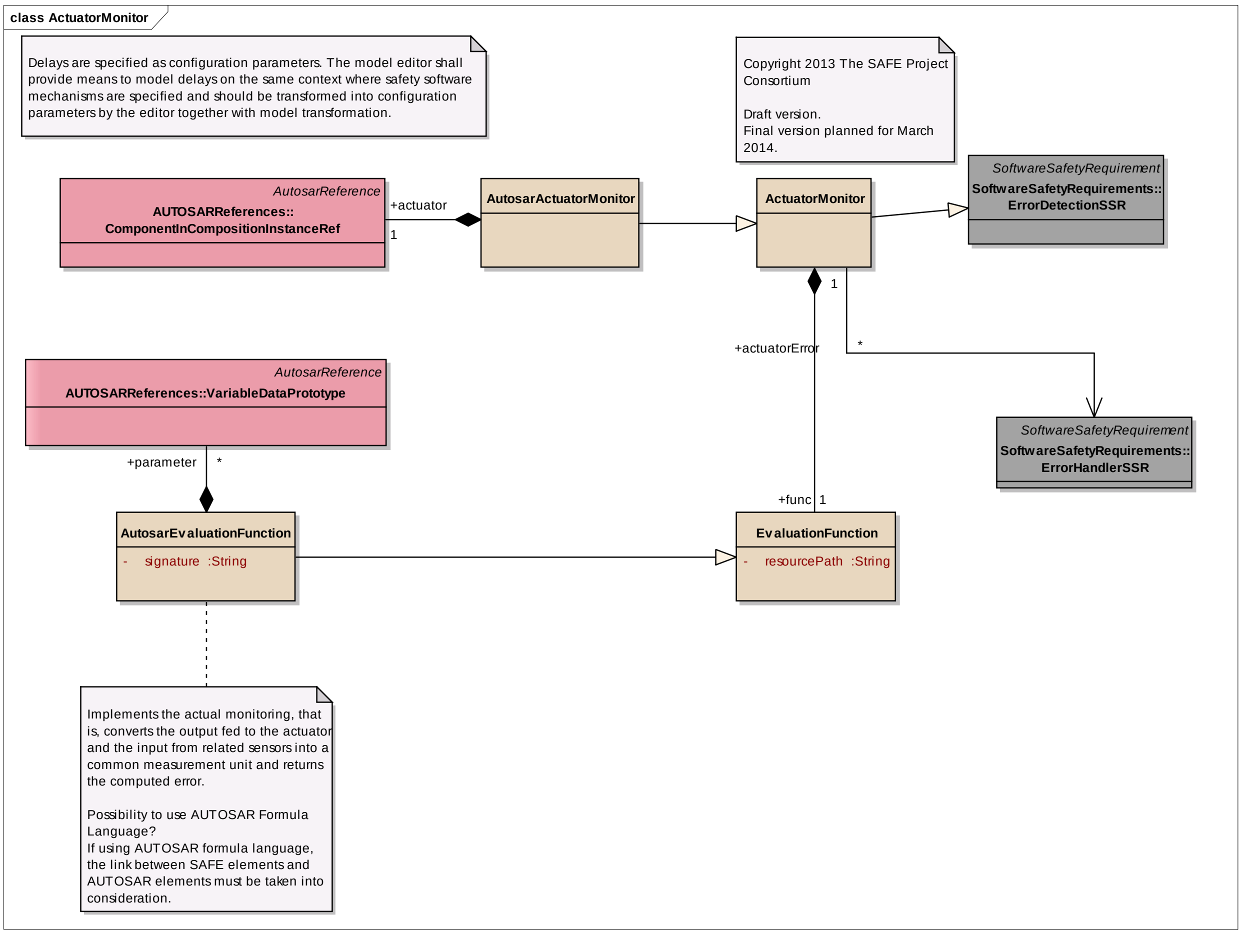


Figure: 46

Element "ActuatorMonitor"

Parent Package: ActuatorMonitor

Stereotype: ,

Notes:

This meta-class represents the actuator monitor requirement. It extends the meta-class SSR from SAFE meta model.

To define an actuator monitor requirement the engineer has to reference the monitored actuator through the ActuatorMonitor attribute actuator and the inputs through which the monitoring occurs, using the attribute sensors. Furthermore, the engineer has to specify an evaluation function which maps input from sensors to the provided actuator output.

Relationships

Name	Source/Target	Notes
	Source: ActuatorMonitor. **Target:** * ErrorHandlerSSR.actuatorError	The error handling SSR specification in case an actuator monitor error occurs.
	Source: ActuatorMonitor. **Target:** ErrorDetectionSSR.	
	Source: AutosarActuatorMonitor. **Target:** ActuatorMonitor.	
	Source: 1 EvaluationFunction.func **Target:** 1 ActuatorMonitor.	The function used to evaluate the actuator behavior.

Element "AutosarActuatorMonitor"

Parent Package: ActuatorMonitor

Stereotype: ,

Notes:

Specific meta class for the AUTOSAR actuator monitor software safety requirement.

Relationships

Name	Source/Target	Notes
	Source: 1 ComponentInCompositionInstanceRef.actuator **Target:** AutosarActuatorMonitor.	The instance reference to the monitored actuator component in AUTOSAR
	Source: AutosarActuatorMonitor. **Target:** ActuatorMonitor.	

Element "AutosarEvaluationFunction"

Parent Package: ActuatorMonitor

Stereotype: ,

Notes:

Evaluation function defined by the engineer to relate the input from sensors to the output value provided by the actuator controller.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	signature	String							

Relationships

Name	Source/Target	Notes
	Source: <anonymous>. **Target:** AutosarEvaluationFunction.	
	Source: AutosarEvaluationFunction. **Target:** EvaluationFunction.	
	Source: * VariableDataPrototype.parameter **Target:** AutosarEvaluationFunction.	

Element "EvaluationFunction"

Parent Package: ActuatorMonitor

Stereotype: ,

Notes:

Abstract element representing an user defined evaluation function for deciding if the actuator behaves correctly.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	resourcePath	String							

Relationships

Name	Source/Target	Notes
	Source: AutosarEvaluationFunction. **Target:** EvaluationFunction.	
	Source: 1 EvaluationFunction.func **Target:** 1 ActuatorMonitor.	The function used to evaluate the actuator behavior.

Package "CRCChecksum"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

This package groups the elements related to the software safety mechanism CRC Checksum

Diagram "CRCChecksum"

Notes:

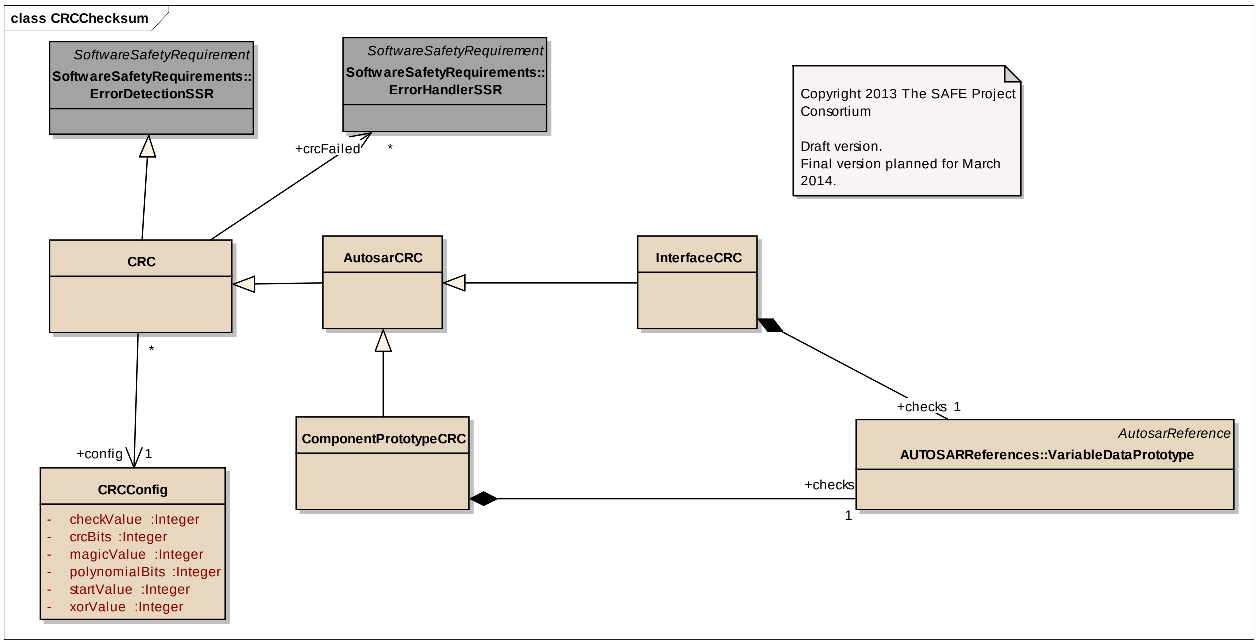


Figure: 47

Element "AutosarCRC"

Parent Package: CRCChecksum

Stereotype: ,

Notes:

AUTOSAR specific CRC requirement.

Relationships

Name	Source/Target	Notes
	Source: InterfaceCRC. **Target:** AutosarCRC.	
	Source: ComponentPrototypeCRC. **Target:** AutosarCRC.	

Name	Source/Target	Notes
	Source: AutosarCRC. **Target:** CRC.	

Element "CRC"

Parent Package: CRCChecksum

Stereotype: ,

Notes:

This meta-class represents the requirement cyclic-redundancy-check.

Relationships

Name	Source/Target	Notes
	Source: CRC. **Target:** * ErrorHandlerSSR.crcFailed	The error handling SSR specification in case a CRC error occurs.
	Source: AutosarCRC. **Target:** CRC.	
	Source: * CRC. **Target:** 1 CRCConfig.config	The configuration information for the CRC mechanism.
	Source: CRC. **Target:** ErrorDetectionSSR.	

Element "CRCConfig"

Parent Package: CRCChecksum

Stereotype: ,

Notes:

This meta-class provides the means for configuring the CRC requirement. The attributes are used by both specializations of the abstract version of the SSR.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	checkValue	Integer			0	0	0		
	crcBits	Integer			0	0	0		

	magicValue	Integer			0	0	0		
	polynomialBits	Integer			0	0	0		
	startValue	Integer			0	0	0		
	xorValue	Integer			0	0	0		

Relationships

Name	Source/Target	Notes
	Source: * CRC. **Target:** 1 CRCConfig.config	The configuration information for the CRC mechanism.

Element "ComponentPrototypeCRC"

Parent Package: CRCChecksum

Stereotype: ,

Notes:

This meta-class is the specialization of the abstract SSR CRC which is deployed for a specific instance of a component providing a given interface.

Relationships

Name	Source/Target	Notes
	Source: 1 VariableDataPrototype.checks **Target:** ComponentPrototypeCRC.	The AUTOSAR variable data prototype of a software component whose value is check using the CRC SSR.
	Source: ComponentPrototypeCRC. **Target:** AutosarCRC.	

Element "InterfaceCRC"

Parent Package: CRCChecksum

Stereotype: ,

Notes:

This meta-class is the specialization of the CRC SSR deployed as the property of data elements of a given interface. This means the CRC is a property of the interface and not of a specific instance (realization) of the interface.

Relationships

Name	Source/Target	Notes
	Source: **1** VariableDataPrototype.checks **Target:** InterfaceCRC.	The AUTOSAR variable data prototype of an interface whose value is check using the CRC SSR.
	Source: InterfaceCRC. **Target:** AutosarCRC.	

Package "Comparison"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

This package groups the elements related to the software safety mechanism comparison

Diagram "Comparison"

Notes:

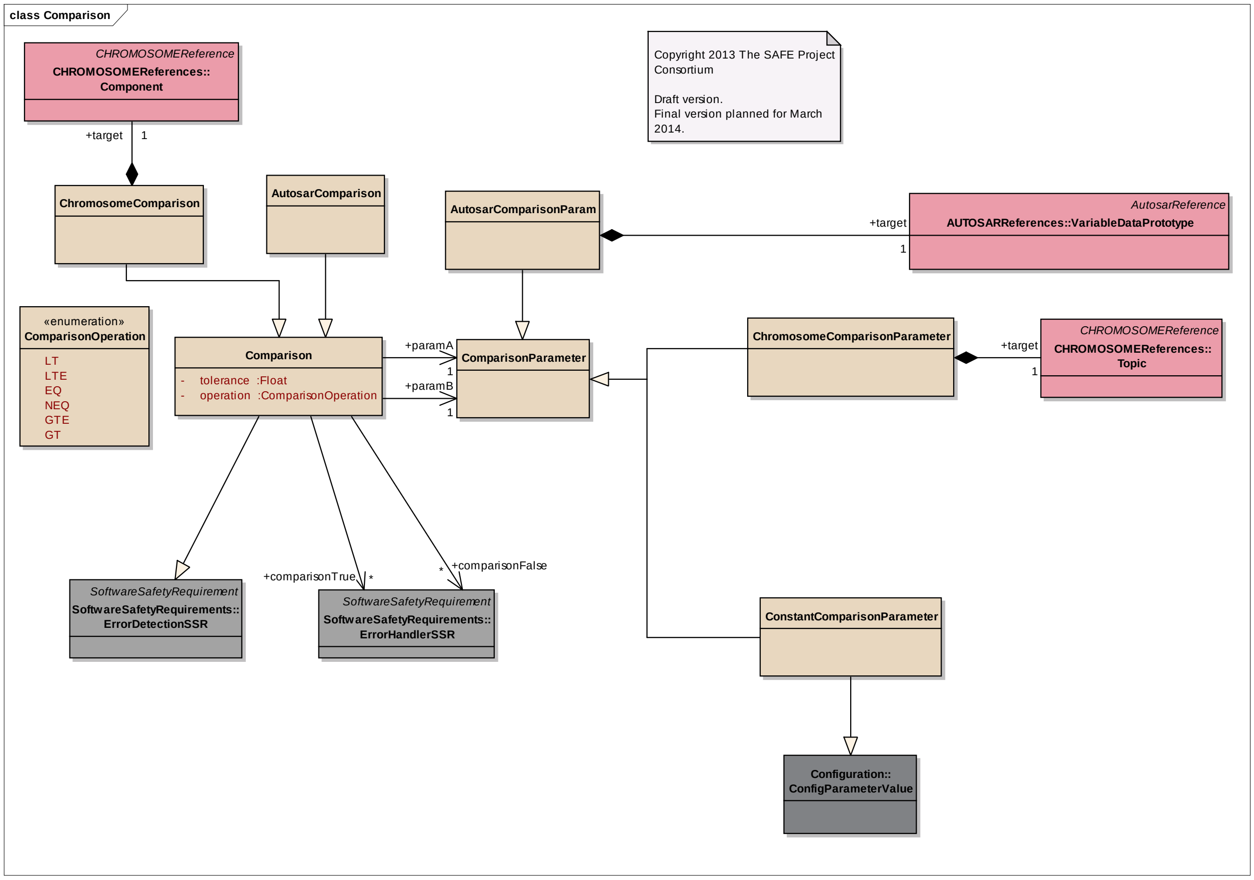


Figure: 48

Element "AutosarComparison"

Parent Package: Comparison

Stereotype: ,

Notes:

AUTOSAR specific comparison SSR. The compared values are provided as AUTOSAR variable instance references and the operation applied to inputs is defined through the Operation enumeration.

Relationships

Name	Source/Target	Notes
	Source: AutosarComparison. **Target:** Comparison.	

Element "AutosarComparisonParam"

Parent Package: Comparison

Stereotype: ,

Notes:

AUTOSAR specific comparison parameters. Allows to reference AUTOSAR elements used in the comparison.

Relationships

Name	Source/Target	Notes
	Source: AutosarComparisonParam. **Target:** ComparisonParameter.	
	Source: 1 VariableDataPrototype.target **Target:** AutosarComparisonParam.	The target variable data prototype of an AUTOSAR software component which provides a value for the comparison operation.

Element "ChromosomeComparison"

Parent Package: Comparison

Stereotype: ,

Notes:

Chromosome comparison is referencing a specific Chromosome component instance, which is implementing comparison of one or multiple topics.

Relationships

Name	Source/Target	Notes
	Source: ChromosomeComparison. **Target:** Comparison.	
	Source: 1 Component.target **Target:** ChromosomeComparison.	

Element "ChromosomeComparisonParameter"

Parent Package: Comparison

Stereotype: ,

Notes:

Chromosome specific comparison parameter. Allows to reference a specific Chromosome topic, which acts as a comparison entity.

Relationships

Name	Source/Target	Notes
	Source: ChromosomeComparisonParameter. **Target:** ComparisonParameter.	
	Source: **1** Topic.target **Target:** ChromosomeComparisonParameter.	

Element "Comparison"

Parent Package: Comparison

Stereotype: ,

Notes:

Value comparison SSR.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	tolerance	Float			0	0	0		
	operation	ComparisonOperation							

Relationships

Name	Source/Target	Notes
	Source: Comparison. **Target:** ***** ErrorHandlerSSR.comparisonFalse	The error handling SSR when a comparison results in a false value.
	Source: Comparison. **Target:** ***** ErrorHandlerSSR.comparisonTrue	The error handling SSR when a comparison results in a true value.
	Source: AutosarComparison. **Target:** Comparison.	
	Source: Comparison.	The second parameter for the

Name	Source/Target	Notes
	Target: **1** ComparisonParameter.paramB	comparison.
	Source: Comparison. **Target:** **1** ComparisonParameter.paramA	The first parameter for the comparison.
	Source: ChromosomeComparison. **Target:** Comparison.	
	Source: Comparison. **Target:** ErrorDetectionSSR.	

Element "ComparisonOperation"

Parent Package: Comparison

Stereotype: «enumeration»,

Notes:

Enumeration providing the valid operations for a comparison SSR.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	LT				0	0	0		
	LTE				0	0	0		
	EQ				0	0	0		
	NEQ				0	0	0		
	GTE				0	0	0		
	GT				0	0	0		

Element "ComparisonParameter"

Parent Package: Comparison

Stereotype: ,

Notes:

Abstract comparison parameter element.

Relationships

Name	Source/Target	Notes
	Source: ChromosomeComparisonParameter. **Target:** ComparisonParameter.	

Name	Source/Target	Notes
	Source: ConstantComparisonParameter. **Target:** ComparisonParameter.	
	Source: Comparison. **Target: 1** ComparisonParameter.paramB	The second parameter for the comparison.
	Source: AutosarComparisonParam. **Target:** ComparisonParameter.	
	Source: Comparison. **Target: 1** ComparisonParameter.paramA	The first parameter for the comparison.

Element "ConstantComparisonParameter"

Parent Package: Comparison

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: ConstantComparisonParameter. **Target:** ConfigParameterValue.	
	Source: ConstantComparisonParameter. **Target:** ComparisonParameter.	

Package "ContextRangeCheck"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

This package groups the elements related to the software safety mechanism Context Range Check

Diagram "ContextRangeCheck"

Notes:

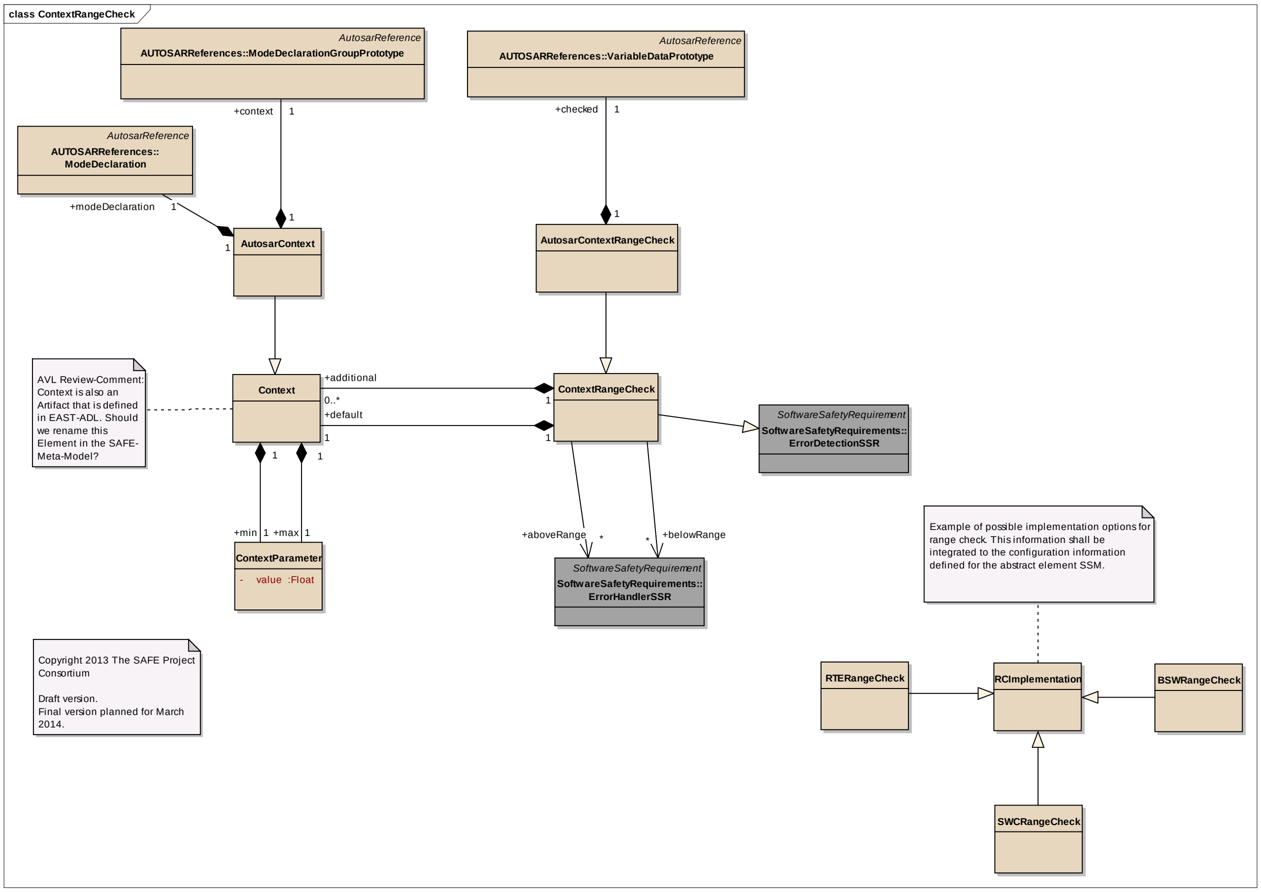


Figure: 49

Element "AutosarContext"

Parent Package: ContextRangeCheck

Stereotype: ,

Notes:

AUTOSAR specific meta class for defining the context used by the range check SSR.

Relationships

Name	Source/Target	Notes
	Source: 1 ModeDeclaration.modeDeclaration **Target: 1** AutosarContext.	The mode used to define a context based range
	Source: 1 ModeDeclarationGroupPrototype.context **Target: 1** AutosarContext.	The instance of a mode group declaration whose modes are used to define a value

Name	Source/Target	Notes
		range context
	Source: AutosarContext. **Target:** Context.	

Element "AutosarContextRangeCheck"

Parent Package: ContextRangeCheck

Stereotype: ,

Notes:

AUTOSAR specific meta class defining the context range check SSR.

