
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

## WP4

# Deliverable D4.2.6b: Final version of plug-in for safety and multi criteria architecture modeling and benchmarking

**Due date of deliverable:** 29/04/2014

**Actual submission date:** 07/05/2014

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

**Start date of the work task:** 30/05/2012 **Duration:** 23 months

**Project coordinator name:** Stefan Voget

**Organization name of lead contractor for this deliverable:** FZI Forschungszentrum Informatik

Editors: Nico Adler, Stefan Otten

Contributors: Nico Adler, Stefan Otten, Eduard Metzker

Reviewers: Eduard Metzker, Martin Hillenbrand

Revision chart and history log

| Version | Date | Reason |
|---|---|---|
| 0.1 | 17.12.2012 | Initialization of document |
| 0.2 | 23.04.2013 | Update document structure |
| 0.3 | 02.05.2013 | Integration of FZI content: hardware safety evaluation |
| 0.4 | 16.08.2013 | Update FZI content: hardware modeling and safety evaluation |
| 0.5 | 23.08.2013 | Integration of Vector content: coverage and consistency checks |
| 0.9 | 28.08.2013 | Integration of review comments |
| 1.0 | 31.08.2013 | Finalization of document |
| 1.1 | 25.02.2014 | Initialization of document D4.2.6b |
| 1.2 | 06.03.2014 | Update of document structure |
| 1.3 | 26.03.2014 | Update FZI contribution: hardware modeling and safety evaluation (chapter 6) |
| 1.4 | 09.04.2014 | Update Vector contribution: generating and consistency checking of safety case reports (chapter 7) |
| 1.5 | 06.05.2014 | Update based on review comments |
| 2.0 | 07.05.2014 | Finalization of document D4.2.6b |

# 1 Table of Contents

## 2 List of Figures

# 3 Executive Summary

The deliverable D4.2.6b *"Final version of plug-in for safety and multi criteria architecture modeling and benchmarking"* is included in the work package 4 "*Technology Platform"* and presents the implementations of concepts and methodologies provided by especially work task WT3.3.3 *"Safety and multi-criteria architecture benchmarking"*. Therefore, this deliverable describes private implementations for model-based safety evaluation in context of ISO 26262. The implementations are partially based on the technology platform PREEvision for model-based description of large-scaled electric and electronic architectures. Two topics are addressed: On the one hand, modeling of hardware designs, enrichment of failure data and evaluation in context of functional safety. On the other hand generating and consistency checking of the safety case report.

# 4 Introduction and Overview of the Implementations

As shown in Figure 1 the work task WT4.2.6 is based on the WT2.1 *"ISO26262 Analysis"* [2] which provides requirements derived from ISO 26262 [1]. Additional requirements are provided by work task WT2.3 *"Use case scenario"* [3]. Based on these requirements, work task WT3.2.2 *"Hardware description"* [4] provides the meta model for hardware modeling including failure data extension. Work task WT3.3.3 *"Safety and Multi Criteria Architecture Benchmarking"* [5] specifies the methodology for the prototype implementations described in this deliverable. Implementations based on the tool environment PREEvision could be potentially linked with work task WT4.3 "*PREEvision extension*".

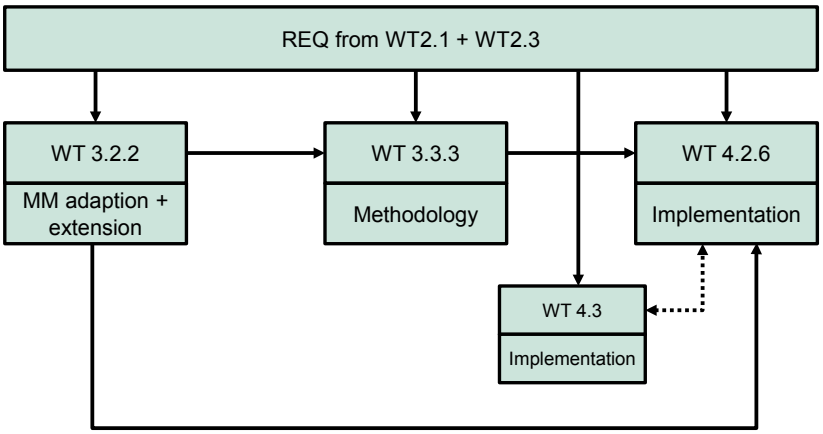



**Figure 1:** Overview of WT4.2.6 links

## 4.1 Hardware Modeling and Safety Evaluation

The prototype implementations provided by FZI are tailored to the demands of ISO 26262 [1] to support hardware modeling and model-based evaluation of hardware designs in context of functional safety. Qualitative and quantitative evaluations for hardware architectural designs and hardware detailed designs are provided including verification against target values. Reports are generated to cope with the documentation of the safety case. The prototype implementations facilitate iterative safety evaluations of hardware designs during different development phases. The implementations are described in Section 6.

## 4.2 Generating and Checking the Consistency of Safety Case Reports

The implementation provided by Vector supports a model-based approach to generate safety case reports. Furthermore consistency checks are provided so that the completeness and maturity of a safety case can be examined at any time during the development process. The provided semi-automatic mechanism have the potential to reduce the effort for creating a safety case report document and the effort involved in reviewing the structure and completeness of safety cases. Both safety case report and safety case consistency checks are implemented based on the PREEvision technology. The implementation is described in Section 7.

# 5 Description of Technology Platform PREEvision

PREEvision [6] is a tool environment for the model-based development of large-scaled electric and electronic architectures.

## 5.1 Abstraction Layers

PREEvision contains different abstraction layers for the description of electric and electronic architectures. Figure 2 gives an overview of the abstraction layers.

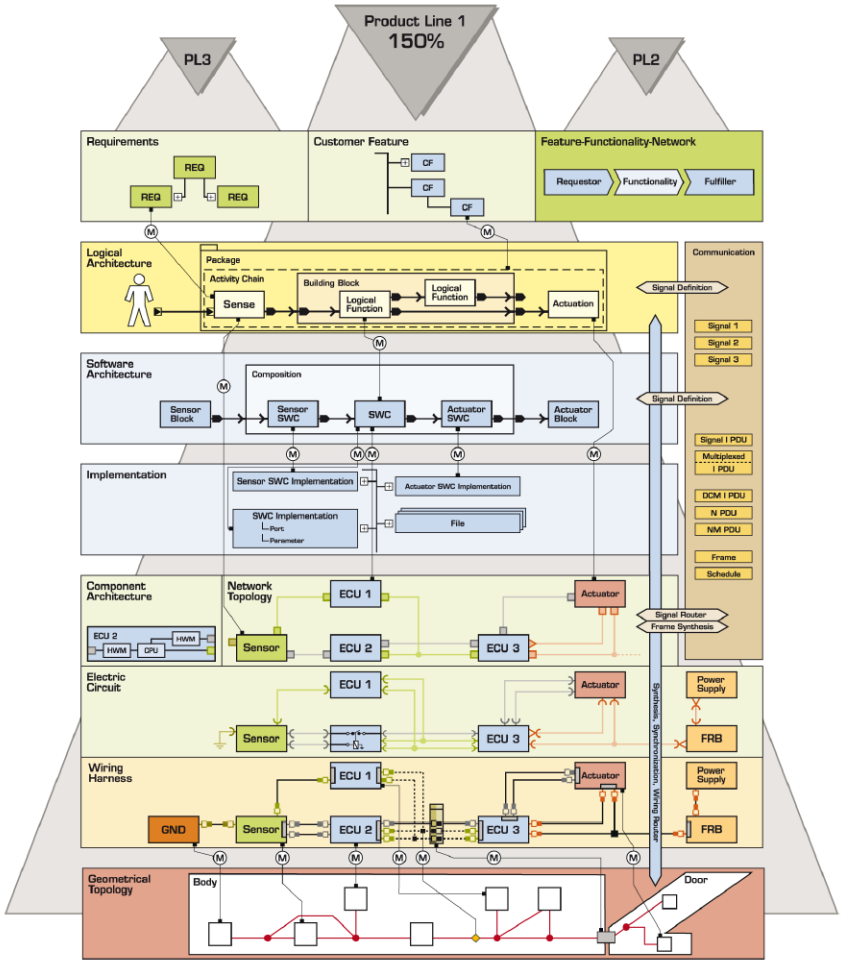



**Figure 2:** Different abstraction layers of PREEvision [6]

Functional and non-functional requirements can be described in the **Requirements Layer**. Additionally, customer features which represent the characteristics of a vehicle can be formulated. In the **Logical Architecture Layer** an abstract functional description of the architecture can be expressed. This functional description is independent from the concrete realization in hardware or software. The **Software Architecture Layer** and **Implementation Layer** facilitate modeling of software architectures, supporting AUTOSAR methodology. The **Hardware Layer** provides a description from hardware components and network topology down to the electric circuit and wiring harness. The **Geometrical Layer** supports a topology modeling including geometrical information. Relevant layers for this deliverable are briefly described in the following.

