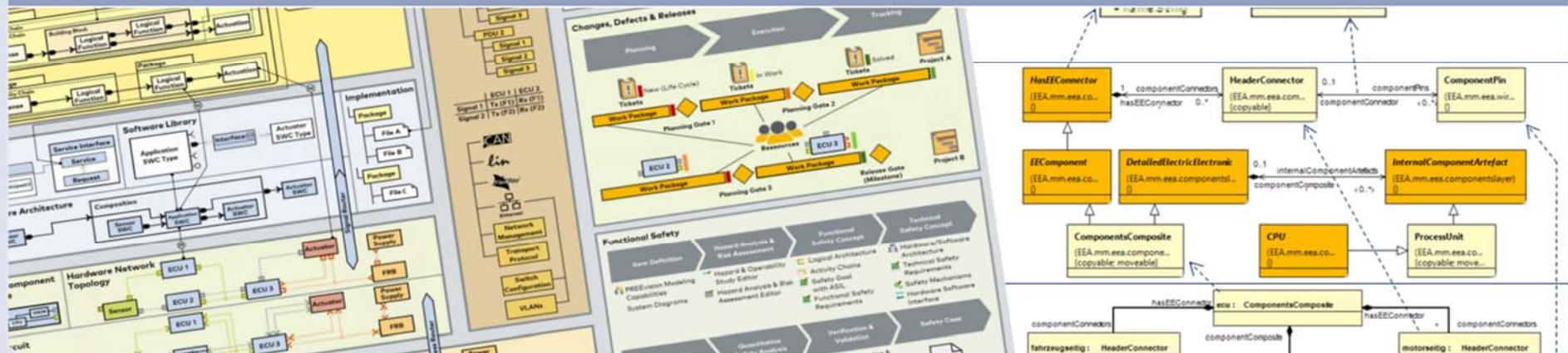


Vorlesung Software Engineering (SE)

Wintersemester 2017/2018

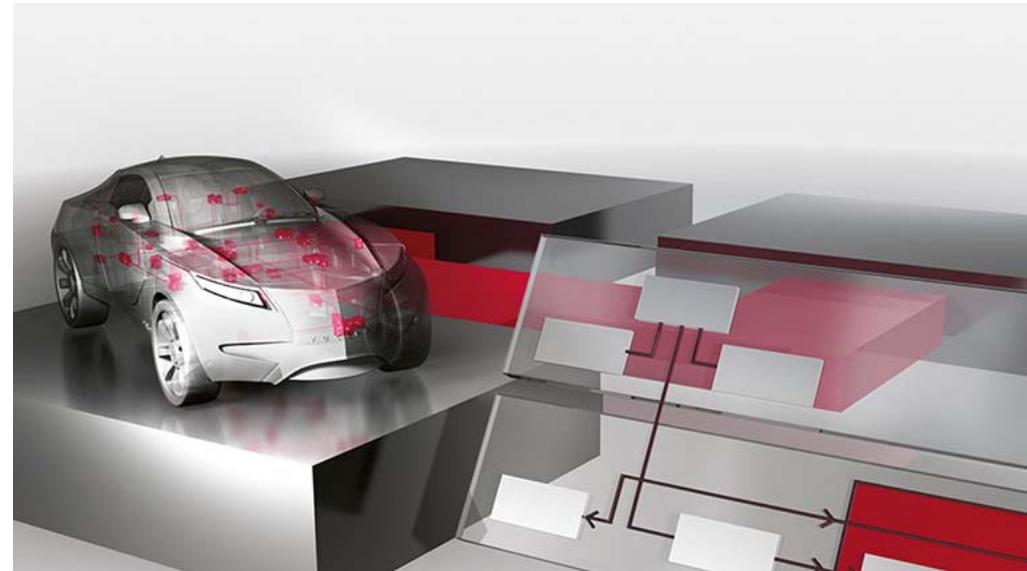
Kapitel 3.5 – Anforderungsmanagement: E/E-Konzeptbewertung