Relationships

Name	Source/Target	Notes
	Source: **1** VariableDataPrototype.checked **Target:** **1** AutosarContextRangeCheck.	The variable data prototype of an AUTOSAR software component whose range is monitored
	Source: AutosarContextRangeCheck. **Target:** ContextRangeCheck.	

Element "BSWRangeCheck"

Parent Package: ContextRangeCheck

Stereotype: ,

Notes:

Range check implemented by a BSW element. For example range check configured for a given ADC BSW element.

Relationships

Name	Source/Target	Notes
	Source: BSWRangeCheck. **Target:** RCImplementation.	

Element "Context"

Parent Package: ContextRangeCheck

Stereotype: ,

Notes:

Defines the parameters used by the range check SSR when the system is in a given AUTOSAR mode.

Relationships

Name	Source/Target	Notes
	Source: AutosarContext. **Target:** Context.	
	Source: **1** ContextParameter.max **Target:** **1** Context.	The maximal range for a given context.
	Source: **1** ContextParameter.min **Target:** **1** Context.	The minimal range for a given context.
	Source: **1** Context.default **Target:** **1** ContextRangeCheck.	The default context required for a context range check.
	Source: <anonymous>. **Target:** Context.	
	Source: **0..*** Context.additional **Target:** **1** ContextRangeCheck.	The additional context specifications for a context range check.

Element "ContextParameter"

Parent Package: ContextRangeCheck

Stereotype: ,

Notes:

Provides the valid range parameters for a given context.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	value	Float			0	0	0		

Relationships

Name	Source/Target	Notes
	Source: **1** ContextParameter.max	The maximal

Name	Source/Target	Notes
	Target: **1** Context.	range for a given context.
	Source: **1** ContextParameter.min **Target:** **1** Context.	The minimal range for a given context.

Element "ContextRangeCheck"

Parent Package: ContextRangeCheck

Stereotype: ,

Notes:

This meta-class defines a range check SSR whose range is defined based on the current system/component context (e.g. AUTOSAR mode).

Relationships

Name	Source/Target	Notes
	Source: ContextRangeCheck. **Target:** ***** ErrorHandlerSSR.belowRange	
	Source: **1** Context.default **Target:** **1** ContextRangeCheck.	The default context required for a context range check.
	Source: AutosarContextRangeCheck. **Target:** ContextRangeCheck.	
	Source: ContextRangeCheck. **Target:** ***** ErrorHandlerSSR.aboveRange	
	Source: ContextRangeCheck. **Target:** ErrorDetectionSSR.	
	Source: **0..*** Context.additional **Target:** **1** ContextRangeCheck.	The additional context specifications for a context range check.

Element "RCImplementation"

Parent Package: ContextRangeCheck

Stereotype: ,

Notes:

Defines the abstract type for referencing to possible realizations of the range check SSR.

Relationships

Name	Source/Target	Notes
	Source: <anonymous>. **Target:** RCImplementation.	
	Source: RTERangeCheck. **Target:** RCImplementation.	
	Source: BSWRangeCheck. **Target:** RCImplementation.	
	Source: SWCRangeCheck. **Target:** RCImplementation.	

Element "RTERangeCheck"

Parent Package: ContextRangeCheck

Stereotype: ,

Notes:

Range check implementation through configuration of the AUTOSAR RTE.

Relationships

Name	Source/Target	Notes
	Source: RTERangeCheck. **Target:** RCImplementation.	

Element "SWCRangeCheck"

Parent Package: ContextRangeCheck

Stereotype: ,

Notes:

Range check implementation using software components.

Relationships

Name	Source/Target	Notes
	Source: SWCRangeCheck. **Target:** RCImplementation.	

Package "CpuSelfTest"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

This package groups the elements related to the software safety mechanism CPU Self Test

Diagram "**CpuSelfTest**"

Notes:

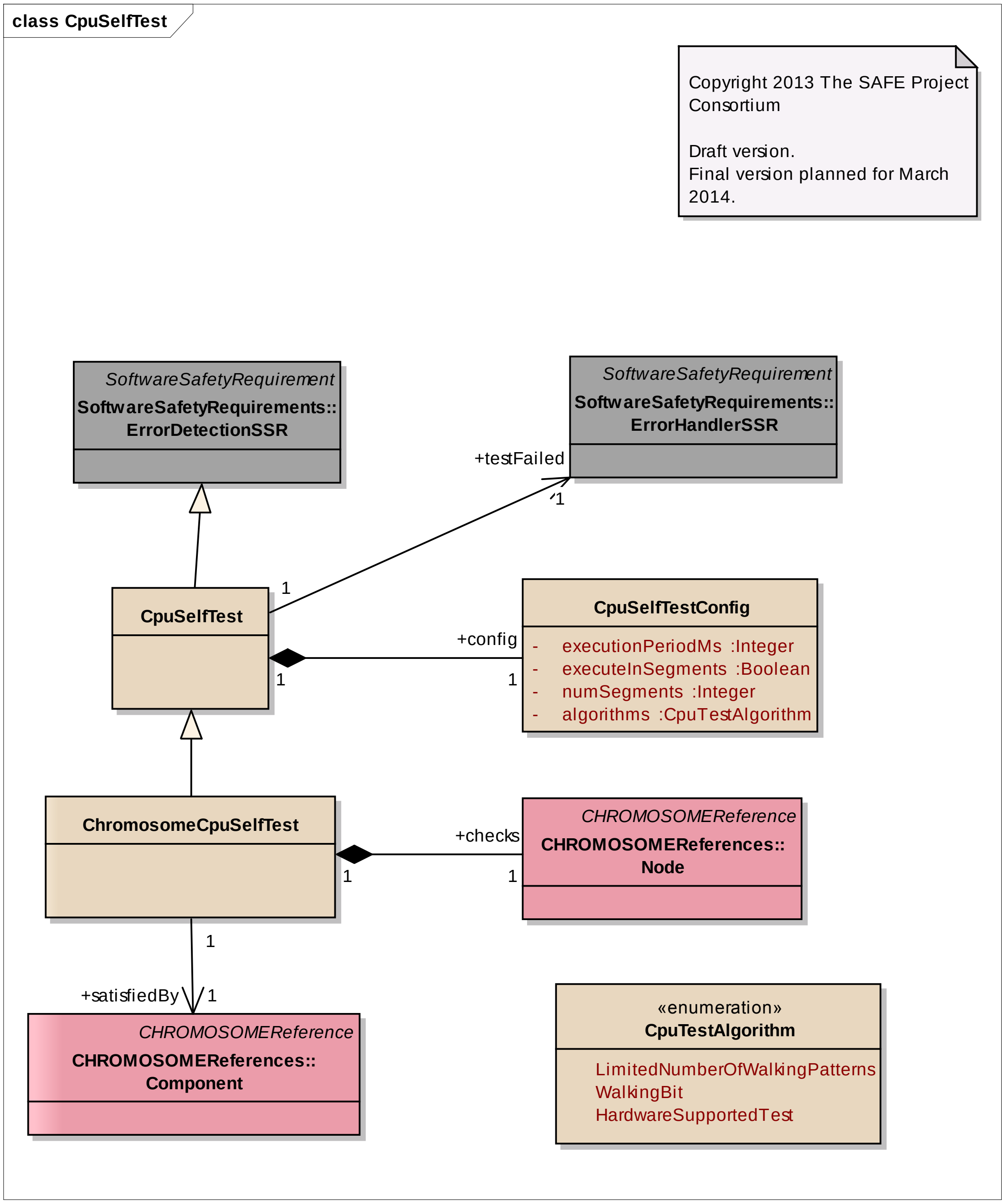


Figure: 50

Element "ChromosomeCpuSelfTest"

Parent Package: CpuSelfTest

Stereotype: ,

Notes:

CHROMOSOME specific CPU self-test meta class.

Relationships

Name	Source/Target	Notes
	Source: **1** Node.checks **Target:** **1** ChromosomeCpuSelfTest.	
	Source: **1** ChromosomeCpuSelfTest. **Target:** **1** Component.satisfiedBy	Generated CHROMOSOME component that satisfies the SSR.
	Source: ChromosomeCpuSelfTest. **Target:** CpuSelfTest.	

Element "CpuSelfTest"

Parent Package: CpuSelfTest

Stereotype: ,

Notes:

This meta-class element defines a CPU self-test SSR. The entity implementing such an SSR is executed periodically according to its configuration, and may trigger error handlers in case of error detection.

Relationships

Name	Source/Target	Notes
	Source: CpuSelfTest. **Target:** ErrorDetectionSSR.	
	Source: ChromosomeCpuSelfTest. **Target:** CpuSelfTest.	
	Source: **1** CpuSelfTestConfig.config **Target:** **1** CpuSelfTest.	Configures the CpuSelfTest SSR with execution configuration and definition of a specific test to be executed.
	Source: **1** CpuSelfTest. **Target:** **1** ErrorHandlerSSR.testFailed	The error handling SSR specification in case a CPU error occurs during the test execution.

Element "CpuSelfTestConfig"

Parent Package: CpuSelfTest

Stereotype: ,

Notes:

Configuration meta class to configure CPU test SSR.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	executionPeriodMs	Integer							
	executeInSegments	Boolean							
	numSegments	Integer							
	algorithms	CpuTestAlgorithm							

Relationships

Name	Source/Target	Notes
	Source: 1 CpuSelfTestConfig.config **Target:** 1 CpuSelfTest.	Configures the CpuSelfTest SSR with execution configuration and definition of a specific test to be executed.

Element "CpuTestAlgorithm"

Parent Package: CpuSelfTest

Stereotype: «enumeration»,

Notes:

Defines possible CPU test algorithms.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	LimitedNumberOfWalkingPatterns								
	WalkingBit								
	HardwareSupportedTest								

Package "Filter"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

Diagram "Filter"

Notes:

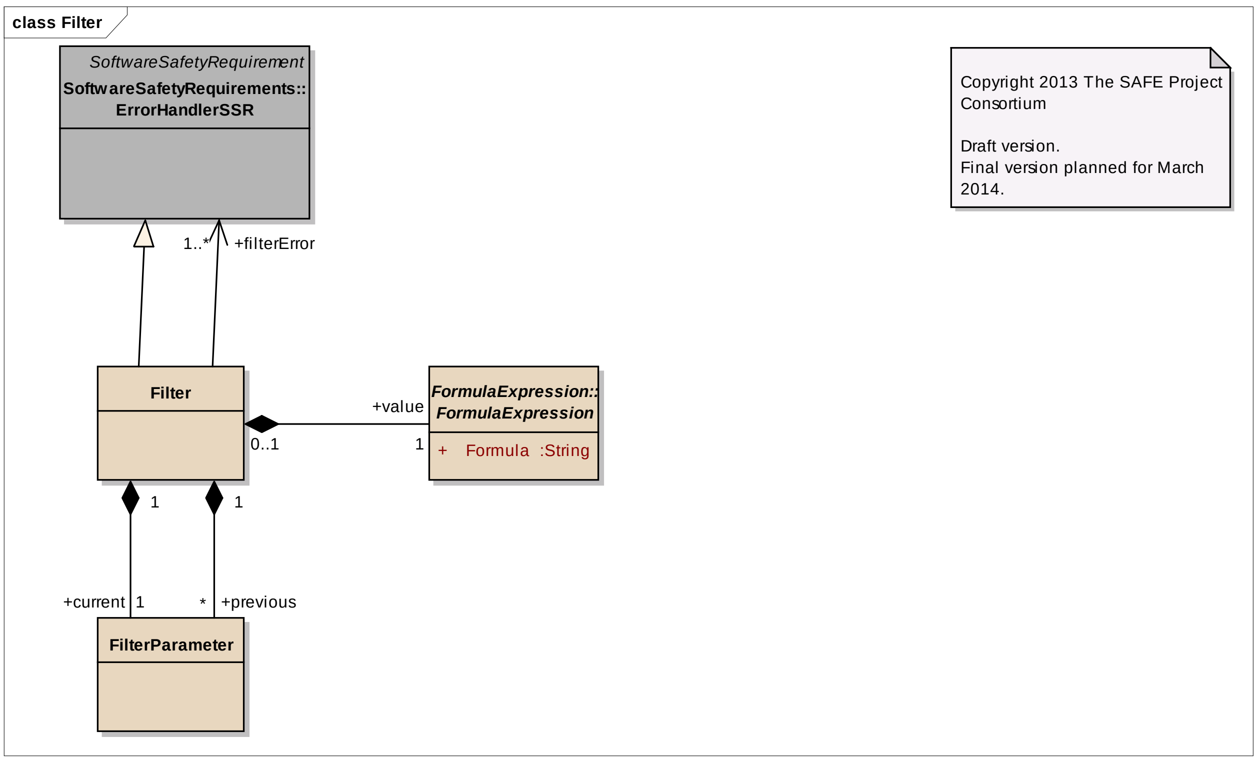


Figure: 51

Element "Filter"

Parent Package: Filter

Stereotype: ,

Notes:

The Filter meta element represents a error handling specification which defines how an erroneous value provided to it can be corrected. The mechanism can be referred by detection software safety requirements.

Relationships

Name	Source/Target	Notes
	Source: Filter. **Target:** ErrorHandlerSSR.	
	Source: **1** FilterParameter.current **Target:** **1** Filter.	The current value provided as input to the filter
	Source: Filter. **Target:** **1..*** ErrorHandlerSSR.filterError	The error handling SSR specification in case a filtering error occurs
	Source: ***** FilterParameter.previous **Target:** **1** Filter.	The previous value(s) of the filter which shall be taken into account while computing a new filtered value
	Source: **1** FormulaExpression.value **Target:** **0..1** Filter.	The formula defining how the value of the filter shall be computed

Element "FilterParameter"

Parent Package: Filter

Stereotype: ,

Notes:

A FilterParameter meta element defines the possible values which can be referred within a Filter. These are used to calculate and correct erroneous values provided to a filter realization.

Relationships

Name	Source/Target	Notes
	Source: **1** FilterParameter.current **Target:** **1** Filter.	The current value provided as input to the filter
	Source: ***** FilterParameter.previous **Target:** **1** Filter.	The previous value(s) of the filter which shall be taken into account while computing a new filtered value

Package "GradientCheck"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

This package groups the elements related to the software safety mechanism Gradient Check

Diagram "GradientCheck"

Notes:

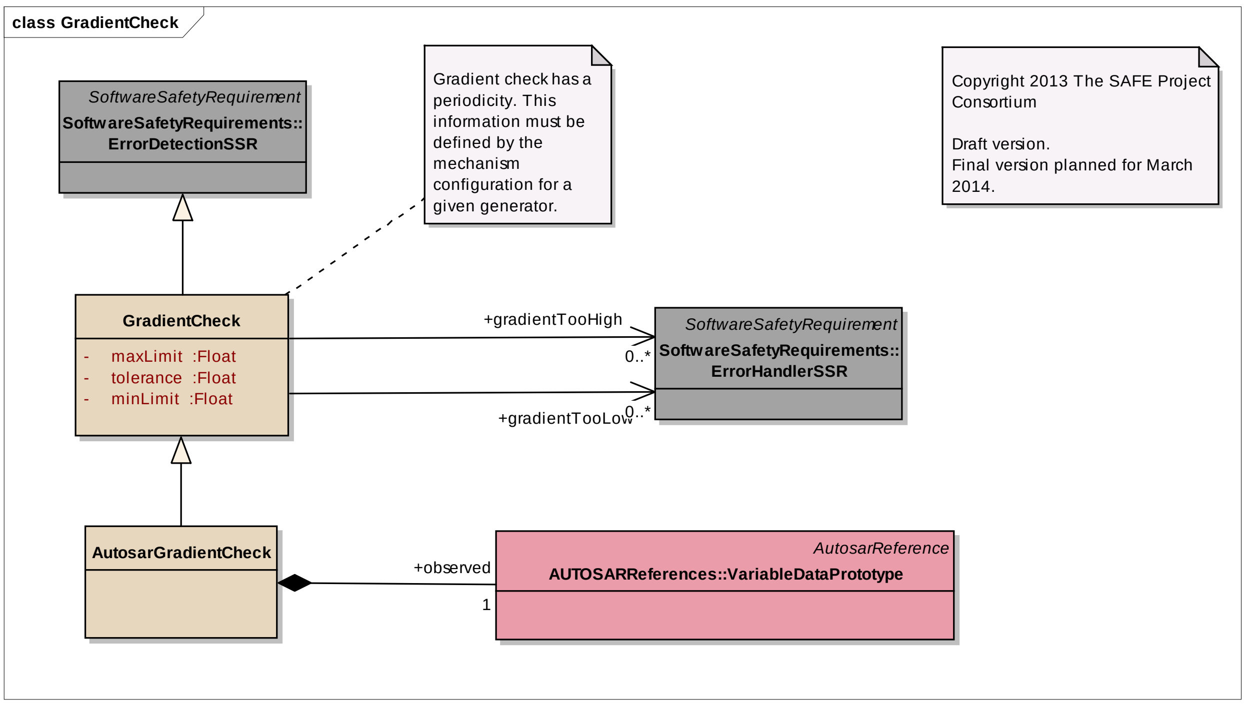


Figure: 52

Element "AutosarGradientCheck"

Parent Package: GradientCheck

Stereotype: ,

Notes:

AUTOSAR specific meta class for defining the gradient check SSR.

Relationships

Name	Source/Target	Notes

Name	Source/Target	Notes
	Source: AutosarGradientCheck. **Target:** GradientCheck.	
	Source: **1** VariableDataPrototype.observed **Target:** AutosarGradientCheck.	The AUTOSAR variable data prototype instance to be monitored by the gradient check

Element "GradientCheck"

Parent Package: GradientCheck

Stereotype: ,

Notes:

This meta-class element defines a gradient check SSR. The limit and tolerance attributes are used to check if the values are varying within a valid range. The SSR is executed periodically according to its period attribute.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	maxLimit	Float			0	0	0		
	tolerance	Float			0	0	0		
	minLimit	Float			0	0	0		

Relationships

Name	Source/Target	Notes
	Source: GradientCheck. **Target:** **0..*** ErrorHandlerSSR.gradientTooHigh	The error handling SSR specification in case the gradient is too high
	Source: GradientCheck. **Target:** **0..*** ErrorHandlerSSR.gradientTooLow	The error handling SSR specification in case the gradient is too low
	Source: <anonymous>. **Target:** GradientCheck.	
	Source: AutosarGradientCheck. **Target:** GradientCheck.	

Name	Source/Target	Notes
	Source: GradientCheck. **Target:** ErrorDetectionSSR.	

Package "HealthMonitor"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

Diagram "HealthMonitor"

Notes:

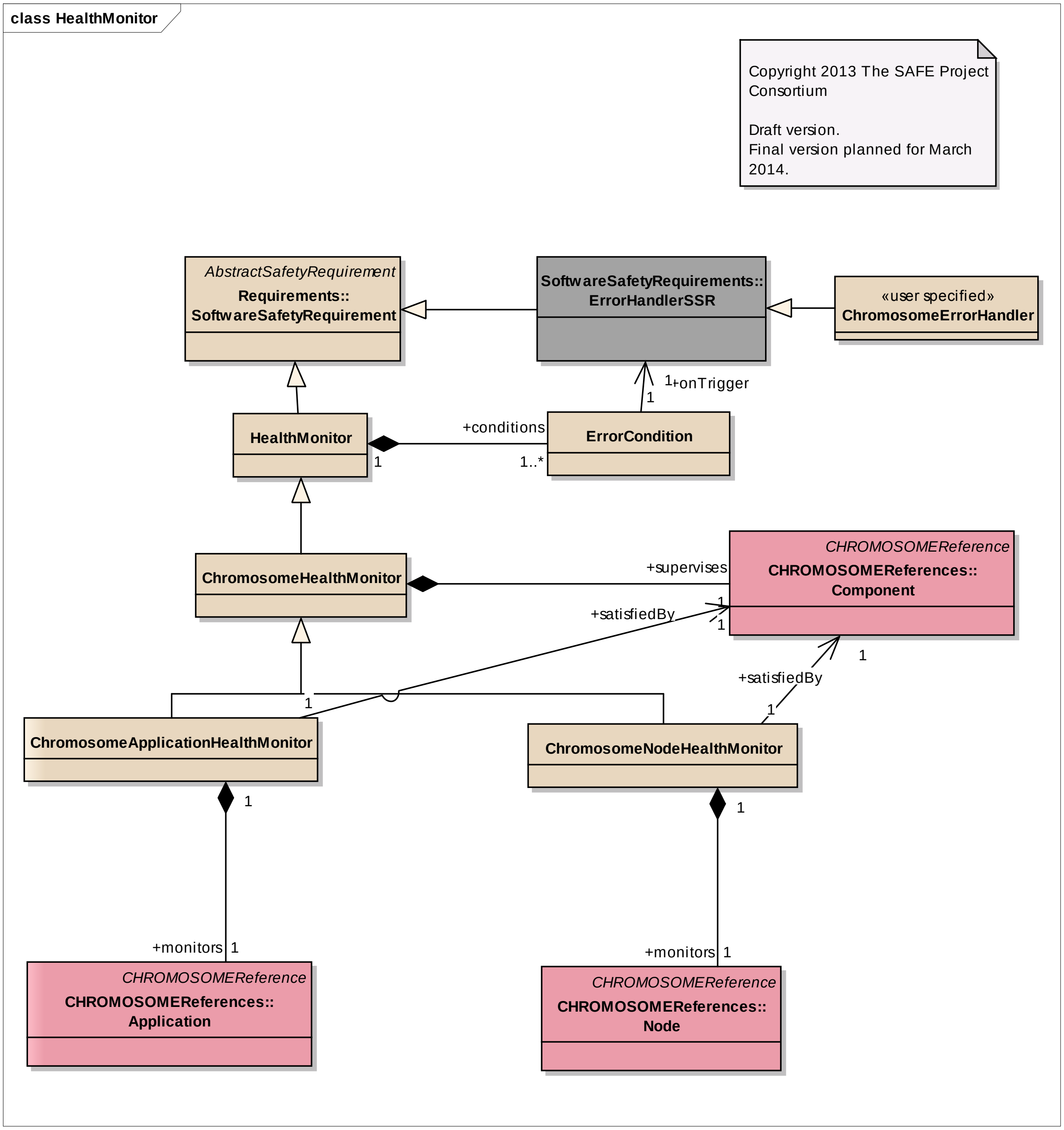


Figure: 53

Diagram "HealthMonitorErrorConditions"

Notes:

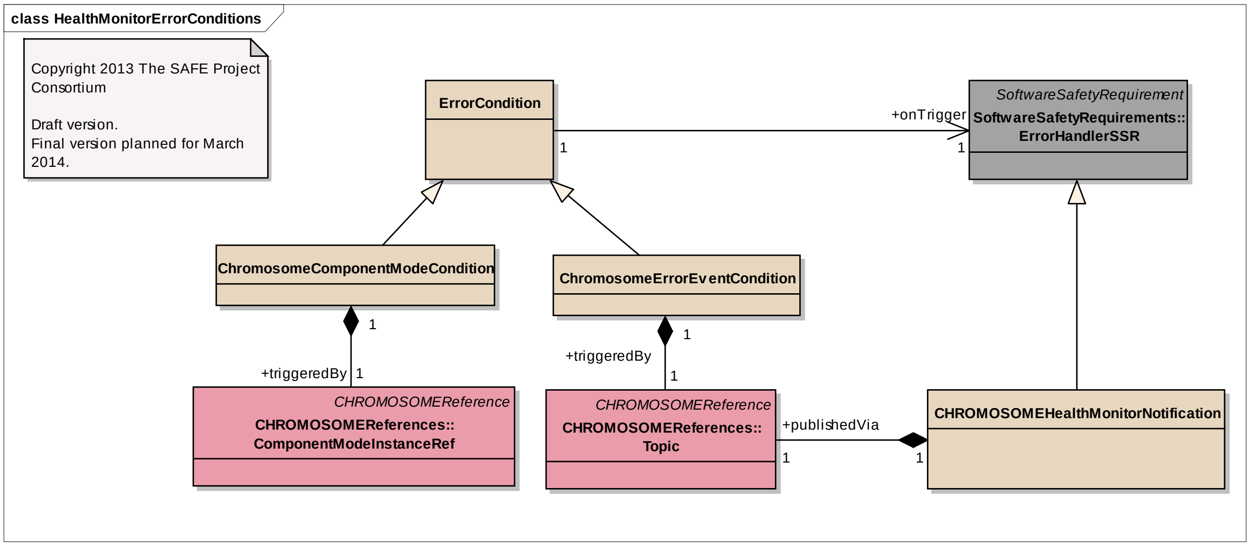


Figure: 54

Element "CHROMOSOMEHealthMonitorNotification"

Parent Package: HealthMonitor

Stereotype: ,

Notes:

Provides a mechanism to delegate error handling to the CHROMOSOME HealthMonitor.