## 5.2 Excerpt of PREEvision Features

### Mapping Concept:

Besides modeling of electric/electronic relevant information, PREEvision provides mappings to link artifacts of different layers. Figure 3 shows exemplarily the mapping of the requirement "*InsideLight*" on the ECU artifact "*LightManagement*". The requirement is contained by the requirement package "*FunctionalRequirements*".


**Figure 3:** Example for mapping container in PREEvision

### Metric Framework and Rule Model:

PREEvision provides a metric engine for the execution of user-defined metrics implemented in Java. For data acquisition, model queries can be specified using the rule model. The execution of such metrics allows amongst others a model-based analysis of the data model. Additionally, model operations can be performed. A basic metric structure in PREEvision is exemplarily shown in Figure 4.

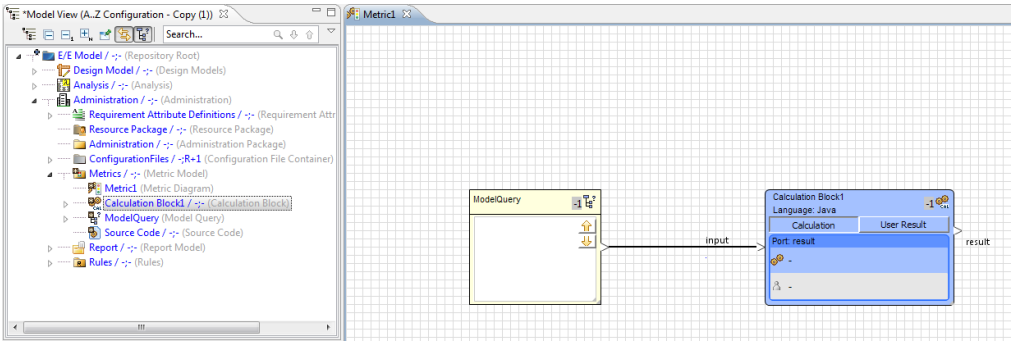



**Figure 4:** Example of metric model in PREEvision

### Report Generation:

PREEvision provides generation of reports for documentation of model data or metric results. Figure 5 shows exemplarily a report template and the specific notation for placeholders. User-defined data from the model or results of metrics are automatically inserted during generation of the report. The output format can be selected as .odt or .pdf.

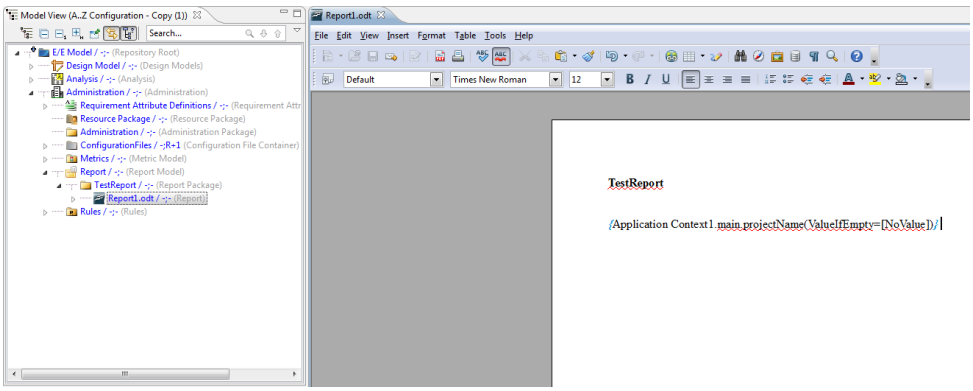



**Figure 5:** Example of report generation in PREEvision

# 6 Hardware Modeling and Safety Evaluation

Based on the SAFE concepts for hardware modeling and safety evaluation presented in the deliverables D3.2.2b [4] and D3.3.3b [5], FZI provides private prototype implementations for structural modeling, enrichment of failure data and safety evaluation for hardware architectural designs and hardware detailed designs in context of ISO 26262, as shown in Figure 6.

Hardware designs at architectural and detailed level can be modeled complementary, based on the technology platform PREEvision v6.0. The model-based hardware safety evaluation in context of ISO 26262 is provided by a prototype implementation in PREEvision and a standalone research prototype implementation, including generation of reports for documentation.

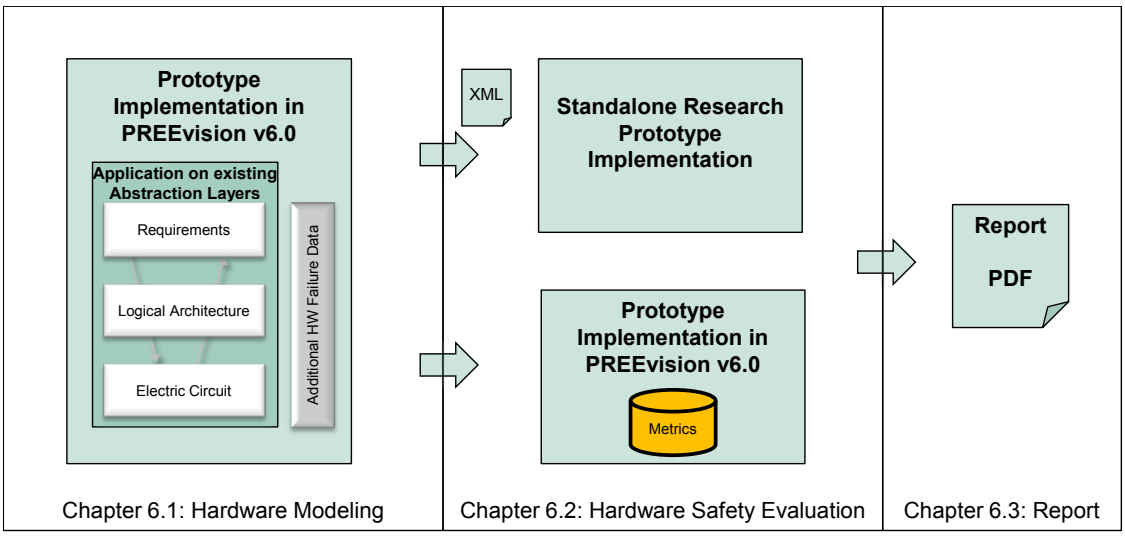



**Figure 6:** Conceptual overview of prototype implementations for hardware safety evaluation

Work task WT3.2.2 "*Hardware description"* focused on the meta model for the description of hardware elements regarding structural modeling and failure data. Based on the analysis of ISO 26262, two main subjects are covered:

- A proposal for meta model adaption of EAST-ADL [10] to fit the needs of WT3.2.2 hardware structural modeling
- A meta model extension provided by FZI regarding annotations of failure data for hardware elements

Work task WT3.3.3 "*Safety and multi-criteria architecture benchmarking"* covers the methodology and process description for hardware modeling and safety evaluation. ISO 26262 [1] Part 5 claims two different assessments for hardware designs: the "evaluation of the hardware architectural metrics" (Clause 8) and the "evaluation of residual risk of safety goal violations" (Clause 9). The hardware architectural metrics, which is the general term, describe two metrics: the single-point fault metric and the latent-fault metric. These allow an evaluation of the robustness of the hardware architecture. Regarding the evaluation of residual risk of safety goal violations, ISO 26262 describes two different methodologies: the probabilistic metric for random hardware failures (PMHF) and the evaluation of each cause of safety goal violation using failure rate classes (FRC). PMHF describes a global probabilistic value for the violation of the safety goal whereas FRC presents an individual evaluation of each violation.