Dr.-Ing. Clemens Reichmann

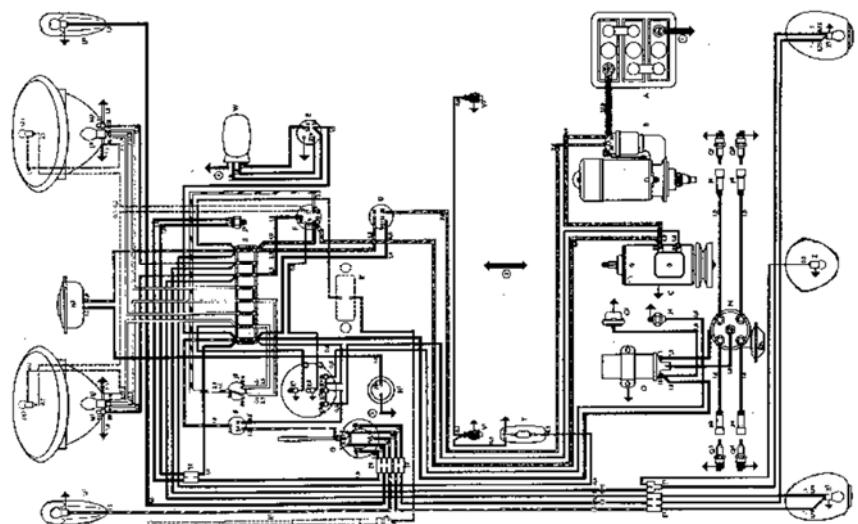
Clemens.Reichmann@vector.com,
Tel. 0721 / 91430-200

Institut für Technik der Informationsverarbeitung
Fakultät für Elektrotechnik & Informationstechnik
Universität Karlsruhe (KIT)