Relationships

Name	Source/Target	Notes
	Source: CHROMOSOMEHealthMonitorNotification. **Target:** ErrorHandlerSSR.	
	Source: **1** Topic.publishedVia **Target:** **1** CHROMOSOMEHealthMonitorNotification.	Includes a reference to a CHROMOSOME topic used to transport error notifications.

Element "ChromosomeApplicationHealthMonitor"

Parent Package: HealthMonitor

Stereotype: ,

Notes:

CHROMOSOME specific Health Monitor meta class for monitoring a Chromosome Application as a set of CHROMOSOME components.

Relationships

Name	Source/Target	Notes
	Source: **1** Application.monitors **Target:** **1** ChromosomeApplicationHealthMonitor.	
	Source: **1** ChromosomeApplicationHealthMonitor. **Target:** **1** Component.satisfiedBy	Generated CHROMOSOME component that satisfies the SSR.
	Source: ChromosomeApplicationHealthMonitor. **Target:** ChromosomeHealthMonitor.	

Element "ChromosomeComponentModeCondition"

Parent Package: HealthMonitor

Stereotype: ,

Notes:

A CHROMOSOME specific condition referencing a Mode of a certain CHROMOSOME component instance. The condition is fired when the component is monitored by the Health Monitor and enters mode specified by triggeredBy relation.

Relationships

Name	Source/Target	Notes
	Source: **1** ComponentModeInstanceRef.triggeredBy **Target:** **1** ChromosomeComponentModeCondition.	Specifies a set of modes of CHROMOSOME component instances, which triggers the component mode condition.
	Source: ChromosomeComponentModeCondition. **Target:** ErrorCondition.	

Element "ChromosomeErrorEventCondition"

Parent Package: HealthMonitor

Stereotype: ,

Notes:

A CHROMOSOME specific error condition, which is triggered by reception of a message via a predefined CHROMOSOME topic referenced by triggeredBy relation.

Relationships

Name	Source/Target	Notes
	Source: ChromosomeErrorEventCondition. **Target:** ErrorCondition.	
	Source: 1 Topic.triggeredBy **Target: 1** ChromosomeErrorEventCondition.	Includes a reference to a CHROMOSOME topic used to transport error notifications from different componens / subsystems, and acting as a trigger for error condition.

Element "ChromosomeErrorHandler"

Parent Package: HealthMonitor

Stereotype: «user specified»,

Notes:

CHROMOSOME specific user-defined error handling function.

Relationships

Name	Source/Target	Notes
	Source: ChromosomeErrorHandler. **Target:** ErrorHandlerSSR.	

Element "ChromosomeHealthMonitor"

Parent Package: HealthMonitor

Stereotype: ,

Notes:

CHROMOSOME specific Health Monitor meta class.

Relationships

Name	Source/Target	Notes

Name	Source/Target	Notes
	Source: **1** Component.supervises **Target:** ChromosomeHealthMonitor.	Defines the set of supervised CHROMOSOME components.
	Source: ChromosomeApplicationHealthMonitor. **Target:** ChromosomeHealthMonitor.	
	Source: ChromosomeHealthMonitor. **Target:** HealthMonitor.	
	Source: ChromosomeNodeHealthMonitor. **Target:** ChromosomeHealthMonitor.	

Element "ChromosomeNodeHealthMonitor"

Parent Package: HealthMonitor

Stereotype: ,

Notes:

CHROMOSOME specific Health Monitor meta class for monitoring a Chromosome Node as a set of CHROMOSOME components.

Relationships

Name	Source/Target	Notes
	Source: **1** ChromosomeNodeHealthMonitor. **Target:** **1** Component.satisfiedBy	Generated CHROMOSOME component that satisfies the SSR.
	Source: **1** Node.monitors **Target:** **1** ChromosomeNodeHealthMonitor.	
	Source: ChromosomeNodeHealthMonitor. **Target:** ChromosomeHealthMonitor.	

Element "ErrorCondition"

Parent Package: HealthMonitor

Stereotype: ,

Notes:

An error condition specifies on one hand a trigger condition, and on another - the error handler corresponding to the trigger conditions.

Relationships

Name	Source/Target	Notes
	Source: **1** ErrorCondition. **Target:** **1** ErrorHandlerSSR.onTrigger	Error handling: executed when triggered by condition in Health Monitor.
	Source: ChromosomeErrorEventCondition. **Target:** ErrorCondition.	
	Source: ChromosomeComponentModeCondition. **Target:** ErrorCondition.	
	Source: **1..*** ErrorCondition.conditions **Target:** **1** HealthMonitor.	A set of error conditions monitored by the Health Monitor.

Element "HealthMonitor"

Parent Package: HealthMonitor

Stereotype: ,

Notes:

This meta-class element defines a health monitor SSR.

Relationships

Name	Source/Target	Notes
	Source: HealthMonitor. **Target:** SoftwareSafetyRequirement.	
	Source: **1..*** ErrorCondition.conditions **Target:** **1** HealthMonitor.	A set of error conditions monitored by the Health Monitor.
	Source: ChromosomeHealthMonitor. **Target:** HealthMonitor.	

Package "Heartbeat"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

This package groups the elements related to the software safety mechanism Heartbeat.

Diagram "Heartbeat"

Notes:

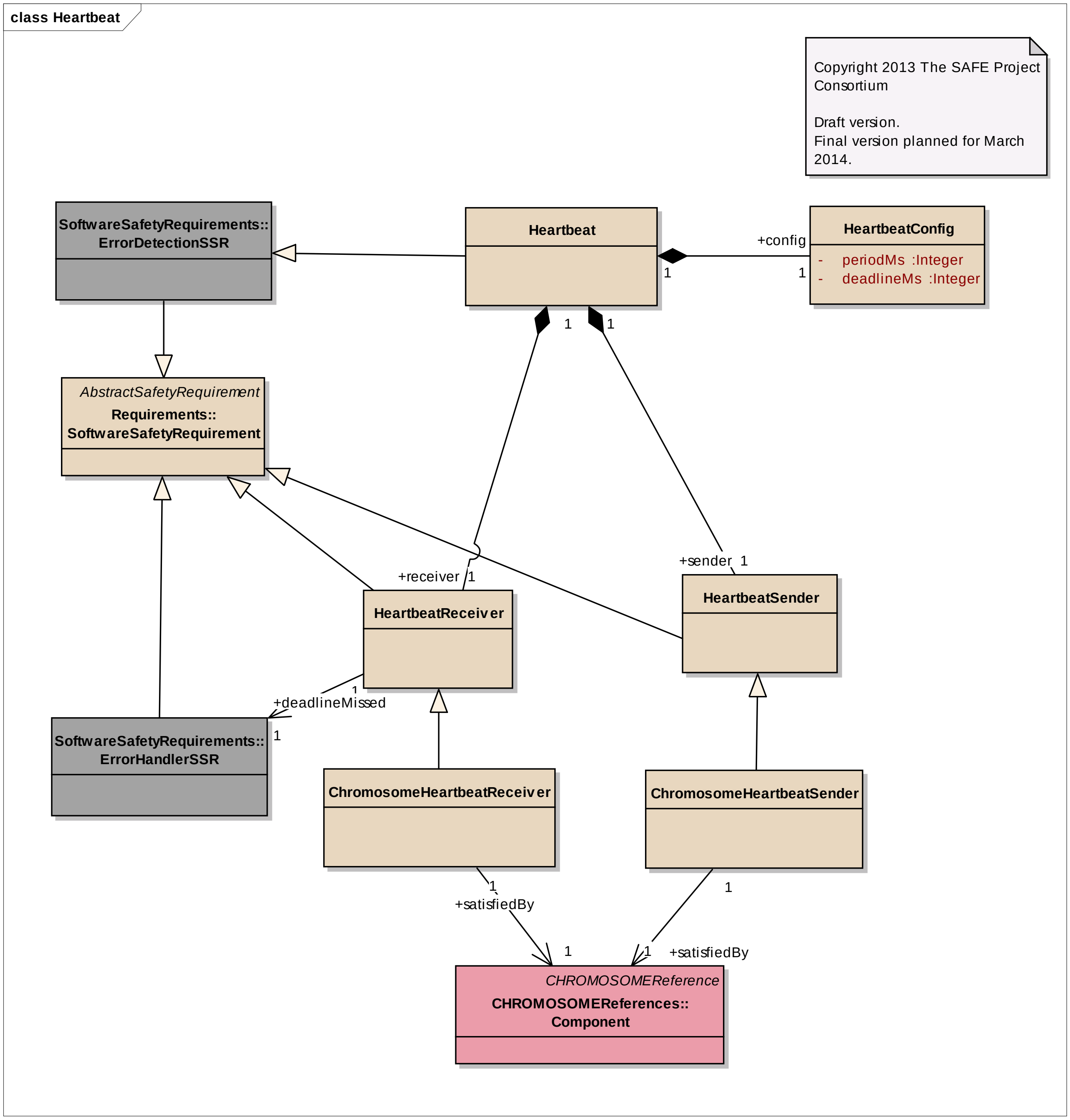


Figure: 55

Element "ChromosomeHeartbeatReceiver"

Parent Package: Heartbeat

Stereotype: ,

Notes:

A CHROMOSOME specific meta class to define a heartbeat receiver

Relationships

Name	Source/Target	Notes
	Source: 1 ChromosomeHeartbeatReceiver. **Target: 1** Component.satisfiedBy	Generated CHROMOSOME component that satisfies the SSR.
	Source: ChromosomeHeartbeatReceiver. **Target:** HeartbeatReceiver.	

Element "ChromosomeHeartbeatSender"

Parent Package: Heartbeat

Stereotype: ,

Notes:

A CHROMOSOME specific meta class to define a heartbeat sender.

Relationships

Name	Source/Target	Notes
	Source: 1 ChromosomeHeartbeatSender. **Target: 1** Component.satisfiedBy	Generated CHROMOSOME component that satisfies the SSR.
	Source: ChromosomeHeartbeatSender. **Target:** HeartbeatSender.	

Element "Heartbeat"

Parent Package: Heartbeat

Stereotype: ,

Notes:

A Heartbeat SSR requires that two runnable entities on different nodes are instantiated and one of them monitors the signals arriving from the other one.

Relationships

Name	Source/Target	Notes

Name	Source/Target	Notes
	Source: 1 HeartbeatConfig.config **Target: 1** Heartbeat.	Configures the Heartbeat SSR
	Source: 1 HeartbeatSender.sender **Target: 1** Heartbeat.	Contains hearbeet sender SSR
	Source: 1 HeartbeatReceiver.receiver **Target: 1** Heartbeat.	Contains hearbeet receiver
	Source: Heartbeat. **Target:** ErrorDetectionSSR.	

Element "HeartbeatConfig"

Parent Package: Heartbeat

Stereotype: ,

Notes:

Configuration of Heartbeat SSR: timing of source and receiver.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	periodMs	Integer							
	deadlineMs	Integer							

Relationships

Name	Source/Target	Notes
	Source: 1 HeartbeatConfig.config **Target: 1** Heartbeat.	Configures the Heartbeat SSR

Element "HeartbeatReceiver"

Parent Package: Heartbeat

Stereotype: ,

Notes:

An SSR representing the receiver of the heartbeat signal.

Relationships

Name	Source/Target	Notes
	Source: HeartbeatReceiver.	

Name	Source/Target	Notes
	Target: SoftwareSafetyRequirement.	
	Source: 1 HeartbeatReceiver.receiver **Target: 1** Heartbeat.	Contains hearbeet receiver
	Source: 1 HeartbeatReceiver. **Target: 1** ErrorHandlerSSR.deadlineMissed	The error handling SSR specification in case a deadline has been missed and no input received at receiver side.
	Source: ChromosomeHeartbeatReceiver. **Target:** HeartbeatReceiver.	

Element "HeartbeatSender"

Parent Package: Heartbeat

Stereotype: ,

Notes:

An SSR representing the sender of the heartbeat signal.

Relationships

Name	Source/Target	Notes
	Source: HeartbeatSender. **Target:** SoftwareSafetyRequirement.	
	Source: 1 HeartbeatSender.sender **Target: 1** Heartbeat.	Contains hearbeet sender SSR
	Source: ChromosomeHeartbeatSender. **Target:** HeartbeatSender.	

Package "MemorySelfTest"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

This package groups the elements related to the software safety mechanism Memory (RAM) Self Test

Diagram "MemorySelfTest"

Notes:

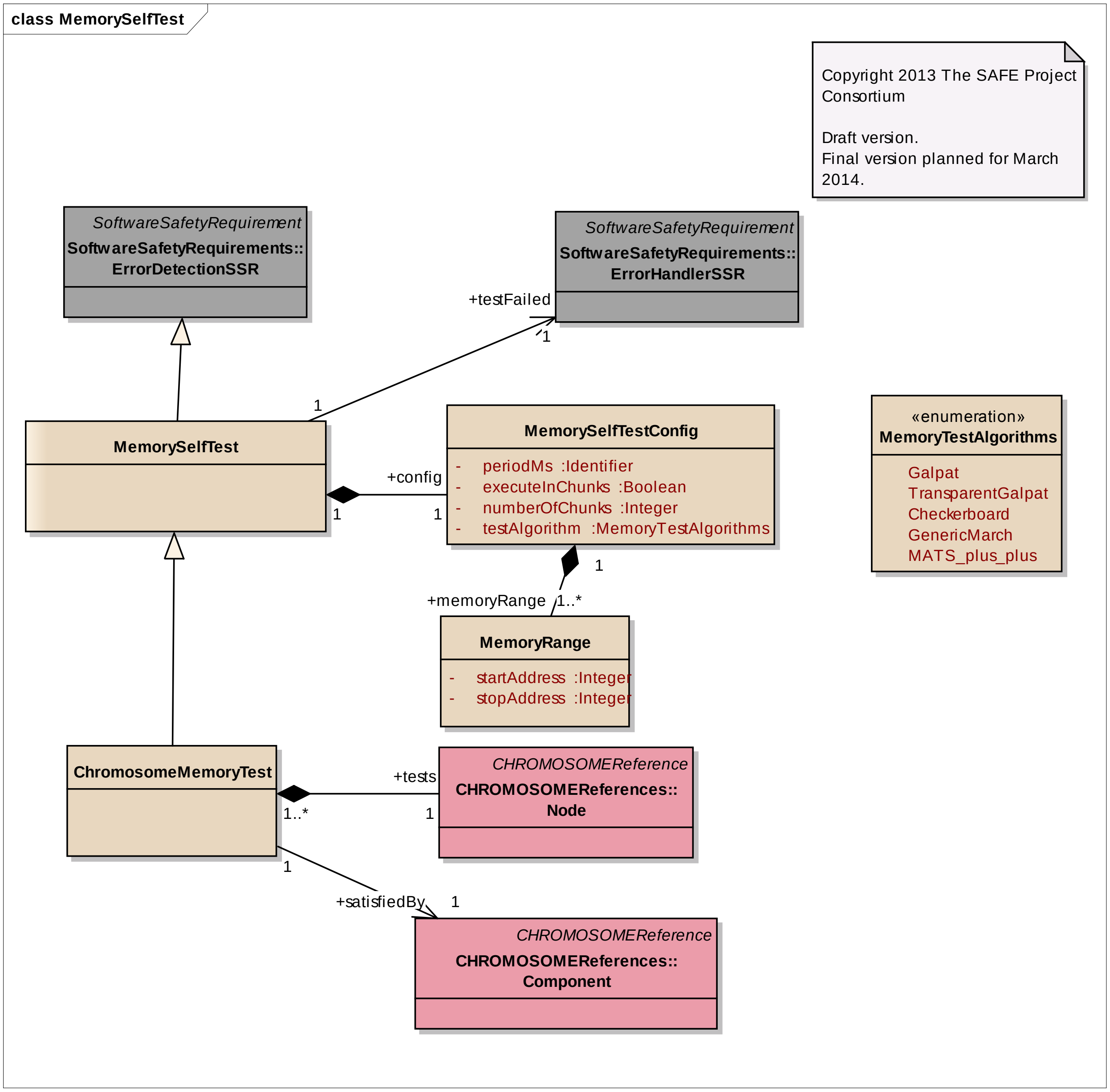


Figure: 56

Element "ChromosomeMemoryTest"

Parent Package: MemorySelfTest

Stereotype: ,

Notes:

A CHROMOSOME-specific MemorySelfTest requirement.

Relationships

Name	Source/Target			Notes
	Source:		ChromosomeMemoryTest.	
	Target:		MemorySelfTest.	
	Source:	**1**	ChromosomeMemoryTest.	A satisfy relation mapping MemorySelfTest SSRs to generated CHROMOSOME components.
	Target:	**1**	Component.satisfiedBy	
	Source:	**1**	Node.tests	
	Target:	**1..***	ChromosomeMemoryTest.	

Element "MemoryRange"

Parent Package: MemorySelfTest

Stereotype: ,

Notes:

An address range is a class to specify the exact range of addresses in the node address space to be tested.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	startAddress	Integer			0	0	0		
	stopAddress	Integer			0	0	0		

Relationships

Name	Source/Target			Notes
	Source:	**1..***	MemoryRange.memoryRange	
	Target:	**1**	MemorySelfTestConfig.	

Element "MemorySelfTest"

Parent Package: MemorySelfTest

Stereotype: ,

Notes:

A meta class defining Memory Self-Test SSR.

Relationships

Name	Source/Target			Notes
	Source:	**1**	MemorySelfTestConfig.config	
	Target:	**1**	MemorySelfTest.	
	Source:		ChromosomeMemoryTest.	
	Target:		MemorySelfTest.	
	Source:	**1**	MemorySelfTest.	The error handling SSR specification in case a RAM error occurs during the test execution.
	Target:	**1**	ErrorHandlerSSR.testFailed	
	Source:		MemorySelfTest.	
	Target:		ErrorDetectionSSR.	

Element "MemorySelfTestConfig"

Parent Package: MemorySelfTest

Stereotype: ,

Notes:

Specifies configuration parameters for MemorySelfTest SSR.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	periodMs	Identifier							
	executeInChunks	Boolean							
	numberOfChunks	Integer							
	testAlgorithm	MemoryTestAlgorithms							

Relationships

Name	Source/Target			Notes
	Source:	**1**	MemorySelfTestConfig.config	
	Target:	**1**	MemorySelfTest.	
	Source:	**1..***	MemoryRange.memoryRange	
	Target:	**1**	MemorySelfTestConfig.	

Element "MemoryTestAlgorithms"

Parent Package: MemorySelfTest

Stereotype: «enumeration»,

Notes:

Defines different memory test algorithms that can be defined in a configuration

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	Galpat				0	0	0		
	TransparentGalpat				0	0	0		
	Checkerboard				0	0	0		
	GenericMarch				0	0	0		
	MATS_plus_plus								

Package "Voting"

Type of Package: **Package**

Parent Package: SoftwareSafetyRequirements

Notes:

This package groups the elements related to the software safety mechanism Voting

Diagram "**Voting**"

Notes:

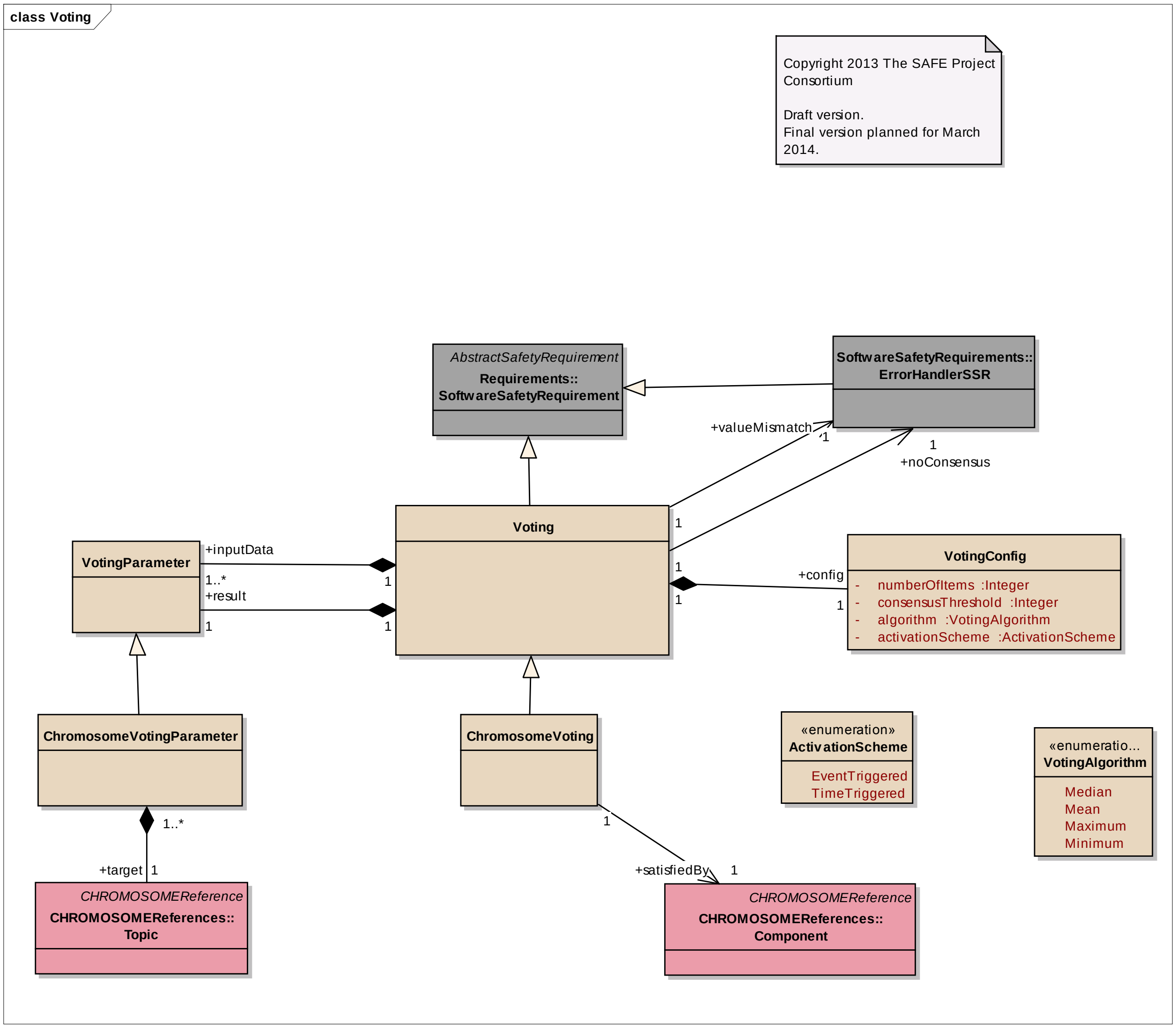


Figure: 57

Element "ActivationScheme"

Parent Package: Voting

Stereotype: «enumeration»,

Notes:

Possible activation schemes for a Voting component.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	EventTriggered				0	0	0		
	TimeTriggered				0	0	0		

Element "ChromosomeVoting"

Parent Package: Voting

Stereotype: ,

Notes:

A CHROMOSOME-specific meta class for Voting

Relationships

Name	Source/Target	Notes
	Source: ChromosomeVoting. **Target:** Voting.	
	Source: 1 ChromosomeVoting. **Target: 1** Component.satisfiedBy	Generated CHROMOSOME component that satisfies the SSR implementing SSM.