## 6.1 Prototype Implementations for Modeling of Hardware Designs

The prototype implementations regarding modeling of hardware designs in context of functional safety are based on PREEvision v6.0.

### 6.1.1 Modeling of Safety Requirements and Safety Mechanisms

For modeling of safety requirements such as safety goals and information about safety mechanisms, the requirements layer of PREEvision was used as shown in Figure 7. The diagnostic coverages with respect to residual faults $K_{DC,RF}$ and with respect to latent faults $K_{DC,MPFL}$ of the safety mechanism are specified as generic attributes. To structure the different types, two requirement packages were introduced, one named "*SafetyRequirements"* and the other named "*SafetyMechanisms"*. Safety requirements and safety mechanisms can be mapped to the hardware elements.

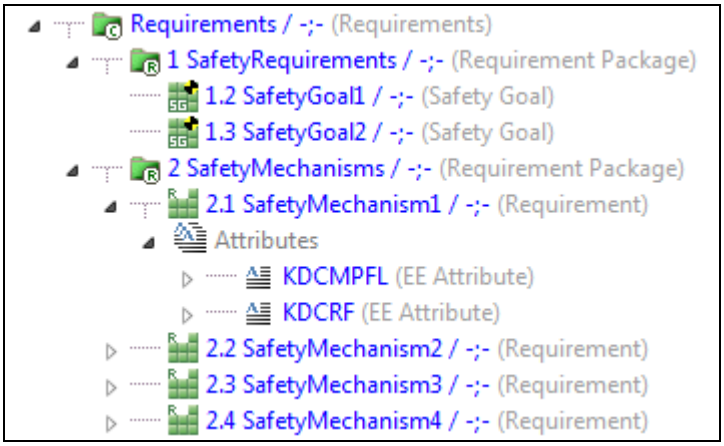



**Figure 7:** Model view for safety requirements and safety mechanisms

### 6.1.2 Hardware Element Type Library

To enhance the modeling of hardware designs in terms of functional safety we provide a type library concept for the annotation of hardware failure data based on the meta model presented in D3.2.2b [4]. The library in PREEvision was extended with generic attributes for each hardware element type. Basic attributes are described in the following Table 1.

| | Description | Data Type |
|---|---|---|
| **FailureMode** | This attribute contains a failure mode of the hardware element type. The name of the attribute specifies the failure (e.g. *ShortCircuit*). The attribute value specifies the failure rate distribution for the failure mode. | A data type named *FailureMode* was created to distinguish the different attributes of the hardware element types. The data type specifies the unit which is percentage [%]. |
| **FailureRate** | This attribute contains the failure rate of the hardware element type. The name of the attribute is *FailureRate*, the value is a floating point number. | A data type *FailureRate* was introduced to distinguish between different attributes of the hardware element type. The data type specifies the unit which is failure-in-time [FIT]. One FIT is one failure in $10^9$ h. |

**Table 1:** Generic attributes for hardware element library types

### 6.1.3 Modeling of Hardware Architectural and Detailed Designs

ISO 26262 Part 5 Clause 7 describes two different levels of abstraction for hardware designs: the hardware detailed design (Clause 7.4.2) at the level of electronic schematics consisting of hardware parts and the hardware architectural design (Clause 7.4.2) representing an initial view on the hardware consisting of hardware components. The iterative application of safety evaluations at both levels of abstraction has to be ensured and is provided by our prototype implementations.

Based on the type library concept with the included failure information, structural modeling of hardware architectural and hardware detailed design is facilitated. Figure 8 gives an overview of the prototype implementations for both levels of abstraction.

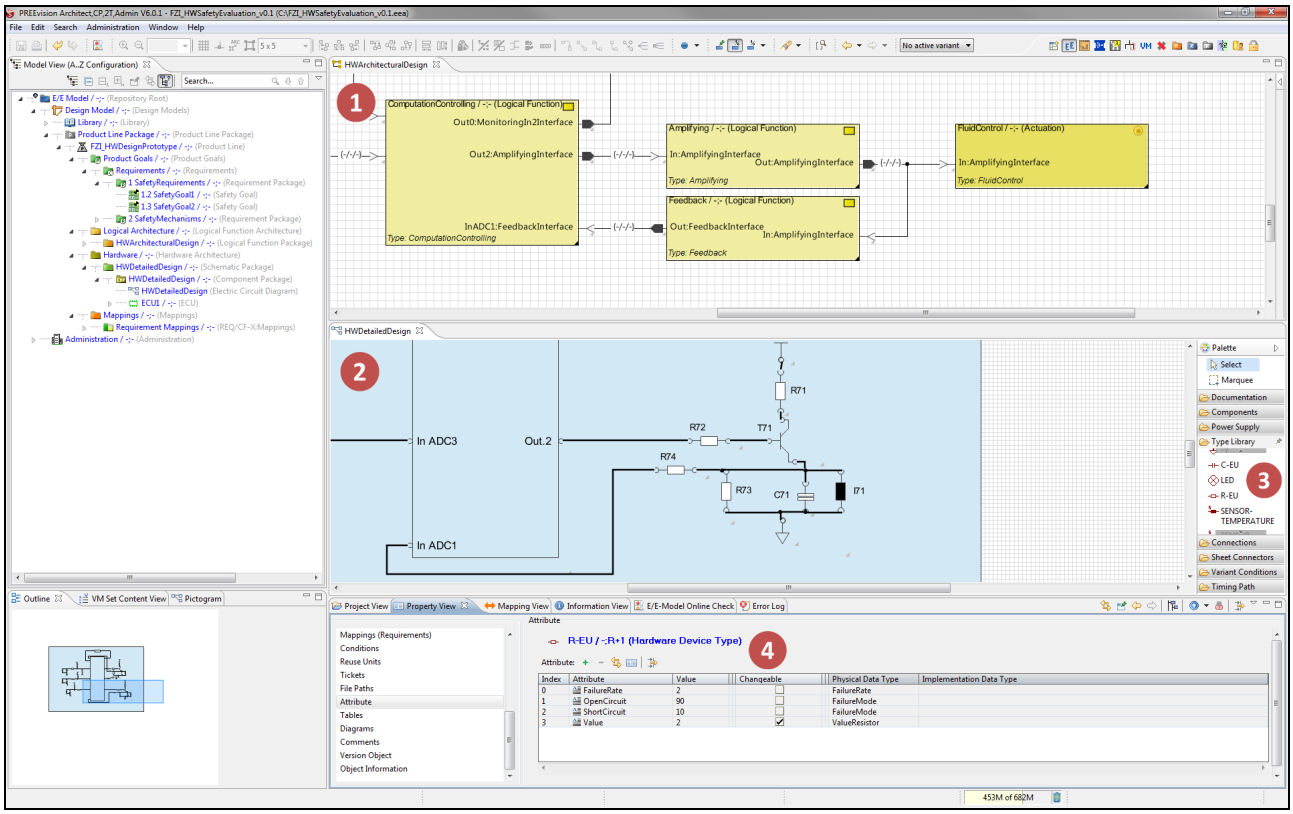



**Figure 8:** Modeling example of hardware architectural and hardware detailed design

The hardware architectural design was implemented in the logical architecture layer ❶, the hardware detailed design in the hardware architecture layer ❷. The instantiation of hardware elements, based on the respective library, can be performed using the palette ❸. The attributes regarding failure information are exemplarily shown in the property view ❹.