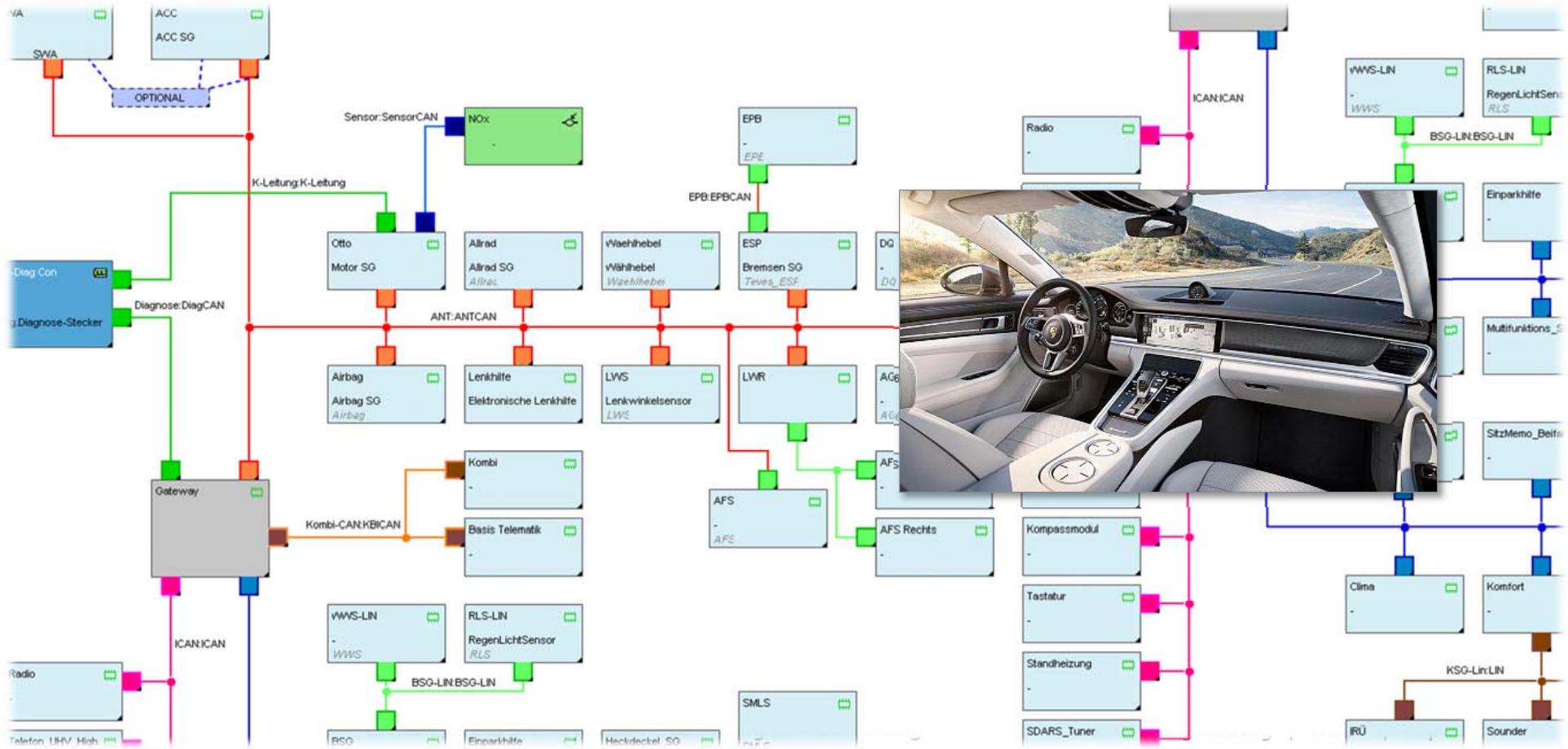


3.5. Elektrik/Elektronik-System-Modellierung u. Konzeptbewertung

Overall E/E Complexity Increasing over Years now...
Past

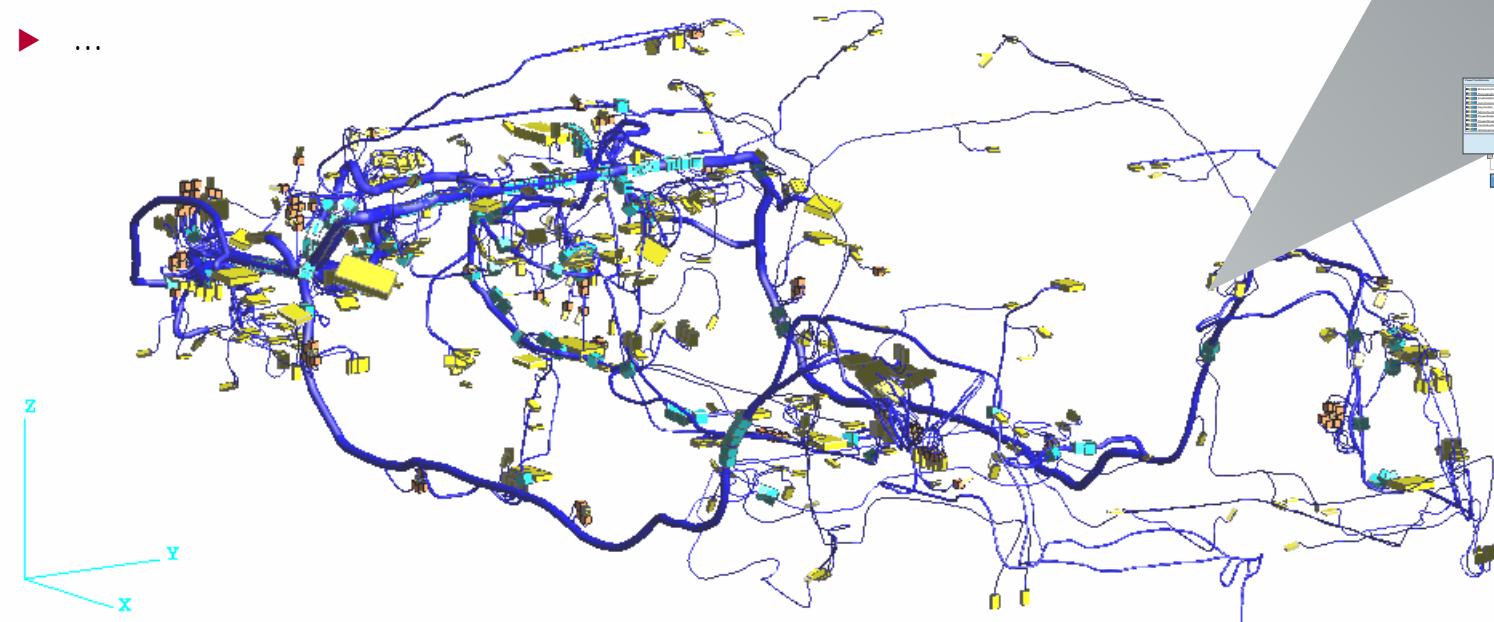


Overall E/E Complexity Increasing over Years now... Present

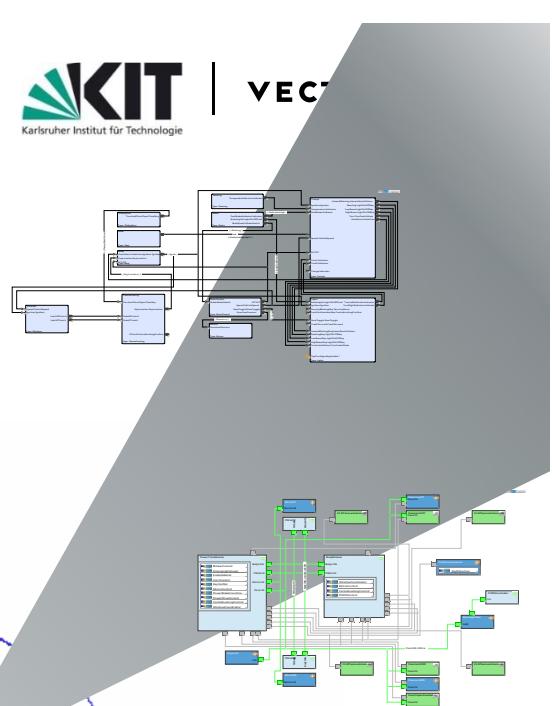


Overall E/E Complexity Increasing over Years now...