Element "ChromosomeVotingParameter"

Parent Package: Voting

Stereotype: ,

Notes:

A CHROMOSOME-specific voting parameter meta class.

Relationships

Name	Source/Target	Notes
	Source: ChromosomeVotingParameter. **Target:** VotingParameter.	
	Source: 1 Topic.target **Target: 1..*** ChromosomeVotingParameter.	References the topic instances, which represent the data for voting and output values

Element "Voting"

Parent Package: Voting

Stereotype: ,

Notes:

A meta-class representing the Voting software safety requirement.

Relationships

Name	Source/Target	Notes
	Source: **1** VotingParameter.result **Target:** **1** Voting.	Voting result
	Source: ChromosomeVoting. **Target:** Voting.	
	Source: **1..*** VotingParameter.inputData **Target:** **1** Voting.	Input data for voting.
	Source: Voting. **Target:** SoftwareSafetyRequirement.	
	Source: **1** VotingConfig.config **Target:** **1** Voting.	Configuration of a Voting SSR
	Source: **1** Voting. **Target:** **1** ErrorHandlerSSR.noConsensus	The error handling SSR specification in case one (or more) values demonstrate mismatch, and the consensus could not be found.
	Source: **1** Voting. **Target:** **1** ErrorHandlerSSR.valueMismatch	The error handling SSR specification in case one (or more) values demonstrate large mismatch, but the consensus still can be found.

Element "VotingAlgorithm"

Parent Package: Voting

Stereotype: «enumeration»,

Notes:

Possible voting algorithms: if the values are different, which one should be returned?

Attributes

PK	Name	Type	Not	Unique	Len	Pre	Scale	Init	Notes

			Null			c			
	Median				0	0	0		
	Mean				0	0	0		
	Maximum				0	0	0		
	Minimum				0	0	0		

Element "VotingConfig"

Parent Package: Voting

Stereotype: ,

Notes:

Configuration of a Voting SSR

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	numberOfItems	Integer							
	consensusThreshold	Integer							
	algorithm	VotingAlgorithm							
	activationScheme	ActivationScheme							

Relationships

Name	Source/Target	Notes
	Source: 1 VotingConfig.config **Target:** 1 Voting.	Configuration of a Voting SSR

Element "VotingParameter"

Parent Package: Voting

Stereotype: ,

Notes:

A meta-class for representing input data and outputs of a Voting SSR.

Relationships

Name	Source/Target	Notes

Name	Source/Target	Notes
	Source: **1** VotingParameter.result **Target:** **1** Voting.	Voting result
	Source: ChromosomeVotingParameter. **Target:** VotingParameter.	
	Source: **1..*** VotingParameter.inputData **Target:** **1** Voting.	Input data for voting.

Package "Software"

Type of Package: **Package**

Parent Package: SAFE Meta-Model

Notes:

Package "Configuration"

Type of Package: **Package**

Parent Package: Software

Notes:

Diagram "Configuration"

Notes:

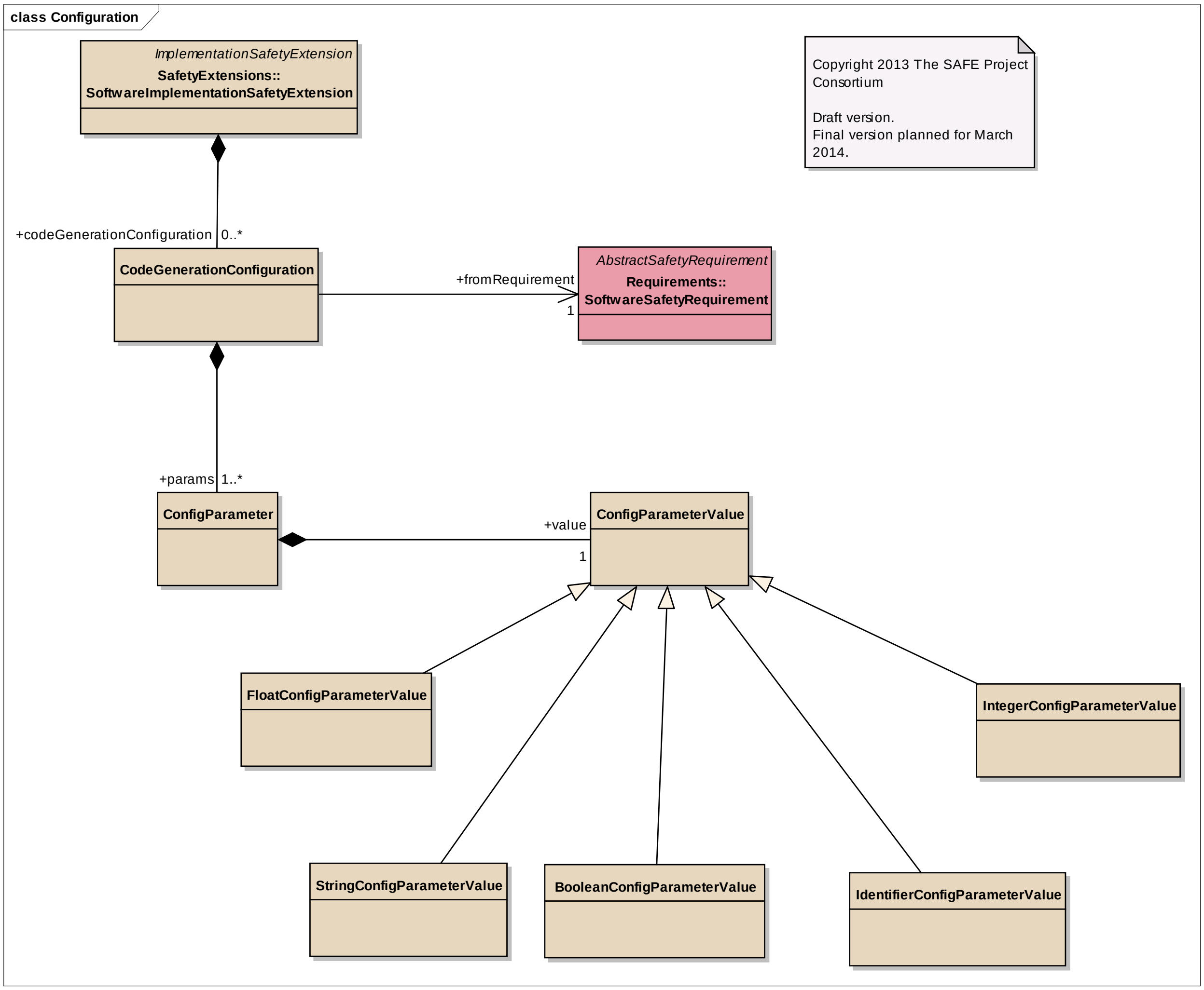


Figure: 58

Element "BooleanConfigParameterValue"

Parent Package: Configuration

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: BooleanConfigParameterValue. **Target:** ConfigParameterValue.	

Element "CodeGenerationConfiguration"

Parent Package: Configuration

Stereotype: ,

Notes:

An SSMConfiguration defines the parameters needed by specific code generators. It provides a set of name/value pairs which contain additional information not defined within the software safety mechanism.

Relationships

Name	Source/Target	Notes
	Source: **0..*** CodeGenerationConfiguration.codeGenerationConfiguration **Target:** SoftwareImplementationSafetyExtension.	
	Source: **0..*** CodeGenerationConfiguration.configuration **Target:** SoftwareSafetyDesign.	Configurations contained in the SafeExtension.
	Source: **1..*** ConfigParameter.params **Target:** CodeGenerationConfiguration.	Parameters necessary for the configuration of a safety mechanism.
	Source: CodeGenerationConfiguration. **Target:** **1** SoftwareSafetyRequirement.fromRequirement	The mechanism configured by the SSMConfiguration element
	Source: **0..*** CodeGenerationConfiguration.codegenconfiguration **Target:** TechnicalSafetyRequirement.	

Element "ConfigParameter"

Parent Package: Configuration

Stereotype: ,

Notes:

A ConfigParameter contains the name of the parameter and the value for this parameter.

Relationships

Name	Source/Target	Notes
	Source: **1..*** ConfigParameter.params	Parameters

Name	Source/Target	Notes
	Target: CodeGenerationConfiguration.	necessary for the configuration of a safety mechanism.
	Source: **1** ConfigParameterValue.value **Target:** ConfigParameter.	The value specification of a ConfigParameter element.

Element "ConfigParameterValue"

Parent Package: Configuration

Stereotype: ,

Notes:

A ConfigParameterValue is the abstract element allowing extensions to be made for code generator parameters. In this fashion new parameter value types can be added later on as needed.

Relationships

Name	Source/Target	Notes
	Source: IdentifierConfigParameterValue. **Target:** ConfigParameterValue.	
	Source: ConstantComparisonParameter. **Target:** ConfigParameterValue.	
	Source: IntegerConfigParameterValue. **Target:** ConfigParameterValue.	
	Source: **1** ConfigParameterValue.value **Target:** ConfigParameter.	The value specification of a ConfigParameter element.
	Source: FloatConfigParameterValue. **Target:** ConfigParameterValue.	
	Source: BooleanConfigParameterValue. **Target:** ConfigParameterValue.	
	Source: StringConfigParameterValue. **Target:** ConfigParameterValue.	

Element "FloatConfigParameterValue"

Parent Package: Configuration

Stereotype: ,

Notes:

The FloatConfigParameterValue defines the float parameter value type.

Relationships

Name	Source/Target	Notes
	Source: FloatConfigParameterValue. **Target:** ConfigParameterValue.	

Element "IdentifierConfigParameterValue"

Parent Package: Configuration

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: IdentifierConfigParameterValue. **Target:** ConfigParameterValue.	

Element "IntegerConfigParameterValue"

Parent Package: Configuration

Stereotype: ,

Notes:

This element defines a specific parameter value type, in this case an integer value.

Relationships

Name	Source/Target	Notes
	Source: IntegerConfigParameterValue. **Target:** ConfigParameterValue.	

Element "StringConfigParameterValue"

Parent Package: Configuration

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: StringConfigParameterValue. **Target:** ConfigParameterValue.	

Package "System"

Type of Package: **Package**

Parent Package: SAFE Meta-Model

Notes:

6 Appendix: Change request to external meta-models

This section provides the specification of change requests to external meta-models.

The packages, diagrams and elements of this part are not part of the SAFE meta-model. They are change requests to external meta-models (e.g. AUTOSAR, EAST-ADL) only. These changes are needed such that the SAFE meta-model can reference them to ensure the safety lifecycle compliant with ISO26262.

Package "Change Requests to External Meta-Models"

Type of Package: **Package**

Parent Package:

Notes:

Package "Change Request to AUTOSAR Meta-Model"

Type of Package: **Package**

Parent Package: Change Requests to External Meta-Models

Notes:

Diagram "**Change Request to AUTOSAR Meta-Model**"

Notes:

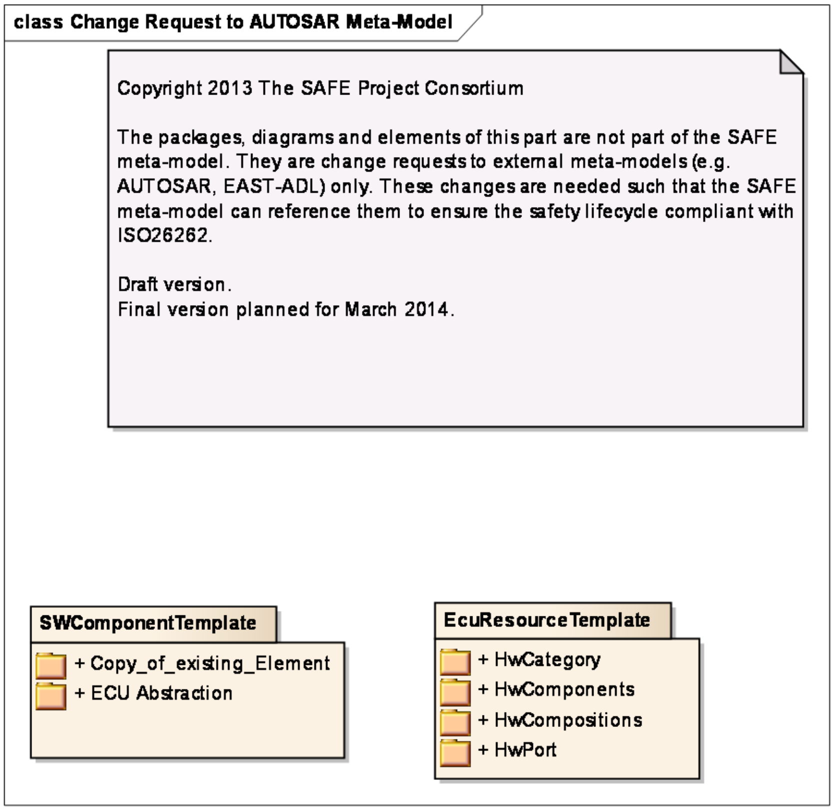


Figure: 1

Element "<anonymous>"

Parent Package: Change Request to AUTOSAR Meta-Model

Stereotype: ,

Notes:

Copyright 2013 The SAFE Project Consortium

The packages, diagrams and elements of this part are not part of the SAFE meta-model. They are change requests to external meta-models (e.g. AUTOSAR, EAST-ADL) only. These changes are needed such that the SAFE meta-model can reference them to ensure the safety lifecycle compliant with ISO26262.

Draft version.

Final version planned for March 2014.

Element "<anonymous>"

Parent Package: Change Request to AUTOSAR Meta-Model

Stereotype: ,

Notes:

Copyright 2013 The SAFE Project Consortium

The packages, diagrams and elements of this part are not part of the SAFE meta-model. They are change requests to external meta-models (e.g. AUTOSAR, EAST-ADL) only. These changes are needed such that the SAFE meta-model can reference them to ensure the safety lifecycle compliant with ISO26262.

Draft version.

Final version planned for March 2014.

Package "SWComponentTemplate"

Type of Package: **Package**

Parent Package: Change Request to AUTOSAR Meta-Model

Notes:

Package "Copy_of_existing_Element"

Type of Package: **Package**

Parent Package: SWComponentTemplate

Notes:

Element "ComplexDeviceDriverSwComponentType"

Parent Package: Copy_of_existing_Element

Stereotype: ,

Notes:

The ComplexDeviceDriverSwComponentType is a special AtomicSwComponentType that has direct access to hardware on an ECU and which is therefore linked to a specific ECU or specific hardware. The ComplexDeviceDriverSwComponentType introduces the possibility to link from the software representation to its hardware description provided by the ECU Resource Template.

Relationships

Name	Source/Target	Notes
	Source: ComplexDeviceDriverSwComponentType. **Target:** **0..*** HwElementType.hardwareElement	

Element "EcuAbstractionSwComponentType"

Parent Package: Copy_of_existing_Element

Stereotype: ,

Notes:

The ECUAbstraction is a special AtomicSwComponentType that resides between a software-component that wants to access ECU periphery and the Microcontroller Abstraction. The EcuAbstractionSwComponentType introduces the possibility to link from the software representation to its hardware description provided by the ECU Resource Template.

Relationships

Name	Source/Target	Notes
	Source: EcuAbstractionSwComponentType. **Target:** **0..*** HwElementType.hardwareElement	

Element "SensorActuatorSwComponentType"

Parent Package: Copy_of_existing_Element

Stereotype: ,

Notes:

The SensorActuatorSwComponentType introduces the possibility to link from the software representation of a sensor/actuator to its hardware description provided by the ECU Resource Template.

Relationships

Name	Source/Target	Notes
	Source: SensorActuatorSwComponentType. **Target:** **1** HwElementType.sensorActuator	

Package "ECU Abstraction"

Type of Package: **Package**

Parent Package: SWComponentTemplate

Notes:

Diagram "DOC_ComplexDeviceDriverComponent"

Notes:

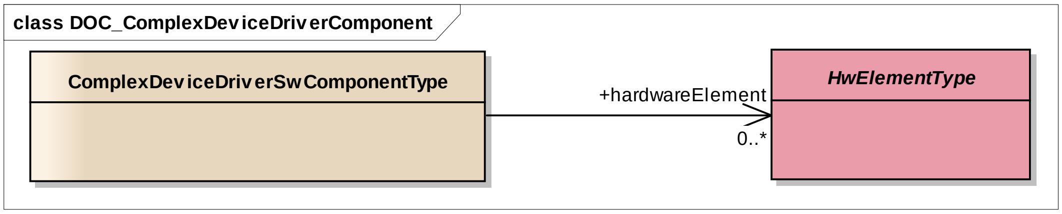


Figure: 2

Diagram "DOC_ECUAbstractionComponent"

Notes:

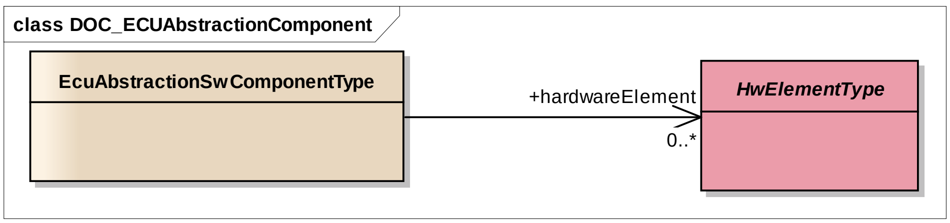


Figure: 3

Diagram "DOC_SensorActuatorComponent"

Notes:

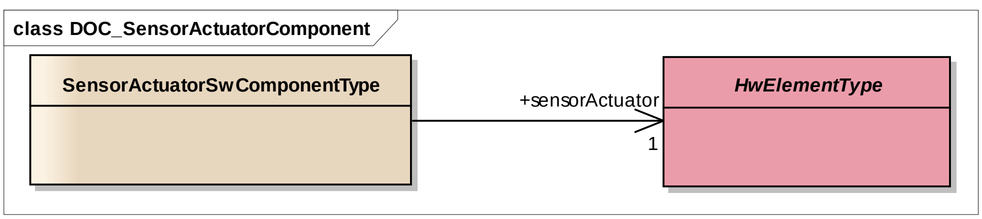


Figure: 4

Package "EcuResourceTemplate"

Type of Package: **Package**

Parent Package: Change Request to AUTOSAR Meta-Model

Notes:

Note! Changes have an impact on other templates, for example references to the former semantics of the element HwElement.

May be HwElmentPrototype shall be called HwElement and HwElement shall be called AtomicHwElement.