In Figure 9 the model for a valve control, described in ISO 26262 [1] Part 5 Annex E is exemplarily shown in PREEvision. For research purposes, a prototypical import for EAGLE schematic files was implemented.

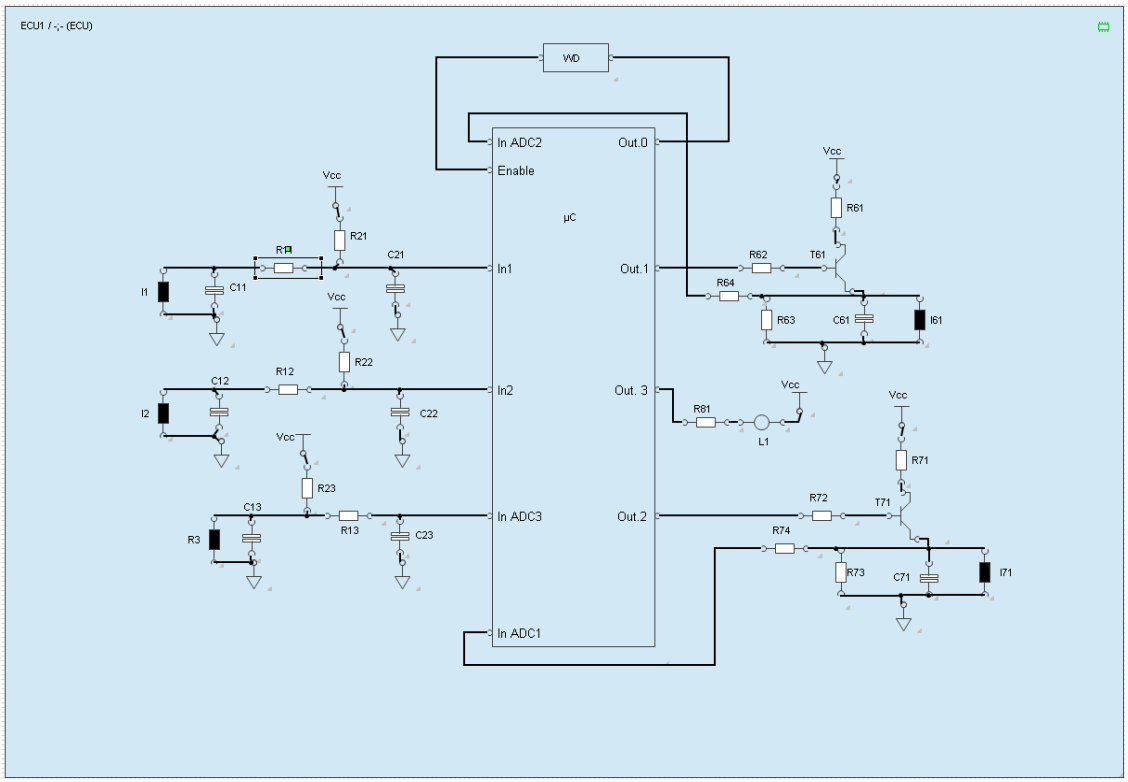



**Figure 9:** Model of ISO 26262 [1] Part 5 Annex E example

## 6.2 Prototype Implementations for Hardware Safety Evaluation

Based on the presented modeling of hardware designs, prototype implementations for the model-based evaluation in context of functional safety are provided. To perform the evaluations, we provide a qualitative and quantitative fault tree analysis [9], which also serves for an automatic classification of failure modes according to ISO 26262 [1] Part 10 Annex B. Additionally to the fault tree analysis, we implemented a fault tree generation based on textual description of failure propagation deposited in the data model. See [7] for a graphical fault tree modeling approach in PREEvision. Quantitative analysis of the fault tree supports the evaluation of residual risk of safety goal violations in terms of PMHF. For each safety requirement to assess, a failure data table according to ISO 26262 [1] Part 5 Annex E is generated. Quantitative evaluation of the hardware architectural metrics and evaluation of each cause of safety goal violation (FRC method) is provided. An excerpt of the prototype implementation in PREEvision based on the metric framework is shown in Figure 10.

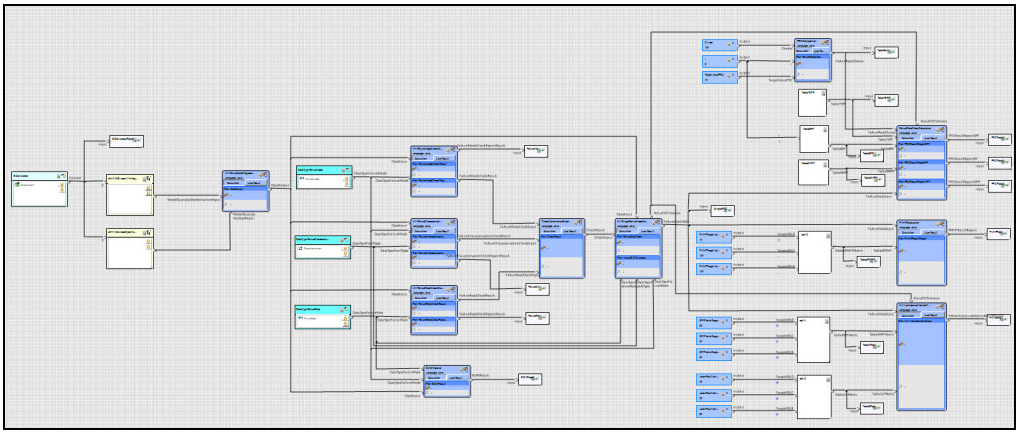


**Figure 10:** Prototype implementation for hardware safety evaluation in PREEvision metric diagram

**Standalone Research Prototype Implementation:**

The modeled hardware designs including the deposited failure data can be imported in a standalone research prototype implementation using an own-defined XML schema. The standalone implementation provides the same features for hardware safety evaluation as described before. An overview is given in Figure 11.