Diagram "DOC_EcuResscourceOverview"

Notes:

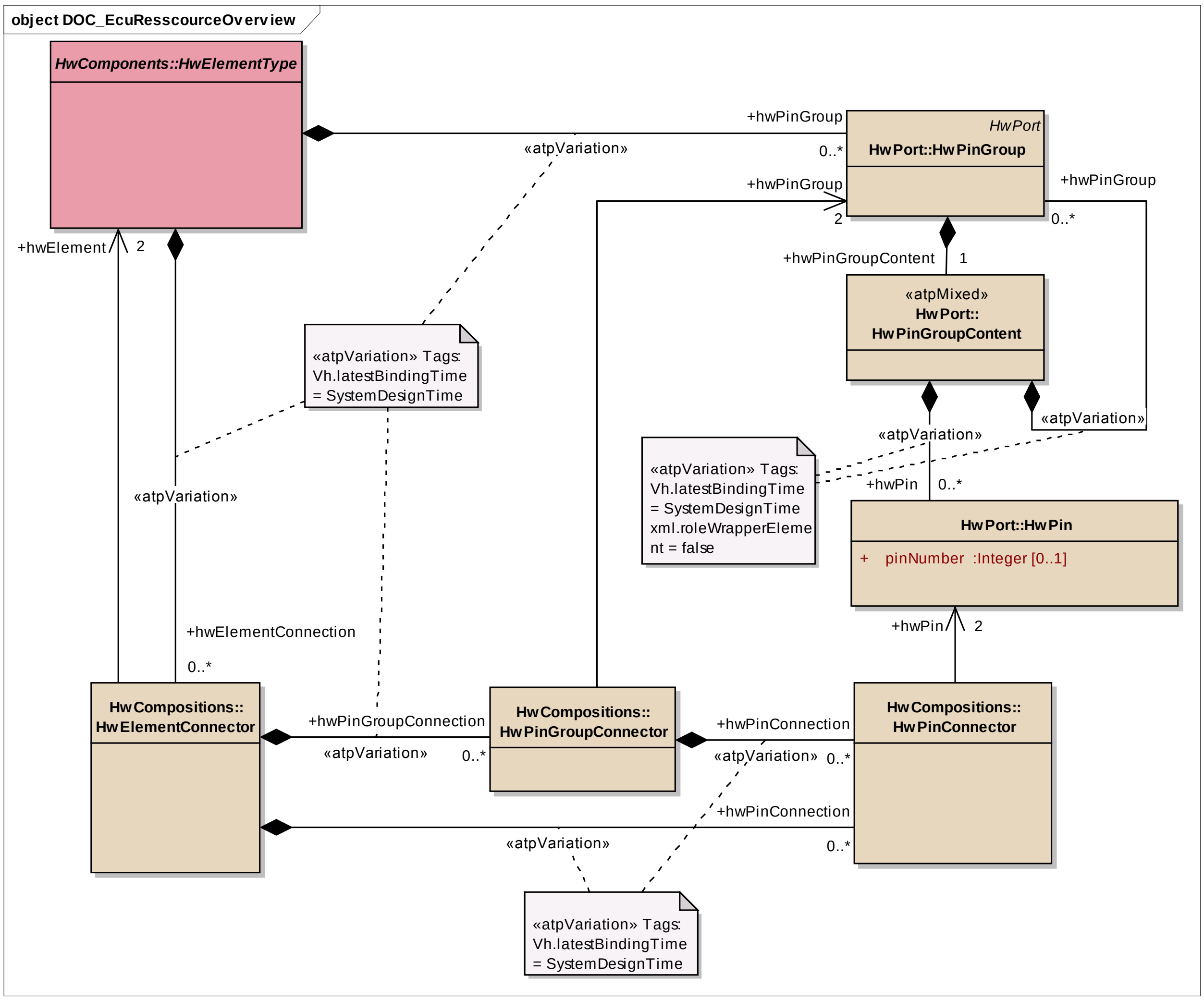


Figure: 5

Element "<anonymous>"

Parent Package: EcuResourceTemplate

Stereotype: ,

Notes:

Element "<anonymous>"

Parent Package: EcuResourceTemplate

Stereotype: ,

Notes:

Element "<anonymous>"

Parent Package: EcuResourceTemplate

Stereotype: ,

Notes:

Package "HwCategory"

Type of Package: **Package**

Parent Package: EcuResourceTemplate

Notes:

Diagram "DOC_HwCategory"

Notes:

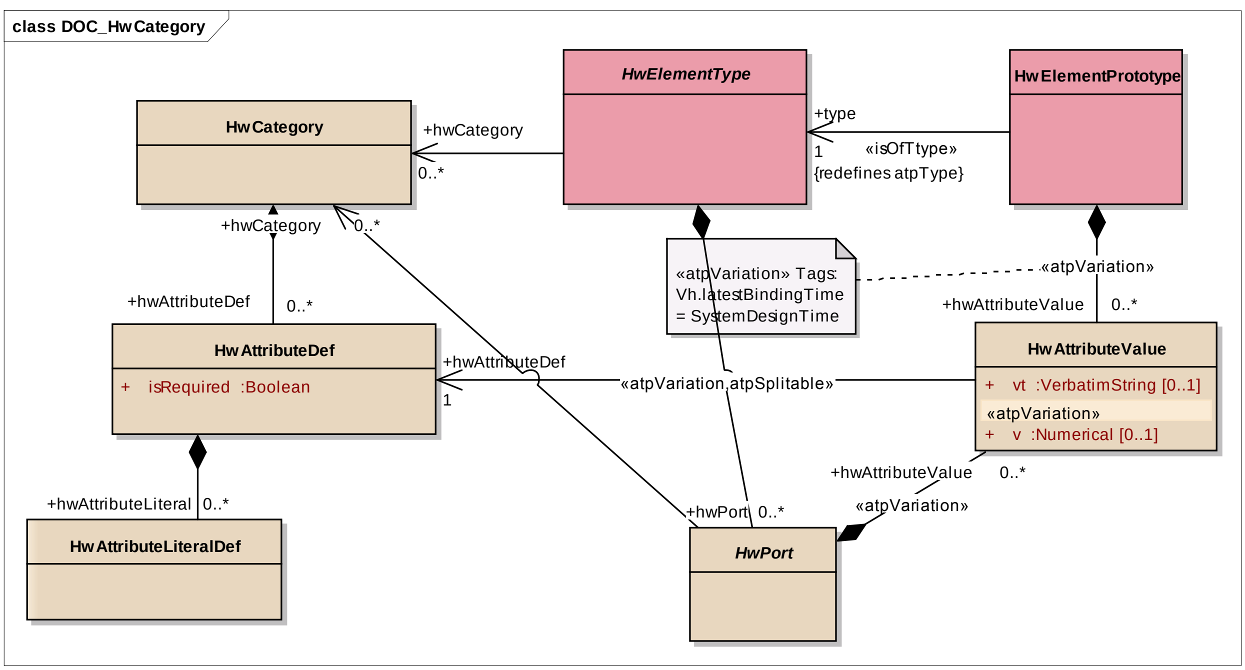


Figure: 6

Element "HwAttributeDef"

Parent Package: HwCategory

Stereotype: ,

Notes:

This metaclass represents the ability to define a particular hardware attribute.

The category of this element defines the type of the attributeValue. If the category is Enumeration the hwAttributeEnumerationLiterals specify the available literals.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	isRequired	Boolean			0	0	0		This attribute specifies if the defined attribute value is required to be provided.

Relationships

Name	Source/Target	Notes
	Source: HwAttributeValue. **Target: 1** HwAttributeDef.hwAttributeDef	
	Source: HwCategory. **Target: 0..*** HwAttributeDef.hwAttributeDef	
	Source: 0..* HwAttributeLiteralDef.hwAttributeLiteral **Target:** HwAttributeDef.	

Element "HwAttributeLiteralDef"

Parent Package: HwCategory

Stereotype: ,

Notes:

One available EnumerationLiteral of the Enumeration definition. Only applicable if the category of the HwAttributeDef equals Enumeration.

Relationships

Name	Source/Target	Notes
	Source: **0..*** HwAttributeLiteralDef.hwAttributeLiteral **Target:** HwAttributeDef.	

Element "HwAttributeValue"

Parent Package: HwCategory

Stereotype: ,

Notes:

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

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	v	Numerical			0	0	0		This represents a numerical hardware attribute value.
	vt	VerbatimString			0	0	0		This represents a textual hardware attribute value.

Relationships

Name	Source/Target	Notes
	Source: HwAttributeValue. **Target:** **1** HwAttributeDef.hwAttributeDef	
	Source: HwElementPrototype. **Target:** **0..*** HwAttributeValue.hwAttributeValue	
	Source: **0..*** HwAttributeValue.hwAttributeValue **Target:** HwPort.	

Element "HwCategory"

Parent Package: HwCategory

Stereotype: ,

Notes:

This metaclass represents the ability to declare hardware categories and its particular attributes.

Relationships

Name	Source/Target	Notes
	Source: HwElementType. **Target:** **0..*** HwCategory.hwCategory	
	Source: HwCategory. **Target:** **0..*** HwAttributeDef.hwAttributeDef	
	Source: HwPort. **Target:** **0..*** HwCategory.hwCategory	

Package "HwComponents"

Type of Package: **Package**

Parent Package: EcuResourceTemplate

Notes:

Diagram "DOC_HwComponents"

Notes:

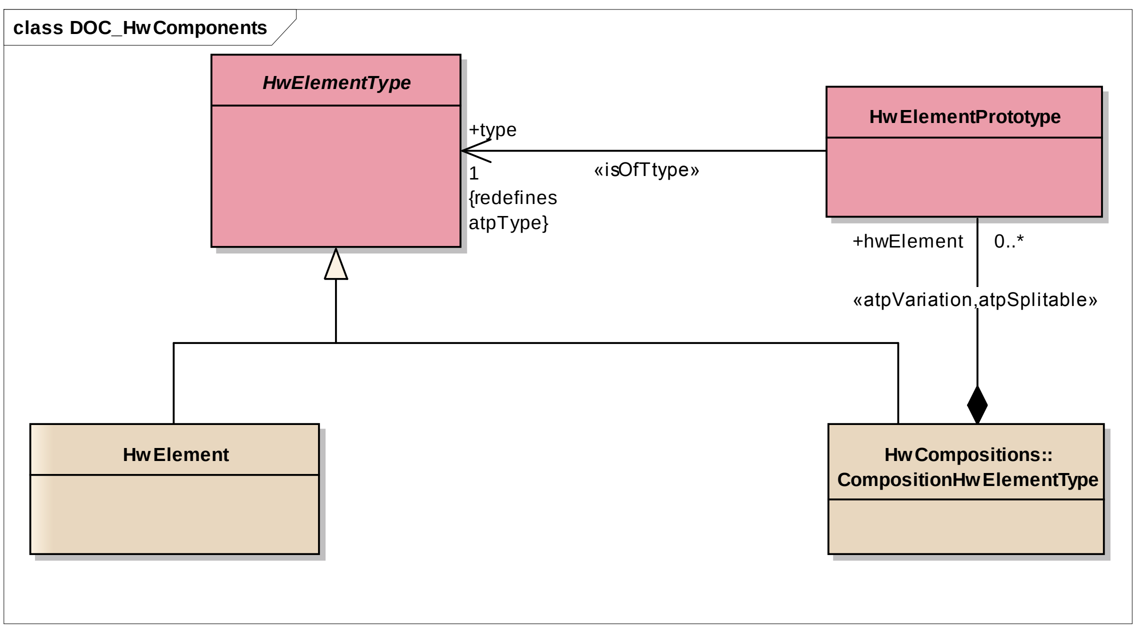


Figure: 7

Element "HwElement"

Parent Package: HwComponents

Stereotype: ,

Notes:

This represents the ability to describe Hardware Elements on an instance level. The particular types of hardware are distinguished by the category. This category determines the applicable attributes. The possible categories and attributes are defined in HwCategory.

Relationships

Name	Source/Target	Notes
	Source: HwElement. **Target:** HwElementType.	

Element "HwElementPrototype"

Parent Package: HwComponents

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: HwElementPrototype. **Target:** **1** HwElementType.type	
	Source: **0..*** HwElementPrototype.hwElement **Target:** CompositionHwElementType.	
	Source: HwElementPrototype. **Target:** **0..*** HwAttributeValue.hwAttributeValue	

Element "HwElementType"

Parent Package: HwComponents

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: HwElementPrototype.	

Name	Source/Target	Notes
	Target: **1** HwElementType.type	
	Source: HwElement. **Target:** HwElementType.	
	Source: CompositionHwElementType. **Target:** HwElementType.	
	Source: **0..*** HwPort.hwPort **Target:** HwElementType.	
	Source: HwElementType. **Target:** **0..*** HwCategory.hwCategory	
	Source: HwElementType. **Target:** **0..*** HwElementConnector.hwElementConnection	
	Source: EcuAbstractionSwComponentType. **Target:** **0..*** HwElementType.hardwareElement	
	Source: HwElementConnector. **Target:** **2** HwElementType.hwElement	
	Source: ComplexDeviceDriverSwComponentType. **Target:** **0..*** HwElementType.hardwareElement	
	Source: SensorActuatorSwComponentType. **Target:** **1** HwElementType.sensorActuator	
	Source: HwElementType. **Target:** **0..*** HwPinGroup.hwPinGroup	

Package "HwCompositions"

Type of Package: **Package**

Parent Package: EcuResourceTemplate

Notes:

Diagram "DOC_HwCompositions"

Notes:

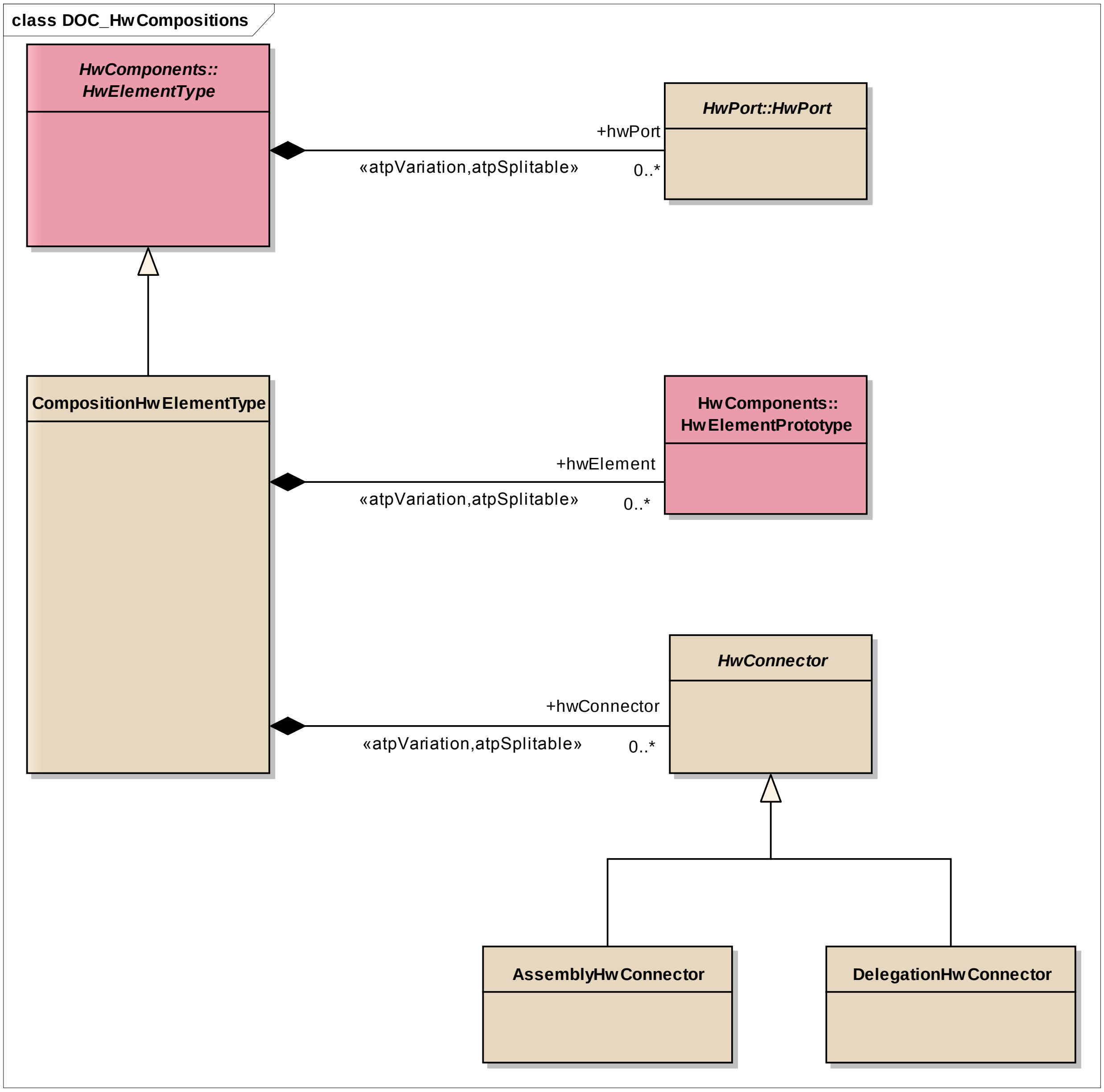


Figure: 8

Element "AssemblyHwConnector"

Parent Package: HwCompositions

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: AssemblyHwConnector. **Target:** HwConnector.	

Element "CompositionHwElementType"

Parent Package: HwCompositions

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: CompositionHwElementType. **Target:** HwElementType.	
	Source: **0..*** HwElementPrototype.hwElement **Target:** CompositionHwElementType.	
	Source: **0..*** HwConnector.hwConnector **Target:** CompositionHwElementType.	

Element "DelegationHwConnector"

Parent Package: HwCompositions

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: DelegationHwConnector. **Target:** HwConnector.	

Element "HwConnector"

Parent Package: HwCompositions

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: DelegationHwConnector. **Target:** HwConnector.	
	Source: AssemblyHwConnector. **Target:** HwConnector.	
	Source: **0..*** HwConnector.hwConnector **Target:** CompositionHwElementType.	

Element "HwElementConnector"

Parent Package: HwCompositions

Stereotype: ,

Notes:

This meta-class represents the ability to connect two hardware elements.

The details of the connection can be refined by hwPinGroupConnection.

Relationships

Name	Source/Target	Notes
	Source: HwElementConnector. **Target:** **0..*** HwPinConnector.hwPinConnection	
	Source: HwElementType. **Target:** **0..*** HwElementConnector.hwElementConnection	
	Source: HwElementConnector. **Target:** **2** HwElementType.hwElement	
	Source: HwElementConnector. **Target:** **0..*** HwPinGroupConnector.hwPinGroupConnection	

Element "HwPinConnector"

Parent Package: HwCompositions

Stereotype: ,

Notes:

This meta-class represents the ability to connect two pins.

Relationships

Name	Source/Target			Notes
	Source:		HwElementConnector.	
	Target:	**0..***	HwPinConnector.hwPinConnection	
	Source:		HwPinGroupConnector.	
	Target:	**0..***	HwPinConnector.hwPinConnection	
	Source:		HwPinConnector.	
	Target:	**2**	HwPin.hwPin	

Element "HwPinGroupConnector"

Parent Package: HwCompositions

Stereotype: ,

Notes:

This meta-class represents the ability to connect two pin groups.

Relationships

Name	Source/Target			Notes
	Source:		HwPinGroupConnector.	
	Target:	**2**	HwPinGroup.hwPinGroup	
	Source:		HwElementConnector.	
	Target:	**0..***	HwPinGroupConnector.hwPinGroupConnection	
	Source:		HwPinGroupConnector.	
	Target:	**0..***	HwPinConnector.hwPinConnection	

Package "HwPort"

Type of Package: **Package**

Parent Package: EcuResourceTemplate

Notes:

Diagram "DOC_HwPort"

Notes:

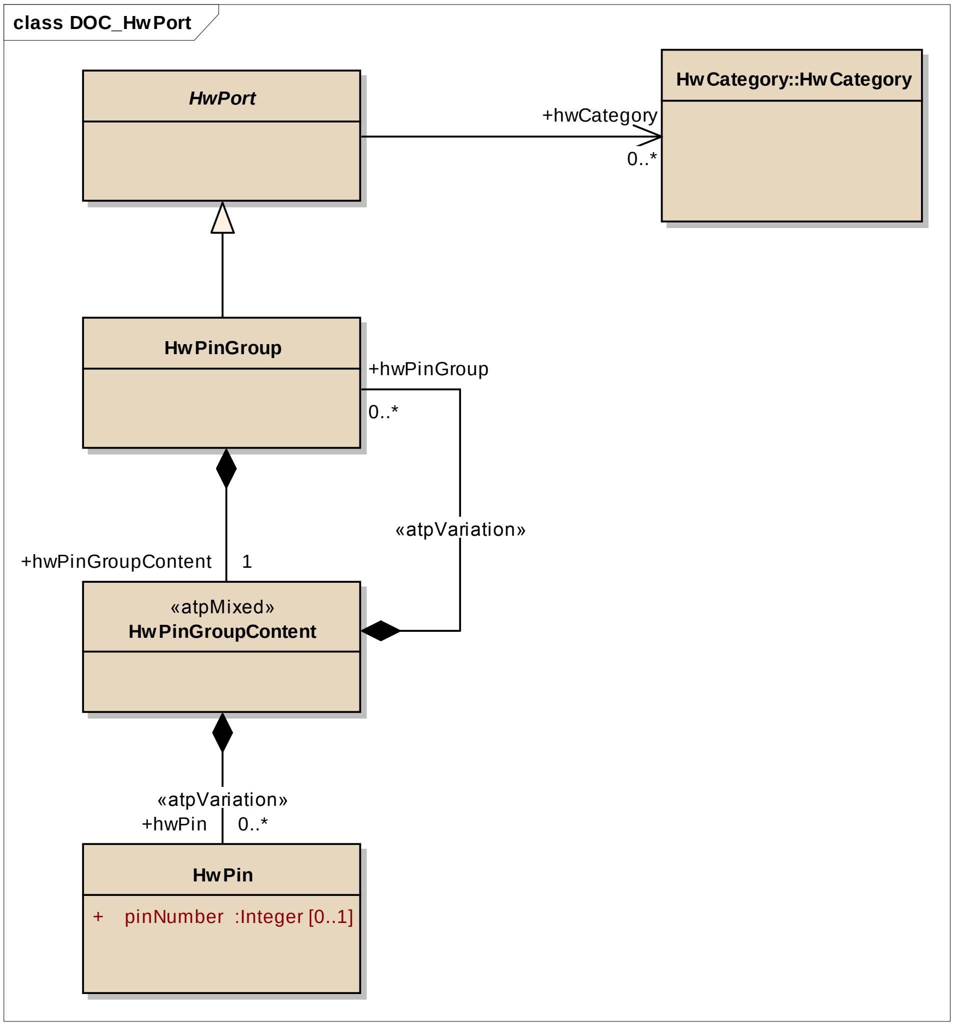


Figure: 9

Element "HwPin"

Parent Package: HwPort

Stereotype: ,

Notes:

This meta-class represents the possibility to describe a hardware pin.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	pinNumber	Integer			0	0	0		This attribute contains the physical pin number.

Relationships

Name	Source/Target	Notes
	Source: HwPinGroupContent. **Target:** **0..*** HwPin.hwPin	
	Source: HwPinConnector. **Target:** **2** HwPin.hwPin	

Element "HwPinGroup"

Parent Package: HwPort

Stereotype: ,

Notes:

This meta-class represents the ability to describe groups of pins which are used to connect hardware elements. This group acts as a bundle of pins. Thereby they allow to describe high level connections. Pin groups can even be nested.

Relationships

Name	Source/Target	Notes
	Source: HwPinGroup. **Target:** **1** HwPinGroupContent.hwPinGroupContent	
	Source: HwPinGroupConnector. **Target:** **2** HwPinGroup.hwPinGroup	
	Source: HwPinGroup. **Target:** HwPort.	
	Source: HwPinGroupContent. **Target:** **0..*** HwPinGroup.hwPinGroup	
	Source: HwElementType. **Target:** **0..*** HwPinGroup.hwPinGroup	

Element "HwPinGroupContent"

Parent Package: HwPort

Stereotype: «atpMixed»,

Notes:

This meta-class specifies a mixture of hwPins and hwPinGroups.

Relationships

Name	Source/Target	Notes
	Source: HwPinGroup. **Target:** **1** HwPinGroupContent.hwPinGroupContent	
	Source: HwPinGroupContent. **Target:** **0..*** HwPin.hwPin	
	Source: HwPinGroupContent. **Target:** **0..*** HwPinGroup.hwPinGroup	

Element "HwPort"

Parent Package: HwPort

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: **0..*** HwPort.hwPort **Target:** HwElementType.	
	Source: HwPort. **Target:** **0..*** HwCategory.hwCategory	
	Source: HwPinGroup. **Target:** HwPort.	
	Source: **0..*** HwAttributeValue.hwAttributeValue **Target:** HwPort.	

Package "Change Request to EAST-ADL Meta-Model"

Type of Package: **Package**

Parent Package: Change Requests to External Meta-Models

Notes:

Diagram "**Change Requests to EAST-ADL**"

Notes:

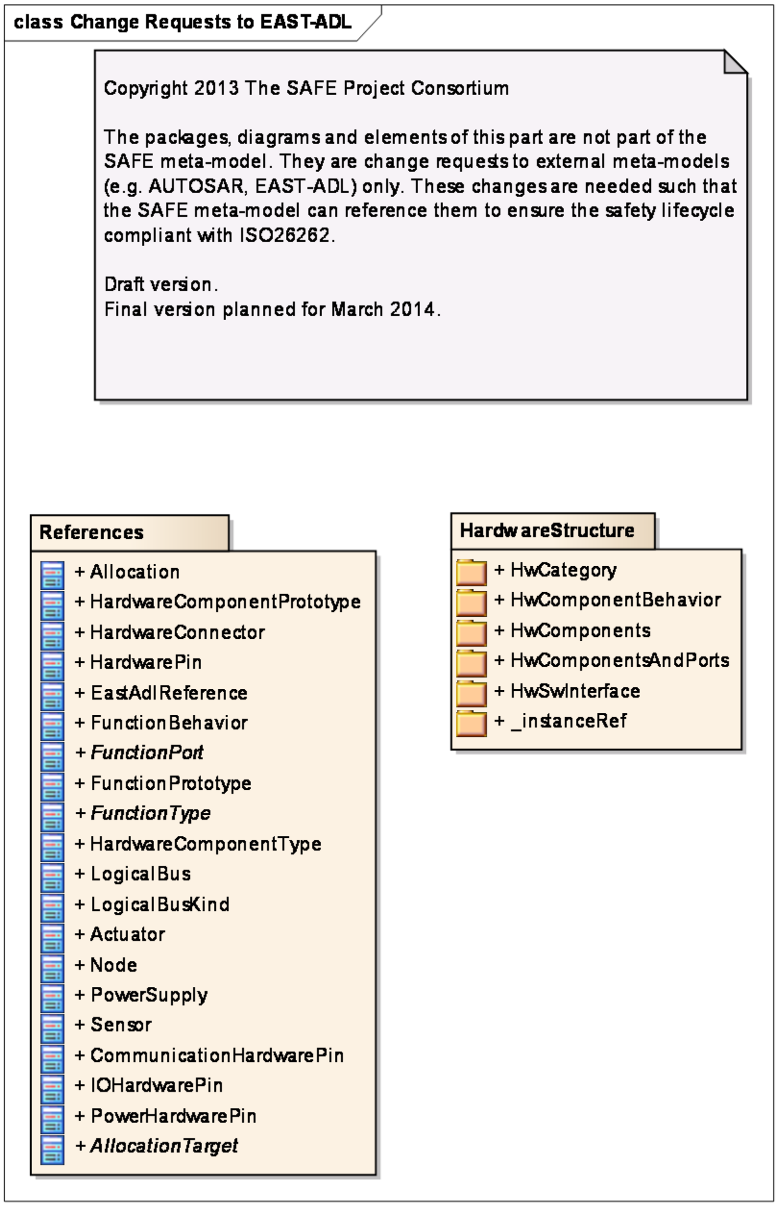


Figure: 10

Package "References"

Type of Package: **Package**

Parent Package: Change Request to EAST-ADL Meta-Model

Notes:

Diagram "EASTADLReferences"

Notes:

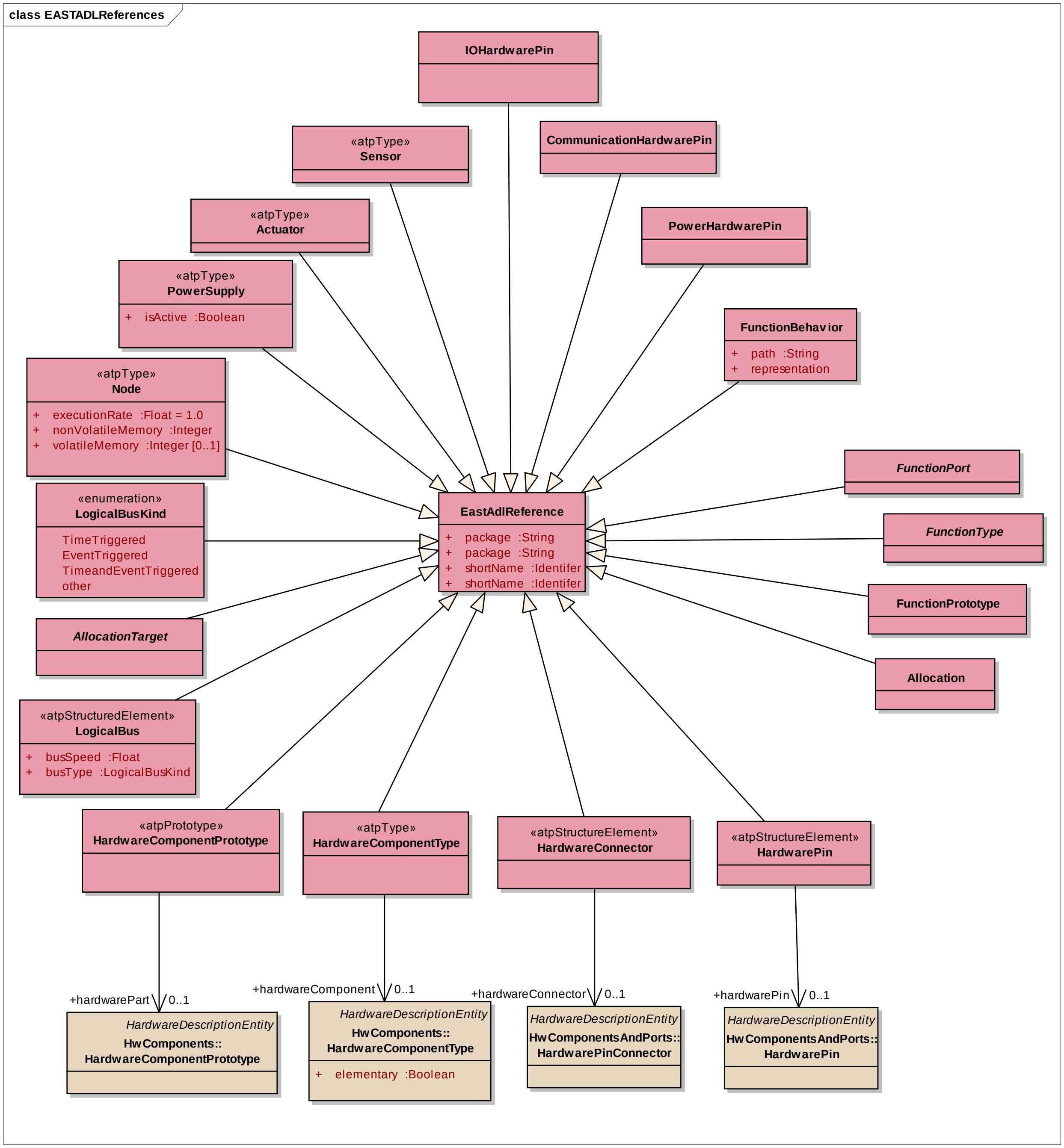


Figure: 11

Element "Allocation"

Parent Package: References

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: Allocation. **Target:** EastAdlReference.	
	Source: **0..*** HwSwInterface.hwswInterfaceAllocation **Target:** **1** Allocation.	

Element "EastAdlReference"

Parent Package: References

Stereotype: ,

Notes:

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	package	String							define packages path in same format used by EAST-ADL (identical to AUTOSAR) internal reference. ex: <UNIT-REF DEST="UNIT">/strictXSD_container/strictXSD_18</UNIT-REF>
	package	String							
	shortName	Identifer							define short name in same format used by EAST-ADL (identical to autosar) internal reference. ex: <UNIT-REF DEST="UNIT">/strictXSD_container/strictXSD_18</UNIT-REF>

	shortName	Identifer							

Relationships

Name	Source/Target	Notes
	Source: FunctionPrototype. **Target:** EastAdlReference.	
	Source: FunctionType. **Target:** EastAdlReference.	
	Source: FunctionPort. **Target:** EastAdlReference.	
	Source: HardwareComponentType. **Target:** EastAdlReference.	
	Source: Allocation. **Target:** EastAdlReference.	
	Source: AllocationTarget. **Target:** EastAdlReference.	
	Source: HardwarePin. **Target:** EastAdlReference.	
	Source: LogicalBusKind. **Target:** EastAdlReference.	
	Source: LogicalBus. **Target:** EastAdlReference.	
	Source: PowerHardwarePin. **Target:** EastAdlReference.	
	Source: Sensor. **Target:** EastAdlReference.	
	Source: IOHardwarePin. **Target:** EastAdlReference.	
	Source: Node. **Target:** EastAdlReference.	
	Source: HardwareComponentPrototype. **Target:** EastAdlReference.	
	Source: PowerSupply. **Target:** EastAdlReference.	

Name	Source/Target		Notes
	Source:	FunctionBehavior.	
	Target:	EastAdlReference.	
	Source:	Actuator.	
	Target:	EastAdlReference.	
	Source:	HardwareConnector.	
	Target:	EastAdlReference.	
	Source:	CommunicationHardwarePin.	
	Target:	EastAdlReference.	

Element "FunctionBehavior"

Parent Package: References

Stereotype: ,

Notes:

FunctionBehavior represents the behavior of a particular FunctionType or HardwareComponentType - referred to by the association to FunctionType or HardwareComponentType. What is meant by behavior is a transfer function performing some data computation (in case of FlowPort interaction for FunctionType or HardwarePin/Port interaction for HardwareType) or an operation that can be called by another function (in case of ClientServer interaction for FunctionType only). The representation property indicates the kind of representation used to describe the behavior (see FunctionBehaviorKind). The representation itself (e.g., defined in an external model file) is identified by a URL String in the path property. If the representation is provided in the same model file as the system itself, the path property is not used. It is merely a placeholder with the purpose of containing information about and links to the external behavioral model.

FunctionBehavior may refer to execution modes by the association to the element Mode. This is not mandatory; however, when provided, the relation indicates the list of execution Modes in which the FunctionBehavior can potentially be executed (see element Mode).

The triggering of a FunctionBehavior is unknown to the behavior. It is defined by FunctionTriggers (see this element).

Note that the association between or FunctionBehavior and HardwareComponentType or FunctionType is specified as a one-way navigable link from FunctionBehavior to HardwareComponentType or FunctionType: what this means is that the EAST-ADL language specification does not require that a HardwareComponentType or FunctionType be aware of the FunctionBehavior it is assigned to. Only the navigation from behavior to function is mandatory; the implementation of a reverse link might however be provided depending on the tool support.

Although each FunctionBehavior can refer to at most one HardwareComponentType or FunctionType, note that several FunctionBehaviors can refer to the same HardwareComponentType or FunctionType. In this case, when a HardwareComponentType or FunctionType has several behaviors, only one behavior shall be active at any given time instant, i.e., no concurrent behaviors are allowed in EAST-ADL functions. For instance we cannot have one active behavior in Simulink and one in Modelica. Both can be referenced in the same function, but at any given time, only one is executable. Conditions such as modes and variability must prevent two behaviors being potentially active.

Semantics:

The representation provided to a FunctionBehavior follows the semantics of the behavioral representation used (for instance SIMULINK, ASCET, etc.). However, in relation to the EAST-ADL model, the FunctionBehavior has synchronous execution semantics:

1. Read inputs from input ports
2. Execute Behavior with fixed inputs (run to completion)
3. Provide outputs to output ports

The data transfer between the EAST-ADL ports and the FunctionBehavior is representation specific and considered part of the execution of the FunctionBehavior.

Notation:

FunctionBehavior appears as a solid-outline rectangle with "Behavior" at the top right. The rectangle contains the name.

Extension: Behavior

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	path	String							The path to the file or model entity containing the behavior.
	representation								The type of representation used

									to describe the behavior.

Relationships

Name	Source/Target	Notes
	Source: **0..1** FunctionBehavior.hwComponentType **Target:** HardwareComponentType.	
	Source: FunctionBehavior. **Target:** EastAdlReference.	

Element "FunctionPort"

Parent Package: References

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: FunctionPort. **Target:** EastAdlReference.	
	Source: ***** FunctionPort.port **Target:** **1** FunctionType.	
	Source: FunctionPortInFunctionTypeHwAbstrRef. **Target:** **1** FunctionPort.targetFunctionPort	
	Source: HwAbstractionFunction. **Target:** **0..1** FunctionPort.functionPort	

Element "FunctionPrototype"

Parent Package: References

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: FunctionPrototype. **Target:** EastAdlReference.	
	Source: FunctionPortInFunctionTypeHwAbstrRef. **Target:** **1** FunctionPrototype.contextFunctionPrototype	
	Source: FunctionPrototype. **Target:** FunctionType.	

Element "FunctionType"

Parent Package: References

Stereotype: ,

Notes:

Relationships

Name	Source/Target	Notes
	Source: FunctionType. **Target:** EastAdlReference.	
	Source: FunctionPrototype. **Target:** FunctionType.	
	Source: HwAbstractionFunction. **Target:** FunctionType.	
	Source: FunctionPortInFunctionTypeHwAbstrRef. **Target:** **1** FunctionType.baseFunctionType	
	Source: * FunctionPort.port **Target:** **1** FunctionType.	

Element "HardwareComponentPrototype"

Parent Package: References

Stereotype: «atpPrototype»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: HardwareComponentPrototype. **Target:** **0..1** HardwareComponentPrototype.hardwarePart	
	Source: HardwareComponentPrototype. **Target:** EastAdlReference.	

Element "HardwareComponentType"

Parent Package: References

Stereotype: «atpType»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: HardwareComponentType. **Target:** EastAdlReference.	
	Source: HardwareComponentType. **Target:** **0..1** HardwareComponentType.hardwareComponent	

Element "HardwareConnector"

Parent Package: References

Stereotype: «atpStructureElement»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: HardwareConnector. **Target:** **0..1** HardwarePinConnector.hardwareConnector	
	Source: HardwareConnector. **Target:** EastAdlReference.	

Element "HardwarePin"

Parent Package: References

Stereotype: «atpStructureElement»,

Notes:

Relationships

Name	Source/Target	Notes
	Source: HardwarePin. **Target:** **0..1** HardwarePin.hardwarePin	
	Source: HardwarePin. **Target:** EastAdlReference.	

Element "LogicalBus"

Parent Package: References

Stereotype: «atpStructuredElement»,

Notes:

The LogicalBus represents logical communication channels. It serves as an allocation target for connectors, i.e. the data exchanged between functions in the FunctionalDesignArchitecture.

Semantics:

The LogicalBus represents a logical connection that carries data from any sender to all receivers. Senders and receivers are identified by the wires of the LogicalBus, i.e. the associated HardwareConnectors. The available busSpeed represents the maximum amount of useful data that can be carried. The busSpeed has already deducted speed reduction resulting from frame overhead, timing effects, etc.

Extension:

Class

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	busSpeed	Float			0	0	0		The net bus speed in bits per second. Used to assess communication delay and schedulability on the bus. Note that scheduling details

									are not represented in the model.
	busType	LogicalBusKind			0	0	0		The type of bus scheduling assumed.

Relationships

Name	Source/Target	Notes
	Source: LogicalBus. **Target:** * HardwarePinConnector.wire	
	Source: * LogicalBus.bus **Target:** HardwareComponentType.	
	Source: LogicalBus. **Target:** AllocationTarget.	
	Source: LogicalBus. **Target:** EastAdlReference.	

Element "LogicalBusKind"

Parent Package: References

Stereotype: «enumeration»,

Notes:

LogicalBusKind is an enumeration type representing different kinds of busses.

Semantics:

LogicalBusKind represents the kind of LogicalBus as given by the definition of the respective Enumeration Literal.

Extension:

Enumeration, no extension.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	TimeTriggered				0	0	0		Bus is time-triggered
	EventTriggered				0	0	0		Bus is event-triggered
	TimeandEventTriggered				0	0	0		Bus is both time and event-triggered

	other				0	0	0		Another type of bus communication

Relationships

Name	Source/Target	Notes
	Source: LogicalBusKind. **Target:** EastAdlReference.	

Element "Actuator"

Parent Package: References

Stereotype: «atpType»,

Notes:

The Actuator is the element that represents electrical actuators, such as valves, motors, lamps, brake units, etc. Non-electrical actuators such as the engine, hydraulics, etc. are considered part of the plant model (environment). Plant models are not part of the Hardware Design Architecture.

Semantics:

The Actuator metaclass represents the physical and electrical aspects of actuator hardware. The logical aspect is represented by a HardwareFunctionType associated with the Actuator.

Notation:

Actuator is shown as a solid-outline rectangle with double vertical borders. The rectangle contains the name, and its ports or port groups on the perimeter.

Relationships

Name	Source/Target	Notes
	Source: Actuator. **Target:** HardwareComponentType.	
	Source: Actuator. **Target:** EastAdlReference.	

Element "Node"

Parent Package: References

Stereotype: «atpType»,

Notes:

Node represents the computer nodes of the embedded electrical/electronic system. Nodes consist of processor(s) and may be connected to sensors, actuators and other ECUs via a BusConnector.

Node denotes an electronic control unit that acts as a computing element executing Functions. In case a single CPU ECU is represented, it is sufficient to have a single, non-hierarchical Node.

Semantics:

The Node element represents an ECU, i.e. an Electronic Control Unit, and an allocation target of FunctionPrototypes.

The Node executes its allocated FunctionPrototypes at the specified executionRate. The executionRate denotes how many execution seconds of an allocated functionPrototype´s execution time are processed in each real-time second. Actual execution time is thus found by dividing the parameters of the ExecutionTimeConstraint with executionRate.

Example: If an ECU is 25% faster than a standard ECU (e.g., in a certain context, execution times are given assuming a nominal speed of 100 MHz; our CPU is then 125 MHz), the executionRate is 1.25. An execution time of 5 ms would then become 4 ms on this ECU.

Notation:

Node is shown as a solid-outline rectangle with Node at the top right. The rectangle contains the name, and its ports or port groups on the perimeter.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	executionRate	Float			0	0	0	1.0	ExecutionRate is used to compute an approximate execution time. A nominal execution time divided by executionRate provides the actual execution time to be used e.g. for timing analysis in feasibility studies.
	nonVolatileMemory	Integer			0	0	0		The size in Bytes of the Node's Non-Volatile memory (ROM,

									NRAM, EPROM, etc.).
	volatileMemory	Integer			0	0	0		The size in Bytes of the Node's Volatile memory (RAM)

Relationships

Name	Source/Target	Notes
	Source: Node. **Target:** HardwareComponentType.	
	Source: Node. **Target:** EastAdlReference.	

Element "PowerSupply"

Parent Package: References

Stereotype: «atpType»,

Notes:

PowerSupply represents a hardware element that supplies power.

Semantics:

PowerSupply denotes a power source that may be active (e.g., a battery) or passive (main relay).

Notation:

PowerSupply is shown as a solid-outline rectangle with "PWR" at the top right. The rectangle contains the name, and its ports or port groups on the perimeter.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	isActive	Boolean			0	0	0		Indicates if the PowerSupply is active or passive.

Relationships

Name	Source/Target	Notes
	Source: PowerSupply. **Target:** HardwareComponentType.	
	Source: PowerSupply.	

Name	Source/Target	Notes
	Target: EastAdlReference.	

Element "Sensor"

Parent Package: References

Stereotype: «atpType»,

Notes:

Sensor represents a hardware entity for digital or analog sensor elements. The Sensor is connected electrically to the electrical entities of the Hardware Design Architecture.

Semantics:

Sensor denotes an electrical sensor. The Sensor represents the physical and electrical aspects of sensor hardware. The logical aspect is represented by a HardwareFunctionType associated with the Sensor.

Notation:

Sensor is shown as an oval. The circle contains the name, and its ports or port groups on the perimeter.

Relationships

Name	Source/Target	Notes
	Source: Sensor. **Target:** HardwareComponentType.	
	Source: Sensor. **Target:** EastAdlReference.	

Element "CommunicationHardwarePin"

Parent Package: References

Stereotype: ,

Notes:

CommunicationHardwarePin represents an electrical connection point that can be used to define how the wire harness is logically defined.

Semantics:

The CommunicationHardwarePin represents the hardware connection point of a communication bus.

Depending on modeling style, one or two pins may be defined for a dual-wire bus.

Notation:

CommunicationHardwarePin is shown as a solid square with a C inside. Its name may appear outside the square.

Relationships

Name	Source/Target	Notes
	Source: CommunicationHardwarePin. **Target:** HardwarePin.	
	Source: CommunicationHardwarePin. **Target:** EastAdlReference.	

Element "IOHardwarePin"

Parent Package: References

Stereotype: ,

Notes:

IOHardwarePin represents an electrical connection point for digital or analog I/O.

Semantics:

The IOHardwarePin represents an electrical pin or connection point.

Notation:

IOHardwarePin is shown as a solid square with an IO inside. Its name may appear outside the square.

Relationships

Name	Source/Target	Notes
	Source: IOHardwarePin. **Target:** HardwarePin.	
	Source: IOHardwarePin. **Target:** EastAdlReference.	

Element "PowerHardwarePin"

Parent Package: References

Stereotype: ,

Notes:

PowerHardwarePin represents a pin that is primarily intended for power supply, either providing or consuming energy.

Semantics:

A PowerHardwarePin is primarily intended to be a power supply. The direction attribute of the pin defines whether it is providing or consuming energy.

Notation:

PowerHardwarePin is shown as a solid square with PWR inside. Its name may appear outside the square.

Relationships

Name	Source/Target	Notes
	Source: PowerHardwarePin. **Target:** HardwarePin.	
	Source: PowerHardwarePin. **Target:** EastAdlReference.	

Element "AllocationTarget"

Parent Package: References

Stereotype: ,

Notes:

The AllocationTarget is a superclass for elements to which AllocateableElements can be allocated.

Semantics:

An AllocationTarget is a resource element in the Hardware Design Architecture which may host functional behaviors in the Functional Design Architecture.

Extension: abstract, no extension

Relationships

Name	Source/Target	Notes
	Source: AllocationTarget. **Target:** EastAdlReference.	
	Source: LogicalBus.	

Name	Source/Target		Notes
	Target:	AllocationTarget.	
	Source:	HardwareComponentPrototype.	
	Target:	AllocationTarget.	

Package "HardwareStructure"

Type of Package: **Package**

Parent Package: Change Request to EAST-ADL Meta-Model

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.

Diagram "HardwareModeling"

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.