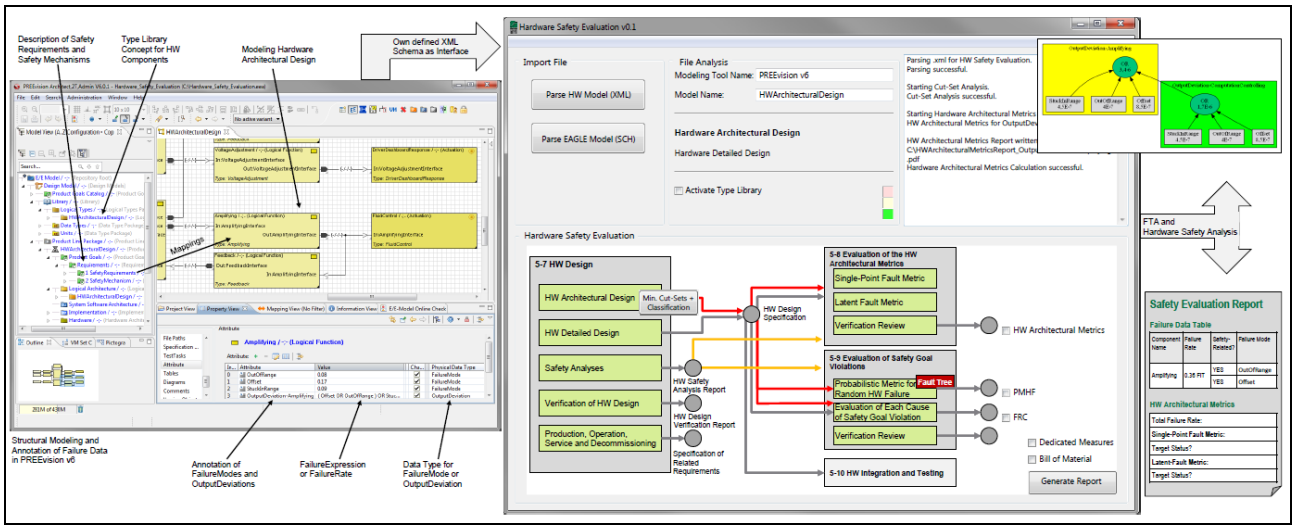


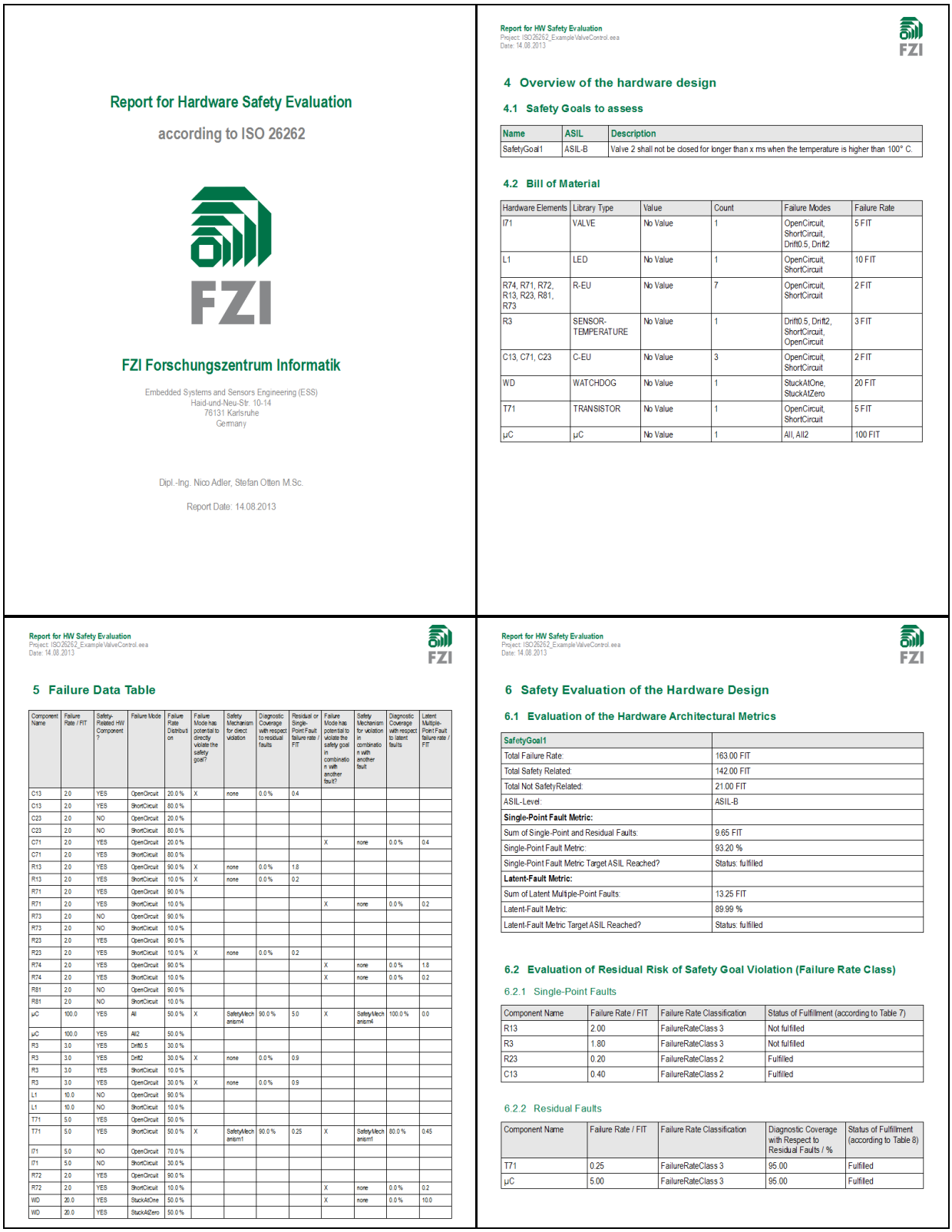

**Figure 11:** Prototype implementation for modeling and failure data annotation in PREEvision linked with prototype implementation for hardware safety evaluation [9]

## 6.3 Reports for Documentation

For documentation of the safety case, several reports including all relevant information from the hardware design model and results of the hardware safety evaluation can be automatically generated. The reports in pdf file format include the following:

- Graphical representation of the hardware design
- Bill of material
- Failure data table
- Results of the evaluation of the hardware architectural metrics
- Results of the evaluation of residual risk of safety goal violations using FRC (including dedicated measures)
- Results of qualitative and quantitative FTA supporting a PMHF
- Graphical visualization of the fault tree including minimal cut-sets

Figure 12 shows exemplarily an excerpt of the generated report for the ISO 26262 example of a valve control including the failure data table, the results of evaluation of the hardware architectural metrics and failure rate class method.



Report for Hardware Safety Evaluation

according to ISO 26262

FZI

FZI Forschungszentrum Informatik

Embedded Systems and Sensors Engineering (ESS)
Haid-und-Neu-Str. 10-14
76131 Karlsruhe
Germany

Dipl.-Ing. Nico Adler, Stefan Otten M.Sc.

Report Date: 14.08.2013

Report for HW Safety Evaluation
Project: ISO26262_ExampleValveControl.eea
Date: 14.08.2013

FZI

## 4 Overview of the hardware design

### 4.1 Safety Goals to assess

| Name | ASIL | Description |
|---|---|---|
| SafetyGoal1 | ASIL-B | Valve 2 shall not be closed for longer than x ms when the temperature is higher than 100° C. |

### 4.2 Bill of Material

| Hardware Elements | Library Type | Value | Count | Failure Modes | Failure Rate |
|---|---|---|---|---|---|
| I71 | VALVE | No Value | 1 | OpenCircuit, ShortCircuit, Drift0.5, Drift2 | 5 FIT |
| L1 | LED | No Value | 1 | OpenCircuit, ShortCircuit | 10 FIT |
| R74, R71, R72, R13, R23, R81, R73 | R-EU | No Value | 7 | OpenCircuit, ShortCircuit | 2 FIT |
| R3 | SENSOR-TEMPERATURE | No Value | 1 | Drift0.5, Drift2, ShortCircuit, OpenCircuit | 3 FIT |
| C13, C71, C23 | C-EU | No Value | 3 | OpenCircuit, ShortCircuit | 2 FIT |
| WD | WATCHDOG | No Value | 1 | StuckAtOne, StuckAtZero | 20 FIT |
| T71 | TRANSISTOR | No Value | 1 | OpenCircuit, ShortCircuit | 5 FIT |
| µC | µC | No Value | 1 | All, All2 | 100 FIT |

Report for HW Safety Evaluation
Project: ISO26262_ExampleValveControl.eea
Date: 14.08.2013

FZI

## 5 Failure Data Table

| Component Name | Failure Rate / FIT | Safety-Related HW Component ? | Failure Mode | Failure Rate Distribution | Failure Mode has potential to directly violate the safety goal? | Safety Mechanism for direct violation | Diagnostic Coverage with respect to residual faults | Residual or Single-Point Fault failure rate / FIT | Failure Mode has potential to violate the safety goal in combination with another fault? | Safety Mechanism for violation in combination with another fault | Diagnostic Coverage with respect to latent faults | Latent Multiple-Point Fault failure rate / FIT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C13 | 2.0 | YES | OpenCircuit | 20.0 % | X | none | 0.0 % | 0.4 | | | | |
| C13 | 2.0 | YES | ShortCircuit | 80.0 % | | | | | | | | |
| C23 | 2.0 | NO | OpenCircuit | 20.0 % | | | | | | | | |
| C23 | 2.0 | NO | ShortCircuit | 80.0 % | | | | | | | | |
| C71 | 2.0 | YES | OpenCircuit | 20.0 % | | | | | X | none | 0.0 % | 0.4 |
| C71 | 2.0 | YES | ShortCircuit | 80.0 % | | | | | | | | |
| R13 | 2.0 | YES | OpenCircuit | 90.0 % | X | none | 0.0 % | 1.8 | | | | |
| R13 | 2.0 | YES | ShortCircuit | 10.0 % | X | none | 0.0 % | 0.2 | | | | |
| R71 | 2.0 | YES | OpenCircuit | 90.0 % | | | | | | | | |
| R71 | 2.0 | YES | ShortCircuit | 10.0 % | | | | | X | none | 0.0 % | 0.2 |
| R73 | 2.0 | NO | OpenCircuit | 90.0 % | | | | | | | | |
| R73 | 2.0 | NO | ShortCircuit | 10.0 % | | | | | | | | |
| R23 | 2.0 | YES | OpenCircuit | 90.0 % | | | | | | | | |
| R23 | 2.0 | YES | ShortCircuit | 10.0 % | X | none | 0.0 % | 0.2 | | | | |
| R74 | 2.0 | YES | OpenCircuit | 90.0 % | | | | | X | none | 0.0 % | 1.8 |
| R74 | 2.0 | YES | ShortCircuit | 10.0 % | | | | | X | none | 0.0 % | 0.2 |
| R81 | 2.0 | NO | OpenCircuit | 90.0 % | | | | | | | | |
| R81 | 2.0 | NO | ShortCircuit | 10.0 % | | | | | | | | |
| µC | 100.0 | YES | All | 50.0 % | X | SafetyMechanism4 | 90.0 % | 5.0 | X | SafetyMechanism4 | 100.0 % | 0.0 |
| µC | 100.0 | YES | All2 | 50.0 % | | | | | | | | |
| R3 | 3.0 | YES | Drift0.5 | 30.0 % | | | | | | | | |
| R3 | 3.0 | YES | Drift2 | 30.0 % | X | none | 0.0 % | 0.9 | | | | |
| R3 | 3.0 | YES | ShortCircuit | 10.0 % | | | | | | | | |
| R3 | 3.0 | YES | OpenCircuit | 30.0 % | X | none | 0.0 % | 0.9 | | | | |
| L1 | 10.0 | NO | OpenCircuit | 90.0 % | | | | | | | | |
| L1 | 10.0 | NO | ShortCircuit | 10.0 % | | | | | | | | |
| T71 | 5.0 | YES | OpenCircuit | 50.0 % | | | | | | | | |
| T71 | 5.0 | YES | ShortCircuit | 50.0 % | X | SafetyMechanism1 | 90.0 % | 0.25 | X | SafetyMechanism1 | 80.0 % | 0.45 |
| I71 | 5.0 | NO | OpenCircuit | 70.0 % | | | | | | | | |
| I71 | 5.0 | NO | ShortCircuit | 30.0 % | | | | | | | | |
| R72 | 2.0 | YES | OpenCircuit | 90.0 % | | | | | | | | |
| R72 | 2.0 | YES | ShortCircuit | 10.0 % | | | | | X | none | 0.0 % | 0.2 |
| WD | 20.0 | YES | StuckAtOne | 50.0 % | | | | | X | none | 0.0 % | 10.0 |
| WD | 20.0 | YES | StuckAtZero | 50.0 % | | | | | | | | |

Report for HW Safety Evaluation
Project: ISO26262_ExampleValveControl.eea
Date: 14.08.2013

FZI

## 6 Safety Evaluation of the Hardware Design

### 6.1 Evaluation of the Hardware Architectural Metrics

| SafetyGoal1 | |
|---|---|
| Total Failure Rate: | 163.00 FIT |
| Total Safety Related: | 142.00 FIT |
| Total Not SafetyRelated: | 21.00 FIT |
| ASIL-Level: | ASIL-B |
| **Single-Point Fault Metric:** | |
| Sum of Single-Point and Residual Faults: | 9.65 FIT |
| Single-Point Fault Metric: | 93.20 % |
| Single-Point Fault Metric Target ASIL Reached? | Status: fulfilled |
| **Latent-Fault Metric:** | |
| Sum of Latent Multiple-Point Faults: | 13.25 FIT |
| Latent-Fault Metric: | 89.99 % |
| Latent-Fault Metric Target ASIL Reached? | Status: fulfilled |

### 6.2 Evaluation of Residual Risk of Safety Goal Violation (Failure Rate Class)

#### 6.2.1 Single-Point Faults

| Component Name | Failure Rate / FIT | Failure Rate Classification | Status of Fulfillment (according to Table 7) |
|---|---|---|---|
| R13 | 2.00 | FailureRateClass 3 | Not fulfilled |
| R3 | 1.80 | FailureRateClass 3 | Not fulfilled |
| R23 | 0.20 | FailureRateClass 2 | Fulfilled |
| C13 | 0.40 | FailureRateClass 2 | Fulfilled |

#### 6.2.2 Residual Faults

| Component Name | Failure Rate / FIT | Failure Rate Classification | Diagnostic Coverage with Respect to Residual Faults / % | Status of Fulfillment (according to Table 8) |
|---|---|---|---|---|
| T71 | 0.25 | FailureRateClass 3 | 95.00 | Fulfilled |
| µC | 5.00 | FailureRateClass 3 | 95.00 | Fulfilled |

**Figure 12:** Excerpt of report for hardware safety evaluation regarding ISO 26262 valve example

# 7 Generating and Consistency Checking of Safety Case Reports

The patterns for sections of a safety case report which have been formalized in WT3.3.3 are implemented based on the PREEvision model query rule engine and the PREEvision report framework. Section 7.1 described the integrated work flow of safety engineering and safety case documentation which is enabled by this approach. Section 7.2 briefly describes where the query rule package structure and the report structure can be found in the tool while section 7.3 describes how a safety case report can be generated.

The coverage and consistency checks described in WT3.3.3 are implemented based on the PREEvision consistency rule engine. PREEvision consistency rules empower the user to describe inconsistencies as a pattern of meta model artifacts, having specific attribute values and being connected by defined relations. Section 7.4 describes where the consistency rules can be found in the tool while section 7.5 describes how the consistency checks for a safety case can be started.

## 7.1 Integrated Approach for Safety Engineering and Safety Case Documentation

The approach developed supports an integrated and cooperative way of working between engineers and safety managers to efficiently create the required documentation. All stakeholders can focus on performing the safety engineering tasks. PREEvision allows checking the formal consistency of individual work products at any time in the development process. This allows finding many problems in the work products often before manual walkthroughs and reviews. The model based work products can be used to generate a safety case report at any time. This concept is illustrated in Figure 13.


**Figure 13:** Implemented safety case report workflow

## 7.2 Implementation of Safety Case Reports

Based on the patterns defined for the report sections PREEvision model queries have been defined which fetch the meta model artifacts required for the report. Furthermore a modular PREEvision report structure has been defined which allows to generate the overall safety case report from these building blocks. Figure 14 shows a section of the query package structure and a section of the report package structure.

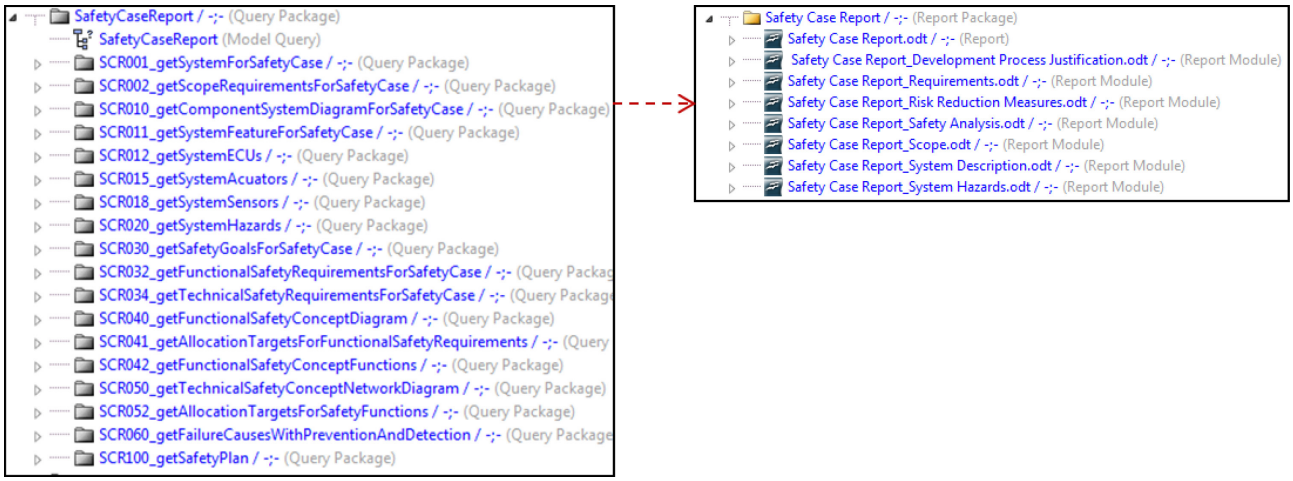



**Figure 14:** Query rules and report modules of a safety case report

## 7.3 Generating a Safety Case Report

In the PREEvision model tree you can now create safety case packages and safety cases as model artifacts. Artifacts can be associated to the safety case via drag and drop from the model tree. The following artifacts can be used to create a safety case report:

- A requirement owner artifact (e.g. a requirement package) with the name “Scope” which contains the requirements which define the scope of the safety case.
- A system artifact which models the system which is subject of the safety case report.
- A hazard and risk analysis which contains the hazards of the system
- Requirement owner artifacts for safety goals, functional safety requirements and technical safety requirements
- An FMEA artifact which contains the failure causes, detection measures and prevention measures for the system
- A Project artifact which contains the work tasks of the safety plan


**Figure 15:** Associating artifacts with a safety case

Figure 15 illustrates how artifacts can be associated with a safety case. On the left side the artifact which represents the hazard and risk analysis is dragged from the model tree and dropped on the property editor of the safety case.

During development of the safety case the consistency of the safety case can be examined by running the consistency checks described in section 7.5.

In this approach the safety case serves as a funnel which can be filled with work products. Based on the meta model, the relevant data is extracted from the work products to create an up to date safety case report document. This concept is illustrated in Figure 16.


**Figure 16:** Safety case as a funnel to create safety case report

The safety case report can be generated by selecting a safety case in the model tree and select "Report → Safety Case Report" from the context menu. The system now generates the report and displays the result as a LibreOffice document.

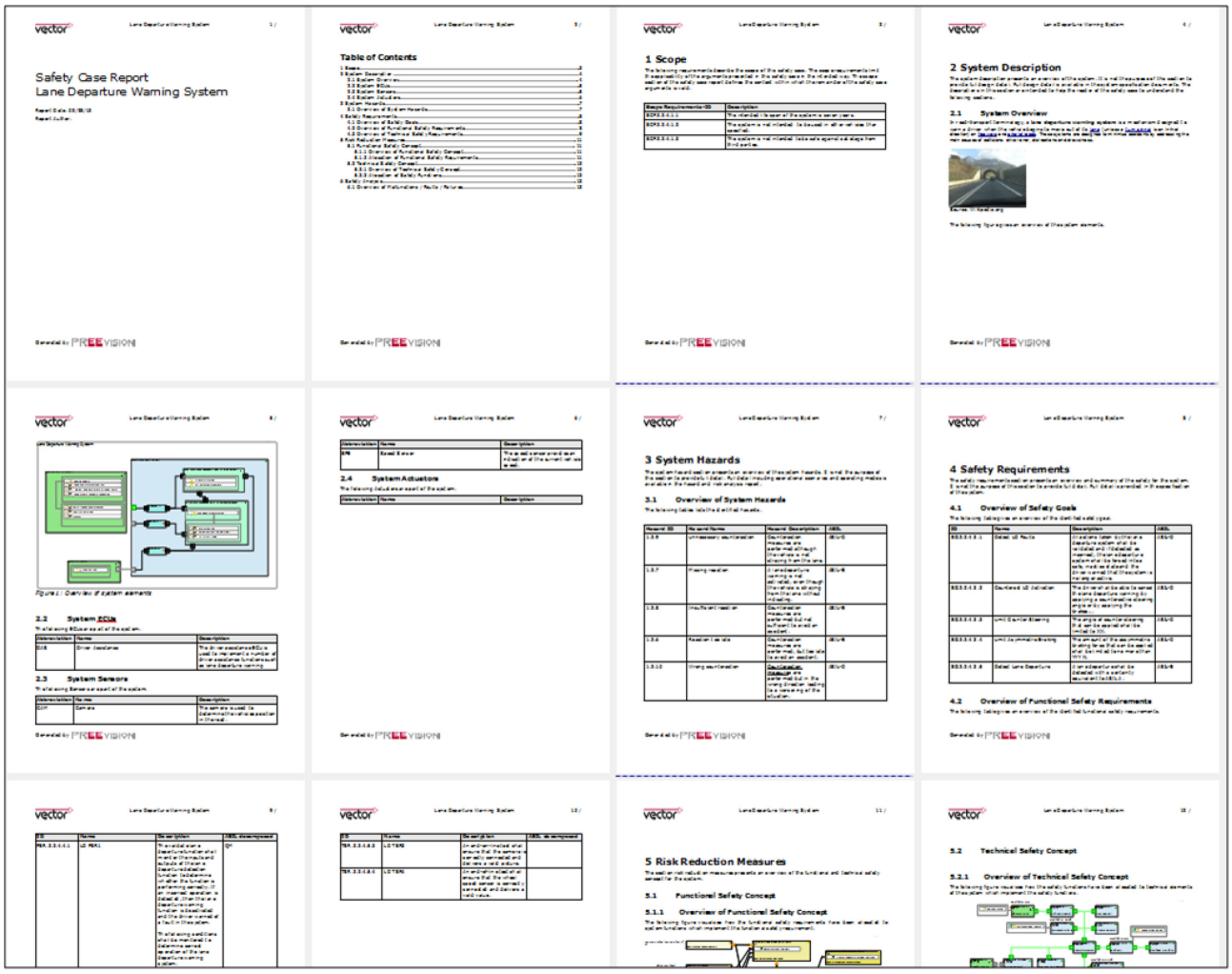



**Figure 17:** Automatically generated safety case report document (overview)

## 7.4 Implementation of Safety Case Consistency Patterns

Based on the patterns defined in the consistency rules, consistency checks can be executed on the integrated architecture and SAFE meta model concepts. By grouping several consistency rules to a group, a set of checks suitable for examining the consistency of a safety case was defined. Figure 18 shows a consistency rule group and a specific rule of that rule group. The rule displayed on the right hand side of the figure checks for hazardous events which have no associated safety goal.

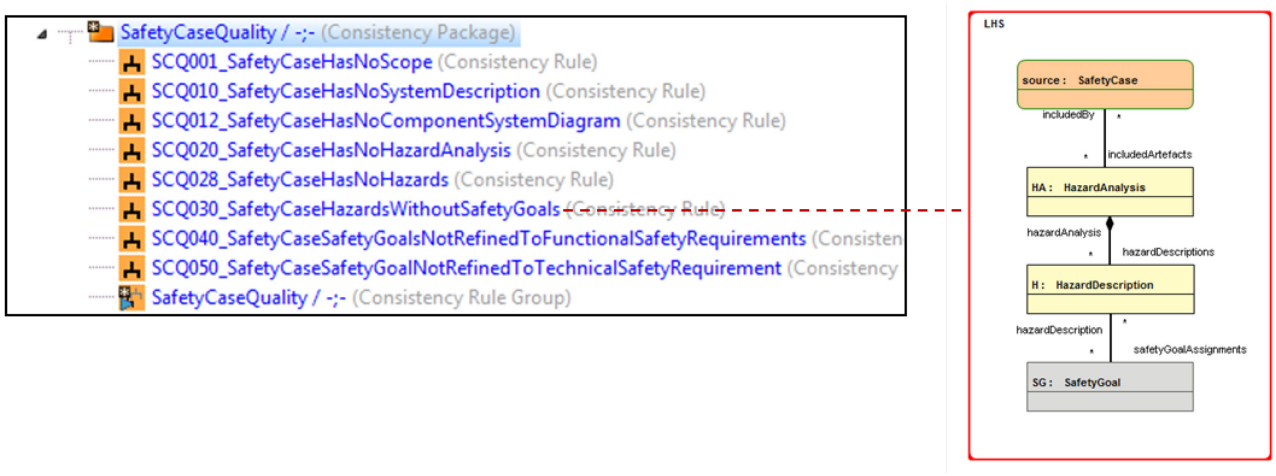



**Figure 18:** Consistency rule group and consistency rule

## 7.5 Performing Checks for Consistency and Coverage of a Safety Case

PREEvisions consistency check perspective can be used to examine and trace the results of consistency checks. In the model view, the user can navigate through all elements relevant to the safety case such as system description, hazard and risk analysis, safety goals etc. While it is possible to execute consistency checks on all artifacts individually, it is most convenient to use the safety case artifact as a starting point, because the safety case artifact serves as a container for all work products. Figure 19 shows an excerpt of the model view including two safety case artifacts.