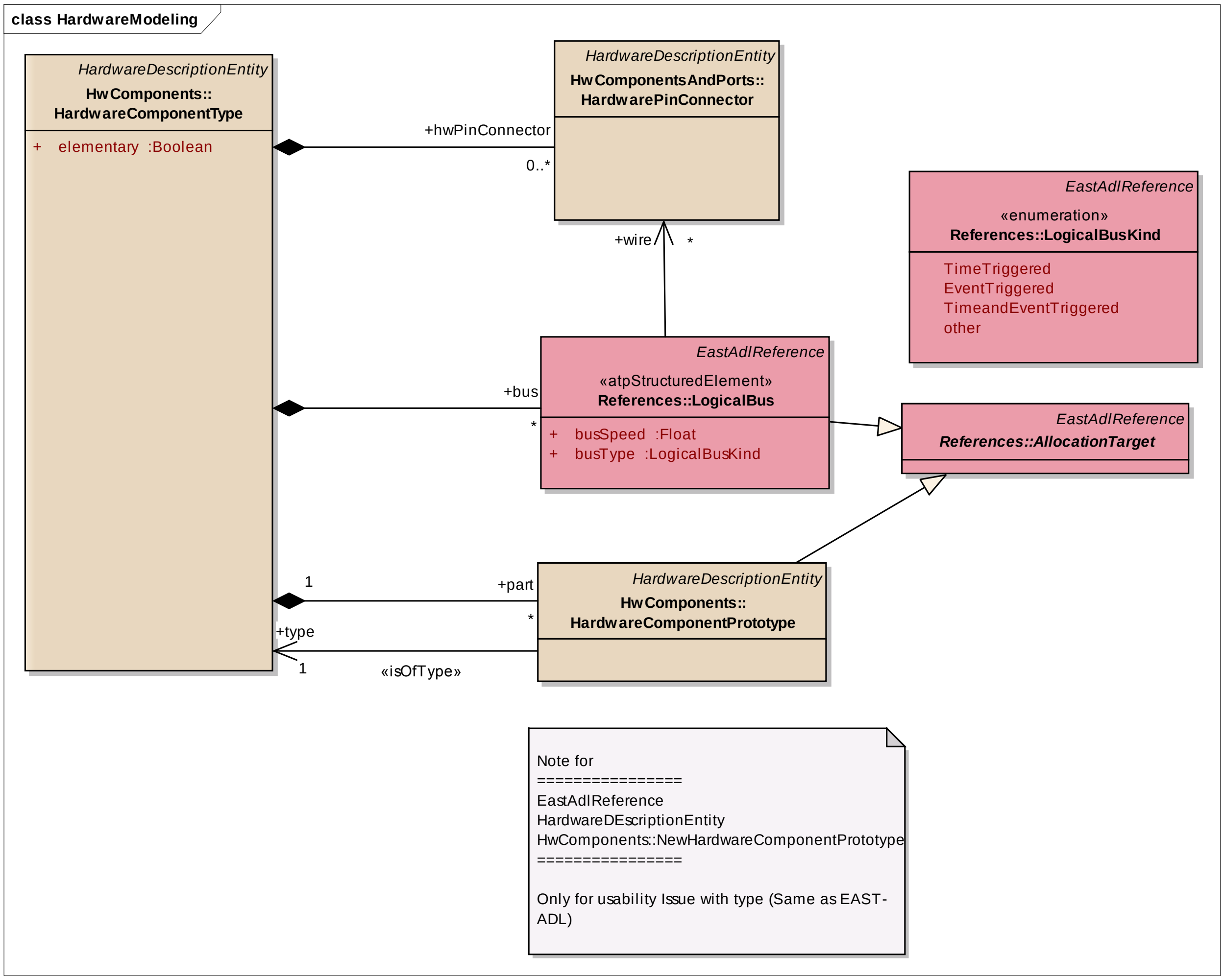


Figure: 12

Element "<anonymous>"

Parent Package: HardwareStructure

Stereotype: ,

Notes:

Note for

================

EastAdlReference

HardwareDEscriptionEntity

HwComponents::NewHardwareComponentPrototype

================

Only for usability Issue with type (Same as EAST-ADL)

Package "HwCategory"

Type of Package: **Package**

Parent Package: HardwareStructure

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.

Diagram "DOC_HwCategory"

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.

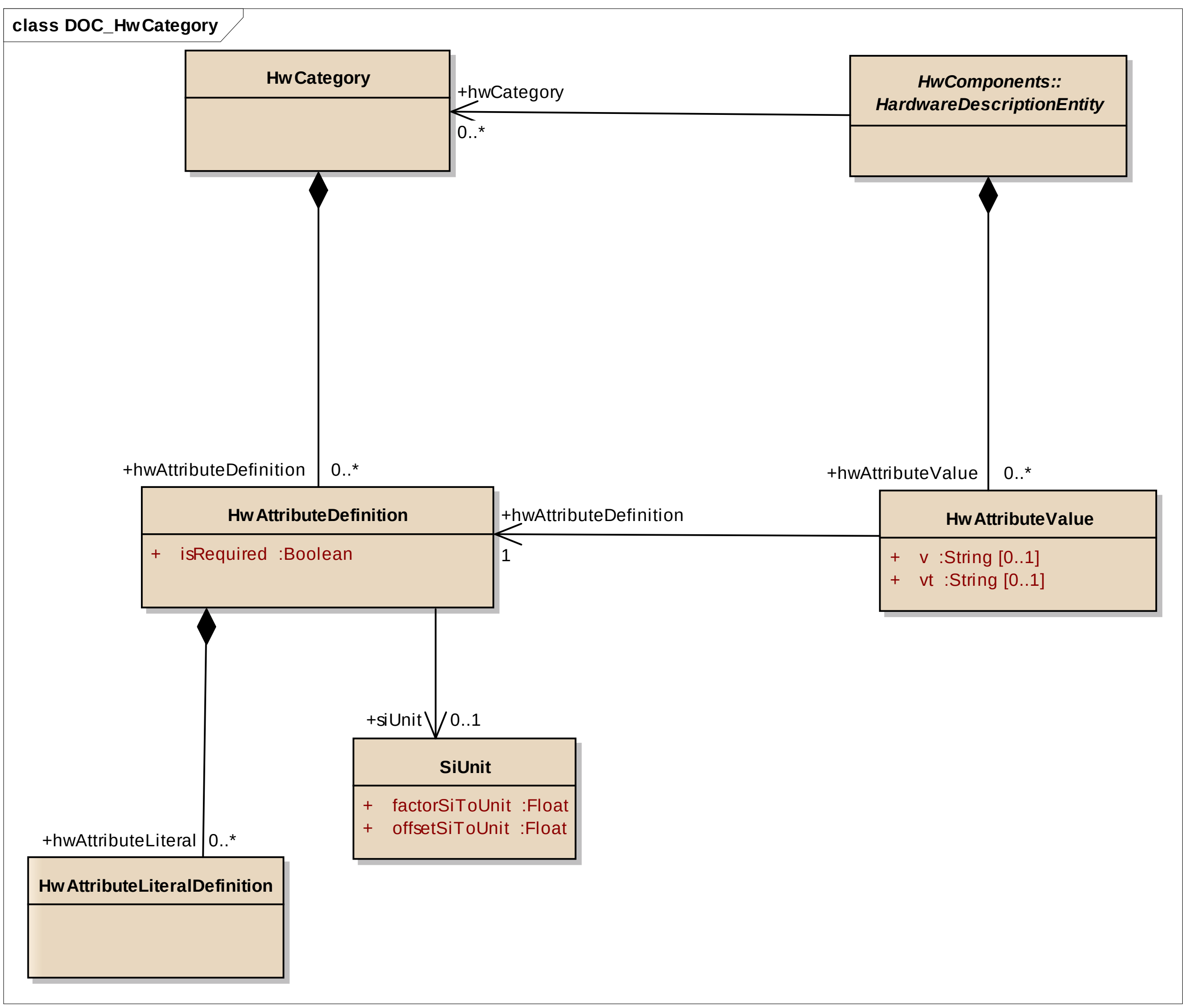


Figure: 13

Element "HwAttributeDefinition"

Parent Package: HwCategory

Stereotype: ,

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

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	isRequired	Boolean			0	0	0		This attribute specifies if the defined attribute value is required to be provided.

Relationships

Name	Source/Target	Notes
	Source: HwAttributeDefinition. **Target:** **0..1** SiUnit.siUnit	This class SiUnit implement the a 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)

Name	Source/Target	Notes
		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
	Source: **0..*** HwAttributeDefinition.hwAttributeDefinition **Target:** HwCategory.	
	Source: HwAttributeValue. **Target:** **1** HwAttributeDefinition.hwAttributeDefinition	
	Source: **0..*** HwAttributeLiteralDefinition.hwAttributeLiteral **Target:** HwAttributeDefinition.	

Element "HwAttributeLiteralDefinition"

Parent Package: HwCategory

Stereotype: ,

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

Relationships

Name	Source/Target	Notes
	Source: **0..*** HwAttributeLiteralDefinition.hwAttributeLiteral **Target:** HwAttributeDefinition.	

Element "HwAttributeValue"

Parent Package: HwCategory

Stereotype: ,

Notes:

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

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	v	String			0	0	0		This represents a textual hardware attribute value.
	vt	String			0	0	0		This represents a numerical hardware attribute value.

Relationships

Name	Source/Target	Notes
	Source: **0..*** HwAttributeValue.hwAttributeValue **Target:** HardwarePort.	
	Source: **0..*** HwAttributeValue.hwAttributeValue **Target:** HardwarePin.	
	Source: **0..*** HwAttributeValue.hwAttributeValue **Target:** HardwareDescriptionEntity.	
	Source: **0..*** HwAttributeValue.hwAttributeValue **Target:** HardwareComponentPrototype.	
	Source: HwAttributeValue. **Target:** **1** HwAttributeDefinition.hwAttributeDefinition	

Element "HwCategory"

Parent Package: HwCategory

Stereotype: ,

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

Relationships

Name	Source/Target	Notes
	Source: HardwarePort. **Target:** HwCategory.	
	Source: **0..*** HwAttributeDefinition.hwAttributeDefinition **Target:** HwCategory.	
	Source: HardwareDescriptionEntity. **Target:** **0..*** HwCategory.hwCategory	
	Source: HardwarePin. **Target:** HwCategory.	
	Source: HardwareComponentType. **Target:** **0..*** HwCategory.hwCategory	

Element "SiUnit"

Parent Package: HwCategory

Stereotype: ,

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

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	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.

Relationships

Name	Source/Target	Notes
	Source: HwAttributeDefinition. **Target:** **0..1** SiUnit.siUnit	This class SiUnit implement the a 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 +

Name	Source/Target	Notes
		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

Package "HwComponentBehavior"

Type of Package: **Package**

Parent Package: HardwareStructure

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 developement as SystemC, Modelica, VHDL-AMS or Verilog-AMS.

Diagram "**DOC_HwComponentBehavior**"

Notes:

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

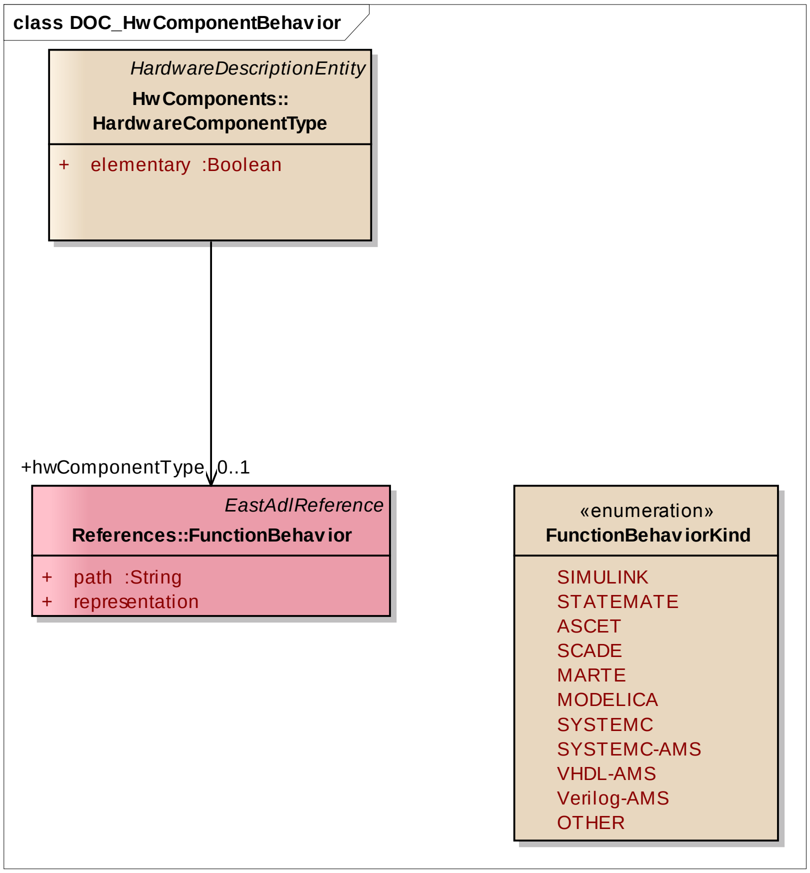


Figure: 14

Element "FunctionBehaviorKind"

Parent Package: HwComponentBehavior

Stereotype: «enumeration»,

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 an pin interface).

Extension:

Enumeration, no extension.

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	SIMULINK								
	STATEMATE								
	ASCET								
	SCADE								
	MARTE								
	MODELICA								
	SYSTEMC								
	SYSTEMC-AMS								
	VHDL-AMS								
	Verilog-AMS								
	OTHER								

Package "HwComponents"

Type of Package: **Package**

Parent Package: HardwareStructure

Notes:

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

Diagram "DOC_HwComponents"

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.

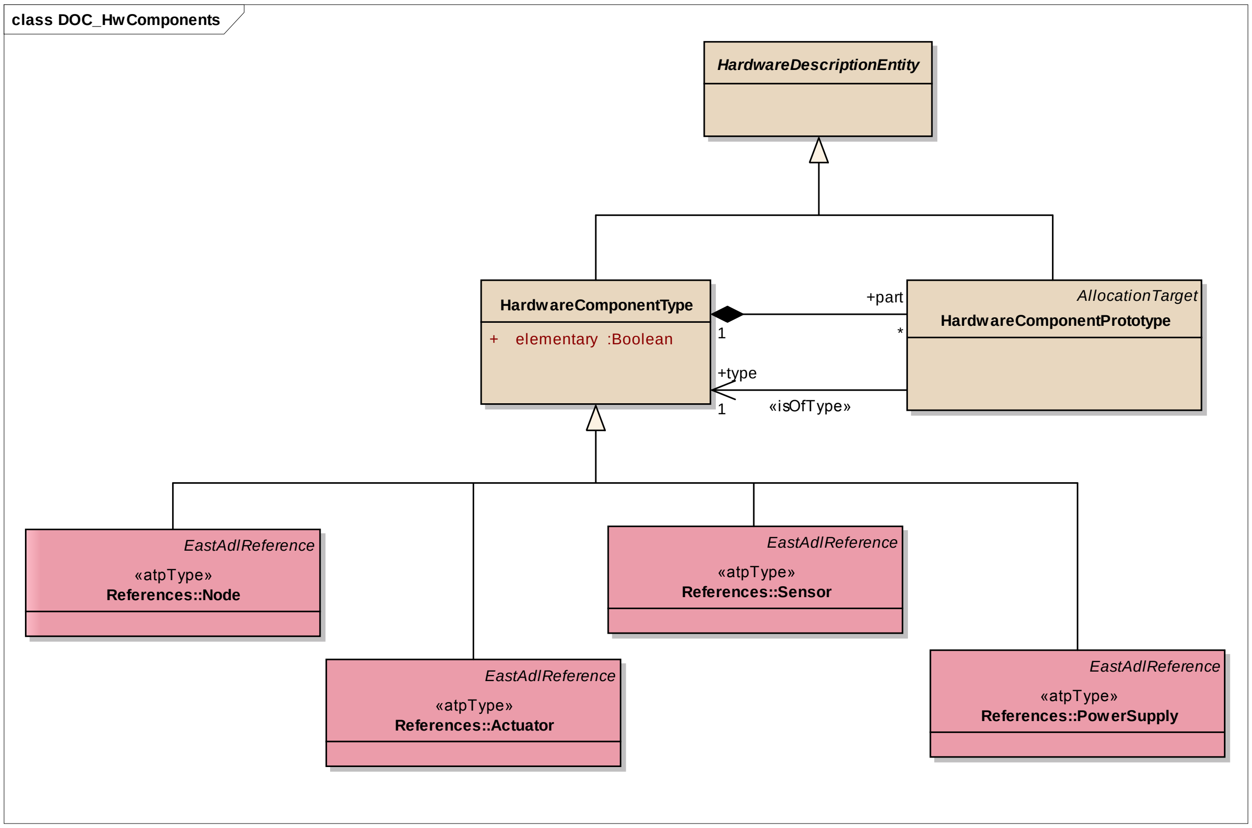


Figure: 15

Element "HardwareDescriptionEntity"

Parent Package: HwComponents

Stereotype: ,

Notes:

This abstract class describes any hardware entity for further use.

Semantic:

none

Relationships

Name	Source/Target	Notes
	Source: HardwareComponentType. **Target:** HardwareDescriptionEntity.	
	Source: HardwarePort. **Target:** HardwareDescriptionEntity.	

Name	Source/Target	Notes
	Source: HardwarePinConnector. **Target:** HardwareDescriptionEntity.	
	Source: HwPortConnector. **Target:** HardwareDescriptionEntity.	
	Source: HardwareComponentPrototype. **Target:** HardwareDescriptionEntity.	
	Source: HardwarePin. **Target:** HardwareDescriptionEntity.	
	Source: HardwareDescriptionEntity. **Target:** **0..*** HwCategory.hwCategory	
	Source: **0..*** HwAttributeValue.hwAttributeValue **Target:** HardwareDescriptionEntity.	

Element "HardwareComponentPrototype"

Parent Package: HwComponents

Stereotype: ,

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

Relationships

Name	Source/Target	Notes
	Source: HardwareComponentPrototype.	

Name	Source/Target	Notes
	Target: 0..1 HardwareComponentPrototype.hardwarePart	
	Source: HardwarePinInHardwareTypeHwAbstrRef. **Target: 1** HardwareComponentPrototype.contextHardwareComponentPrototype	
	Source: HardwareComponentPrototype. **Target: 1** HardwareComponentType.type	
	Source: HwPortInComponentInstanceRef. **Target: 1** HardwareComponentPrototype.contextHwComponent	
	Source: HwPinInHwComponentInstanceRef. **Target: 1** HardwareComponentPrototype.contextHwComponent	
	Source: 1 HardwareComponentType. **Target: *** HardwareComponentPrototype.part	
	Source: HardwareComponentPrototype. **Target:** HardwareDescriptionEntity.	
	Source: 0..* HwAttributeValue.hwAttributeValue **Target:** HardwareComponentPrototype.	
	Source: HardwareComponentPrototype. **Target:** AllocationTarget.	

Element "HardwareComponentType"

Parent Package: HwComponents

Stereotype: ,

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 estart 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

Attributes

PK	Name	Type	Not Null	Unique	Len	Prec	Scale	Init	Notes
	elementary	Boolean			0	0	0		This parameter is used to define if a hardware component is further decomposed with parts.

Relationships

Name	Source/Target	Notes
	Source: HwPinInHwComponentInstanceRef. **Target:** **1** HardwareComponentType.baseHwComponent	
	Source: **0..*** HwPortConnector.hwPortConnector **Target:** HardwareComponentType.	
	Source: **0..*** HardwarePinConnector.hwPinConnector **Target:** HardwareComponentType.	
	Source: **0..1** HardwareComponentType. **Target:** **0..*** HardwarePort.hwPort	
	Source: HwPortInComponentInstanceRef. **Target:** **1** HardwareComponentType.baseHwComponent	
	Source: **1** HardwareComponentType. **Target:** **0..*** HardwarePin.hwPin	
	Source: Actuator. **Target:** HardwareComponentType.	
	Source: PowerSupply. **Target:** HardwareComponentType.	
	Source: Node. **Target:** HardwareComponentType.	
	Source: HardwareComponentPrototype. **Target:** **1** HardwareComponentType.type	

Name	Source/Target	Notes
	Source: **1** HardwareComponentType. **Target:** ***** HardwareComponentPrototype.part	
	Source: Sensor. **Target:** HardwareComponentType.	
	Source: HardwareComponentType. **Target:** HardwareDescriptionEntity.	
	Source: HardwareComponentType. **Target:** **0..*** HwCategory.hwCategory	
	Source: **0..1** FunctionBehavior.hwComponentType **Target:** HardwareComponentType.	
	Source: HardwareComponentType. **Target:** **0..1** HardwareComponentType.hardwareComponent	
	Source: ***** LogicalBus.bus **Target:** HardwareComponentType.	
	Source: HardwarePinInHardwareTypeHwAbstrRef. **Target:** **1** HardwareComponentType.baseHardwareComponentType	

Package "HwComponentsAndPorts"

Type of Package: **Package**

Parent Package: HardwareStructure

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.

Diagram "**DOC_HwComponentsAndPorts**"

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.

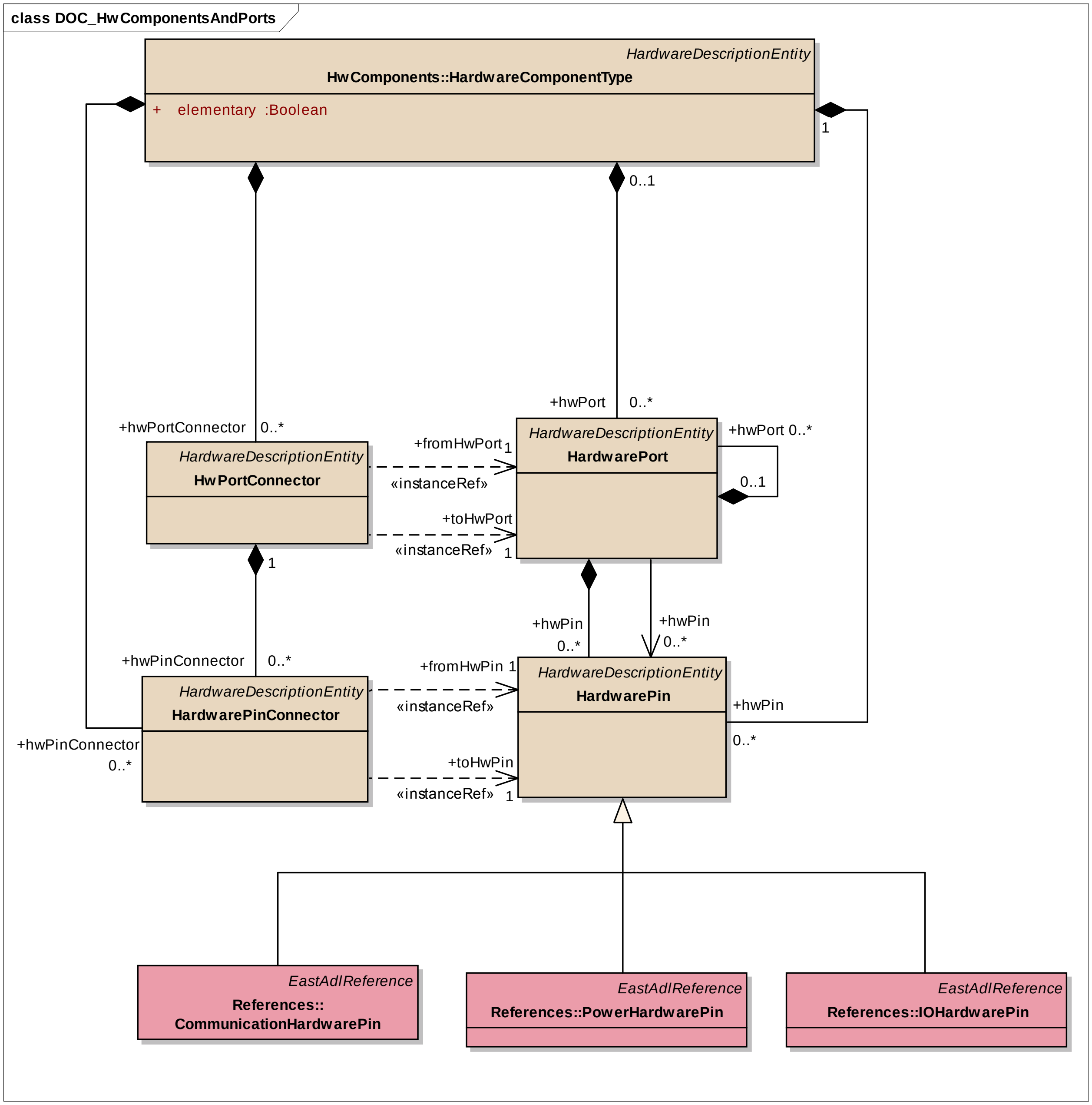


Figure: 16

Element "HwPortConnector"

Parent Package: HwComponentsAndPorts

Stereotype: ,

Notes:

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

Semantics:

The connector joins the two referenced ports electrically.