**Figure 19:** Section of the meta model displayed in the model view

In the following example, the safety case consistency check is performed on the safety case for the lane departure warning system. To execute the check of the safety case the artifact is selected in the model view (Figure 20, ❶) and the consistency check is triggered via the context menu "Execute consistency check → SafetyCaseQuality". PREEvision now applies the consistency rules to the safety case. The results of the consistency check of the safety case is displayed in a table in the inconsistencies view (Figure 20, ❷).The view contains the name of the consistency rule that detected a match together with the elements which are inconsistent. The "Classification" column shows the classification of the match which can be an 'error', 'warning' or 'information'. The column "Short description" provides a brief description of the match which is sufficient for experienced user to fix the inconsistency.

The consistency rule view provides detailed information on the match. The included explanation describes why the inconsistency was reported including a reference to the ISO standard together with additional advice for resolving. The inconsistent model elements are listed below the explanation (Figure 20, ❸) and can be used to directly to navigate to the inconsistent artifact (Figure 20, ❹).

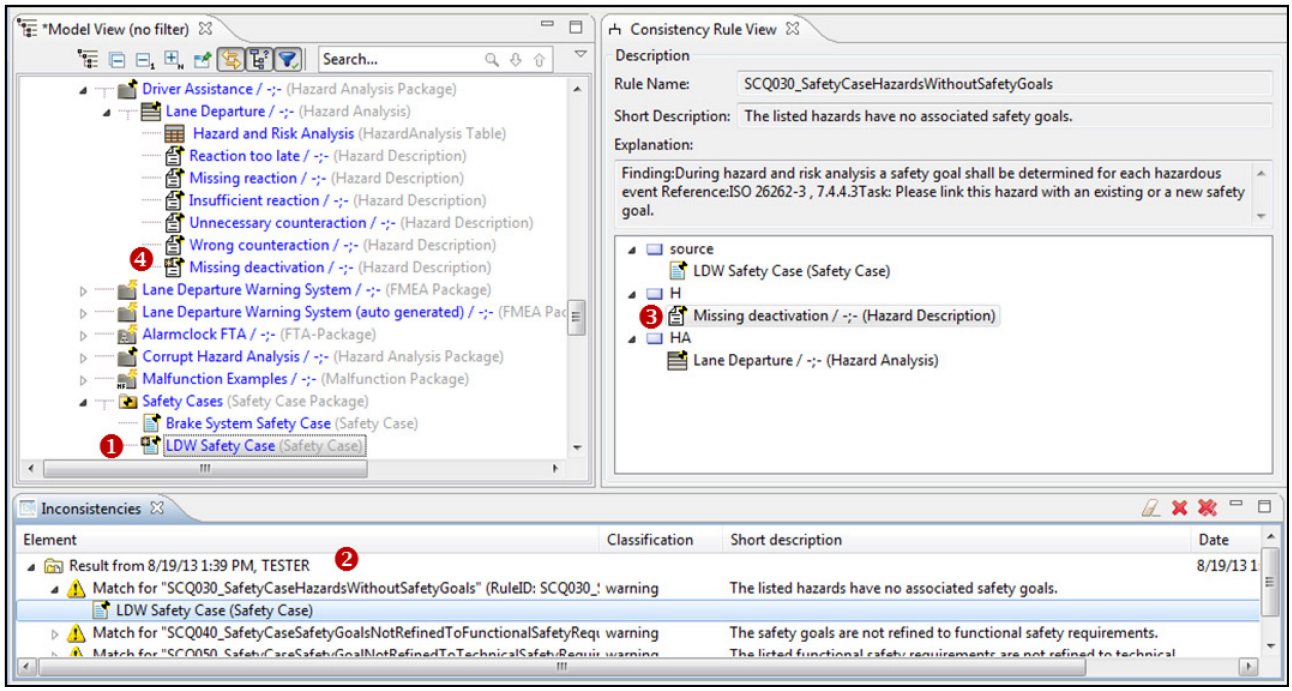



**Figure 20:** Results of consistency checks applied to the safety case

This allows rapidly assessing the progress and formal quality of a safety case including traceability and coverage aspects. By applying this model-based approach the safety case can be evolved in parallel with overall development progress. This represents the major advantage over classical “document based” approaches to compiling safety cases where inconsistencies can reduce the quality and value of a safety case.

# 8 References


[1] International Standards Organization, ISO 26262 Standard, Road Vehicles - Functional Safety, http://www.iso.org/, 2011.

[2] SAFE project: SAFE Deliverable D2.1c: “Needs description to apply ISO26262 with architecture and component modeling”, http://www.safe-project.eu/SAFE-Download.html, 2014.

[3] SAFE project: SAFE Deliverable D2.3.2b: “Detailed description of use case scenarios and list of requirements from industrial practice”, http://www.safe-project.eu/SAFE-Download.html, 2013.

[4] SAFE project: SAFE Deliverable D3.2.2b: “Proposal for extension of meta model for hardware modeling”, http://www.safe-project.eu/SAFE-Download.html, 2013.

[5] SAFE project: SAFE Deliverable D3.3.3b: “Final specification for comparison of architecture”, http://www.safe-project.eu/SAFE-Download.html, 2013.

[6] Vector Informatik GmbH, “PREEvision User Manual Version 6.5”, 2013.

[7] Adler, N., Hillenbrand, M., Müller-Glaser, K.D., Metzker, E., Reichmann, C., "Graphically notated fault modeling and safety analysis in the context of electric and electronic architecture development and functional safety," *Rapid System Prototyping (RSP 2012), 23rd IEEE International Symposium on*, pp.36-42, doi:10.1109/RSP.2012.6380688.

[8] Adler, N., Otten, S., Cuenot, P., and Müller-Glaser, K., "Performing Safety Evaluation on Detailed Hardware Level according to ISO 26262," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 6(1):102-113, 2013, doi:10.4271/2013-01-0182.

[9] Adler, N., Otten, S., Mohrhard, M., Müller-Glaser, K.-D., "Rapid safety evaluation of hardware architectural designs compliant with ISO 26262," *Rapid System Prototyping (RSP), 2013, 24rd IEEE International Symposium on*, pp.66-72, doi:10.1109/RSP.2013.6683960.

[10] ATESST2 Consortium, "EAST-ADL Domain Model Specification - Deliverable D4.1.1," Rev. June 2010.

# 9 Acknowledgments

This document is based on the SAFE and SAFE-E projects. SAFE is in the framework of the ITEA2, EUREKA cluster program Σ! 3674. The work has been funded by the German Ministry for Education and Research (BMBF) under the funding ID 01IS11019, and by the French Ministry of the Economy and Finance (DGCIS). SAFE-E is part of the Eurostars program, which is powered by EUREKA and the European Community. The work has been funded by the German Ministry of Education and Research (BMBF) and the Austrian research association (FFG) under the funding ID E!6095. The responsibility for the content rests with the authors.