Extension:

Connector

Relationships

Name	Source/Target	Notes
	Source: HwPortConnector. **Target:** **1** HardwarePort.toHwPort	
	Source: **0..1** HwPortInComponentInstanceRef. **Target:** HwPortConnector.	
	Source: HwPortConnector. **Target:** **1** HardwarePort.fromHwPort	
	Source: **1** HwPortConnector. **Target:** **0..*** HardwarePinConnector.hwPinConnector	
	Source: **0..*** HwPortConnector.hwPortConnector **Target:** HardwareComponentType.	
	Source: HwPortConnector. **Target:** HardwareDescriptionEntity.	

Element "HardwarePinConnector"

Parent Package: HwComponentsAndPorts

Stereotype: ,

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

Relationships

Name	Source/Target	Notes
	Source: HardwareConnector. **Target:** **0..1** HardwarePinConnector.hardwareConnector	
	Source: **1** HwPortConnector. **Target:** **0..*** HardwarePinConnector.hwPinConnector	
	Source: **0..1** HwPinInHwComponentInstanceRef. **Target:** HardwarePinConnector.	
	Source: HardwarePinConnector. **Target:** **1** HardwarePin.fromHwPin	
	Source: **0..*** HardwarePinConnector.hwPinConnector **Target:** HardwareComponentType.	
	Source: HardwarePinConnector. **Target:** HardwareDescriptionEntity.	
	Source: LogicalBus. **Target:** ***** HardwarePinConnector.wire	
	Source: HardwarePinConnector. **Target:** **1** HardwarePin.toHwPin	

Element "HardwarePin"

Parent Package: HwComponentsAndPorts

Stereotype: ,

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

Relationships

Name	Source/Target	Notes

Name	Source/Target	Notes
	Source: HardwarePin. **Target:** **0..1** HardwarePin.hardwarePin	
	Source: HardwarePinInHardwareTypeHwAbstrRef. **Target:** **1** HardwarePin.targetHardwarePin	
	Source: PowerHardwarePin. **Target:** HardwarePin.	
	Source: IOHardwarePin. **Target:** HardwarePin.	
	Source: HwPinInHwComponentInstanceRef. **Target:** **1** HardwarePin.targetHwPin	
	Source: HwAbstractionFunction. **Target:** **0..1** HardwarePin.hardwarePin	
	Source: **0..*** HardwarePin.hwPin **Target:** HardwarePort.	
	Source: HardwarePort. **Target:** **0..*** HardwarePin.hwPin	
	Source: CommunicationHardwarePin. **Target:** HardwarePin.	
	Source: HardwarePinConnector. **Target:** **1** HardwarePin.fromHwPin	
	Source: **1** HardwareComponentType. **Target:** **0..*** HardwarePin.hwPin	
	Source: HardwarePin. **Target:** HardwareDescriptionEntity.	
	Source: HardwarePin. **Target:** HwCategory.	
	Source: **0..*** HwAttributeValue.hwAttributeValue **Target:** HardwarePin.	
	Source: HardwarePinConnector. **Target:** **1** HardwarePin.toHwPin	

Element "HardwarePort"

Parent Package: HwComponentsAndPorts

Stereotype: ,

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 is 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

Relationships

Name	Source/Target	Notes
	Source: HwPortConnector. **Target:** **1** HardwarePort.toHwPort	
	Source: **0..*** HardwarePin.hwPin **Target:** HardwarePort.	
	Source: HwPortInComponentInstanceRef. **Target:** **1** HardwarePort.targetHwPort	
	Source: HardwarePort. **Target:** **0..*** HardwarePin.hwPin	
	Source: **0..1** HardwarePort. **Target:** **0..*** HardwarePort.hwPort	
	Source: HwPortConnector.	

Name	Source/Target	Notes
	Target: **1** HardwarePort.fromHwPort	
	Source: **0..1** HardwareComponentType. **Target:** **0..*** HardwarePort.hwPort	
	Source: HardwarePort. **Target:** HardwareDescriptionEntity.	
	Source: HardwarePort. **Target:** HwCategory.	
	Source: **0..*** HwAttributeValue.hwAttributeValue **Target:** HardwarePort.	

Package "HwSwInterface"

Type of Package: **Package**

Parent Package: HardwareStructure

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.

Diagram "**HwSwInterface**"

Notes:

This class diagram represent the definition of the HwSWInterface. A software element is represented by a DesignFunction and a hardware element by a HarwdareComponent.

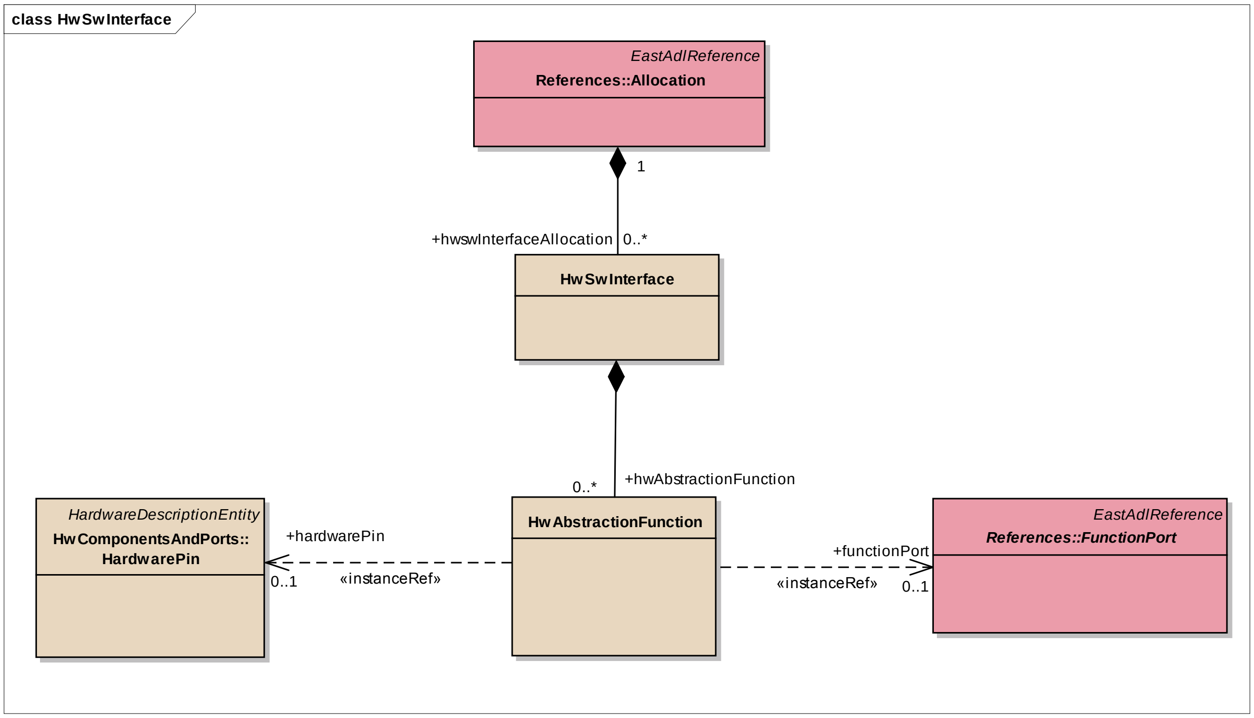


Figure: 17

Element "HwAbstractionFunction"

Parent Package: HwSwInterface

Stereotype: ,

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 to make 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 as 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.

Relationships

Name	Source/Target	Notes
	Source: 0..1 HardwarePinInHardwareTypeHwAbstrRef. **Target:** HwAbstractionFunction.	
	Source: 0..1 FunctionPortInFunctionTypeHwAbstrRef. **Target:** HwAbstractionFunction.	
	Source: 0..* HwAbstractionFunction.hwAbstractionFunction **Target:** HwSwInterface.	
	Source: HwAbstractionFunction. **Target:** 0..1 HardwarePin.hardwarePin	
	Source: HwAbstractionFunction. **Target:** FunctionType.	
	Source: HwAbstractionFunction. **Target:** 0..1 FunctionPort.functionPort	

Element "HwSwInterface"

Parent Package: HwSwInterface

Stereotype: ,

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 elements 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 HwAbstrationFunction.

Relationships

Name	Source/Target	Notes

Name	Source/Target			Notes
	Source:	**0..***	HwAbstractionFunction.hwAbstractionFunction	
	Target:		HwSwInterface.	
	Source:	**0..***	HwSwInterface.hwswInterfaceAllocation	
	Target:	**1**	Allocation.	

Package "_instanceRef"

Type of Package: **Package**

Parent Package: HardwareStructure

Notes:

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

Diagram "FunctionPortInFunctionType"

Notes:

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

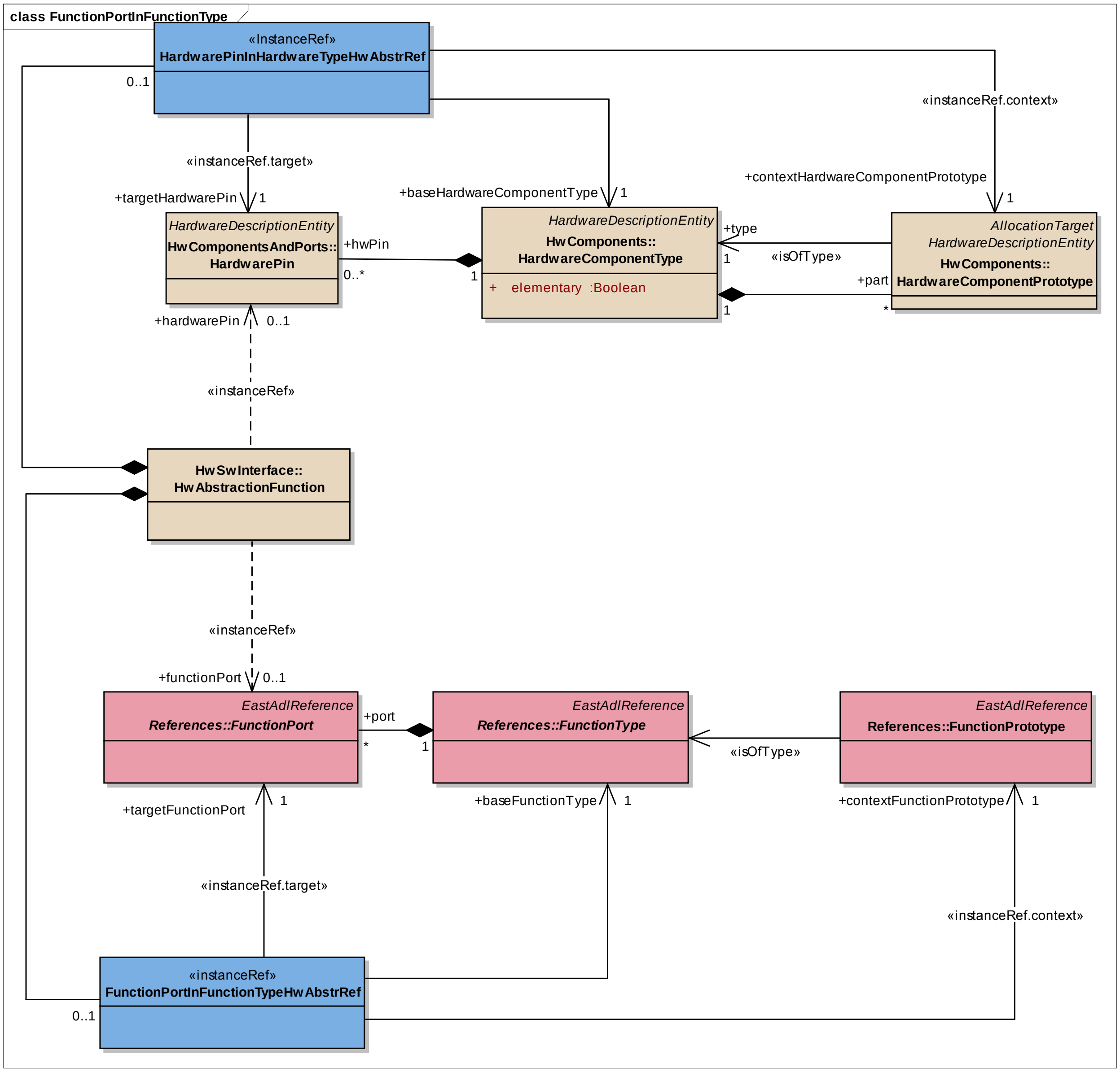


Figure: 18

Diagram "HwPinInHwComponentType"

Notes:

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

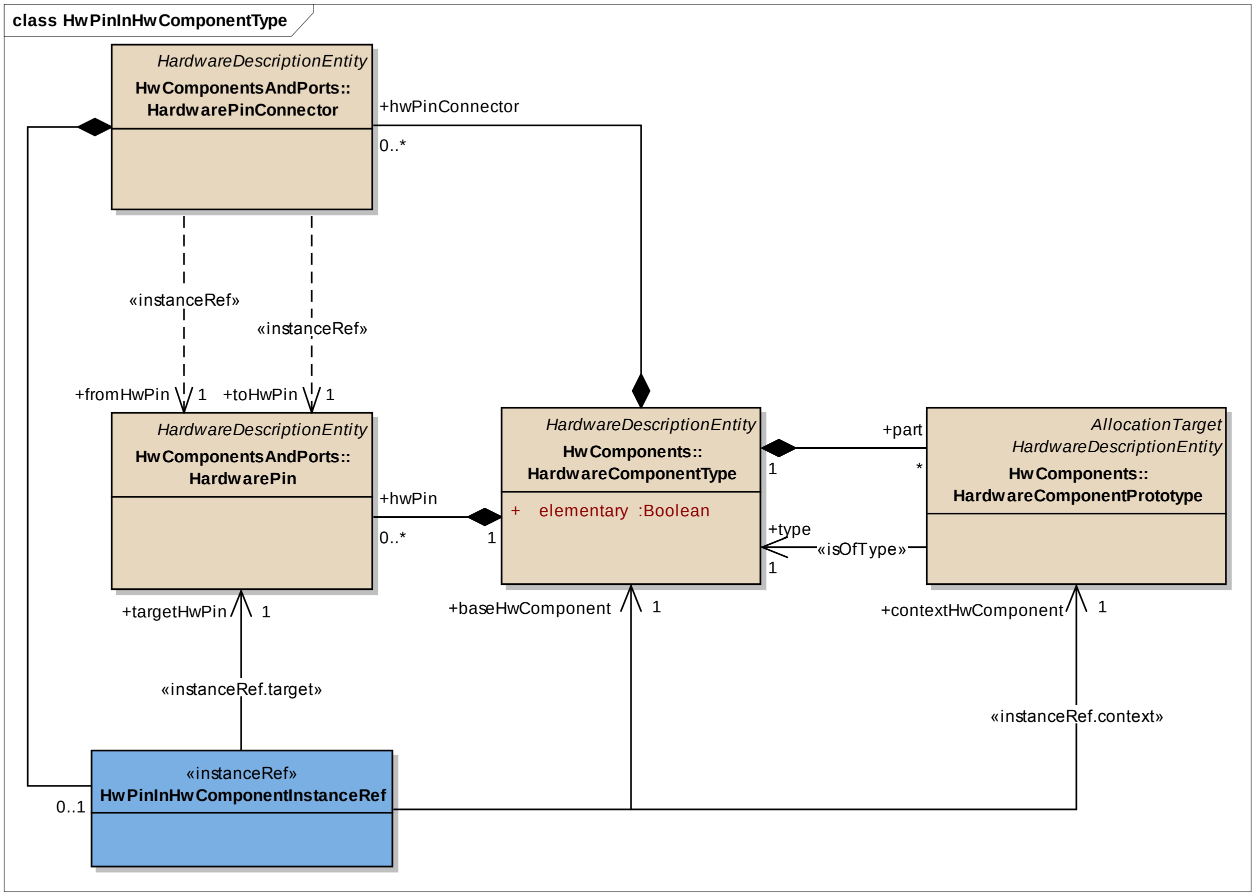


Figure: 19

Diagram "HwPortInHwComponentType"

Notes:

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

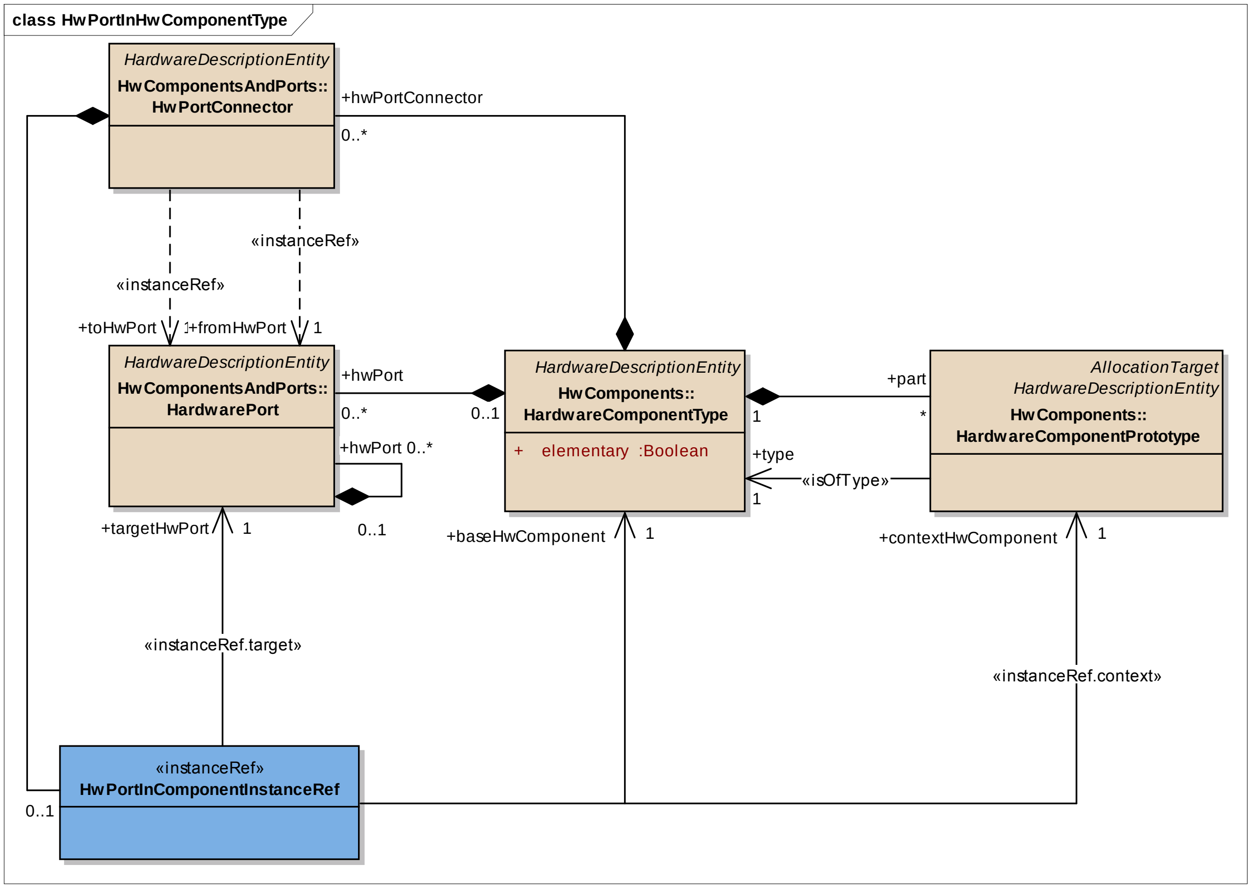


Figure: 20

Element "FunctionPortInFunctionTypeHwAbstrRef"

Parent Package: _instanceRef

Stereotype: «instanceRef»,

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).

Relationships

Name	Source/Target	Notes
	Source: **0..1** FunctionPortInFunctionTypeHwAbstrRef. **Target:** HwAbstractionFunction.	
	Source: FunctionPortInFunctionTypeHwAbstrRef. **Target:** **1** FunctionPrototype.contextFunctionPrototype	
	Source: FunctionPortInFunctionTypeHwAbstrRef.	

Name	Source/Target	Notes
	Target: **1** FunctionType.baseFunctionType	
	Source: FunctionPortInFunctionTypeHwAbstrRef. **Target:** **1** FunctionPort.targetFunctionPort	

Element "HardwarePinInHardwareTypeHwAbstrRef"

Parent Package: _instanceRef

Stereotype: «InstanceRef»,

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).

Relationships

Name	Source/Target	Notes
	Source: **0..1** HardwarePinInHardwareTypeHwAbstrRef. **Target:** HwAbstractionFunction.	
	Source: HardwarePinInHardwareTypeHwAbstrRef. **Target:** **1** HardwarePin.targetHardwarePin	
	Source: HardwarePinInHardwareTypeHwAbstrRef. **Target:** **1** HardwareComponentPrototype.contextHardwareComponentPrototype	
	Source: HardwarePinInHardwareTypeHwAbstrRef. **Target:** **1** HardwareComponentType.baseHardwareComponentType	

Element "HwPinInHwComponentInstanceRef"

Parent Package: _instanceRef

Stereotype: «instanceRef»,

Notes:

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

Relationships

Name	Source/Target	Notes
	Source: HwPinInHwComponentInstanceRef.	

Name	Source/Target	Notes
	Target: **1** HardwarePin.targetHwPin	
	Source: **0..1** HwPinInHwComponentInstanceRef. **Target:** HardwarePinConnector.	
	Source: HwPinInHwComponentInstanceRef. **Target:** **1** HardwareComponentType.baseHwComponent	
	Source: HwPinInHwComponentInstanceRef. **Target:** **1** HardwareComponentPrototype.contextHwComponent	

Element "HwPortInComponentInstanceRef"

Parent Package: _instanceRef

Stereotype: «instanceRef»,

Notes:

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

Relationships

Name	Source/Target	Notes
	Source: **0..1** HwPortInComponentInstanceRef. **Target:** HwPortConnector.	
	Source: HwPortInComponentInstanceRef. **Target:** **1** HardwarePort.targetHwPort	
	Source: HwPortInComponentInstanceRef. **Target:** **1** HardwareComponentType.baseHwComponent	
	Source: HwPortInComponentInstanceRef. **Target:** **1** HardwareComponentPrototype.contextHwComponent	

7 References

SAFE Website: www.safe-project.eu

SAFE_D2.1.a-ISO-Part_2.pdf (Management of functional safety)

SAFE_D2.1.a-ISO-Part_3.pdf (Concept Phase)

SAFE_D2.1.a-ISO-Part_4.pdf (Product development at the system level)

SAFE_D2.1.a-ISO-Part_5.pdf (Product development at the hardware level)

SAFE_D2.1.a-ISO-Part_6.pdf (Product development at the software level)

SAFE_D2.1.a-ISO-Part_7.pdf (Production and operation)

SAFE_D2.1.a-ISO-Part_8.pdf (Supporting Processes)

SAFE_D2.1.a-ISO-Part_9.pdf (Automotive Safety Integrity Level (ASIL)-oriented safety-oriented analysis