- ▶ Active Management of System Complexity is necessary
 - ▶ Economical **cost** targets
 - ▶ Technical targets: weight, **fuel** consumption, ...
 - ▶ Installation **space** is restricted, deeper integration
 - ▶ Upcoming technologies have to be considered and integrated
 - ▶ ...



5

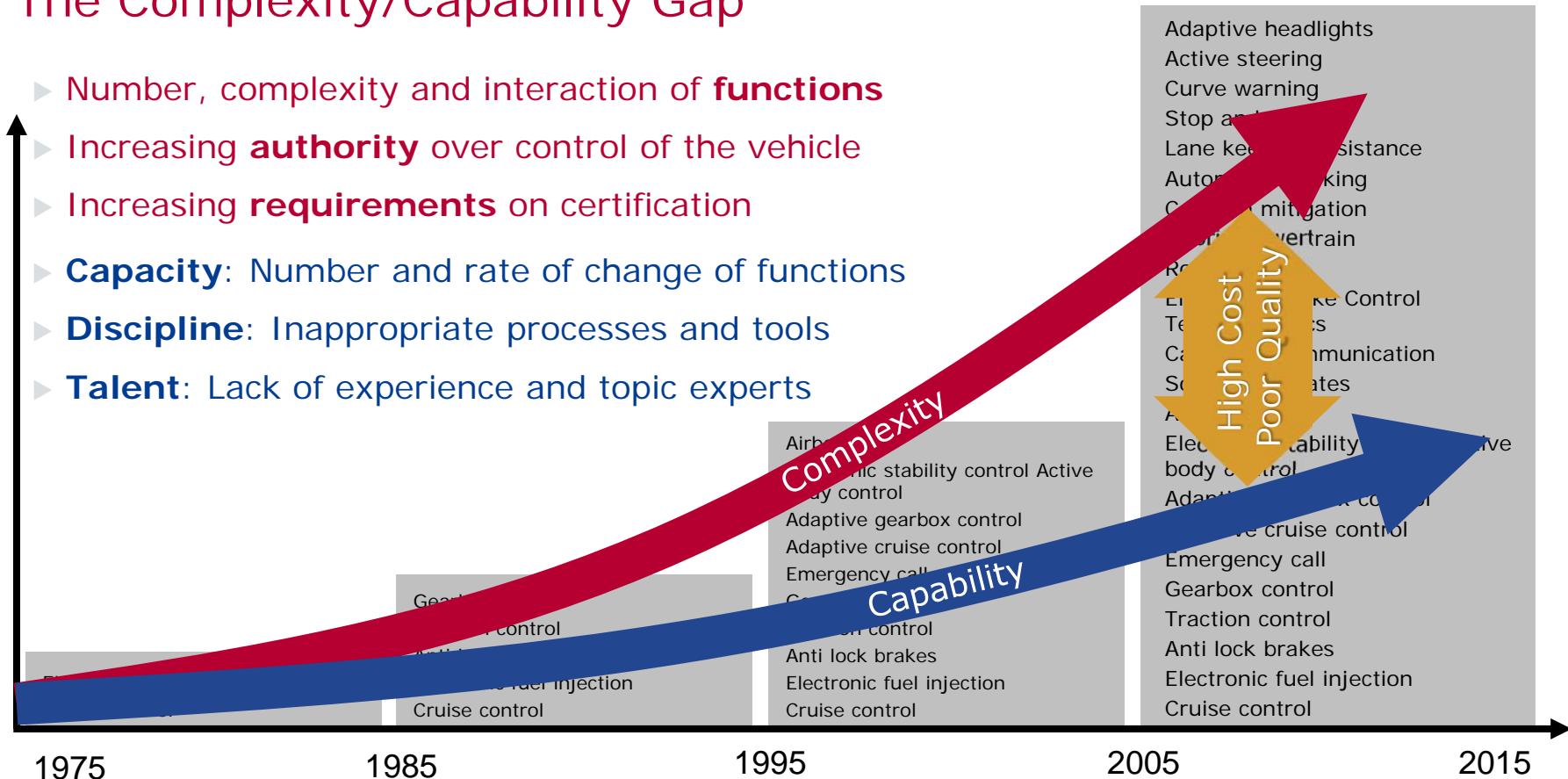


Kapitel 3.5

Challenges in E/E Development

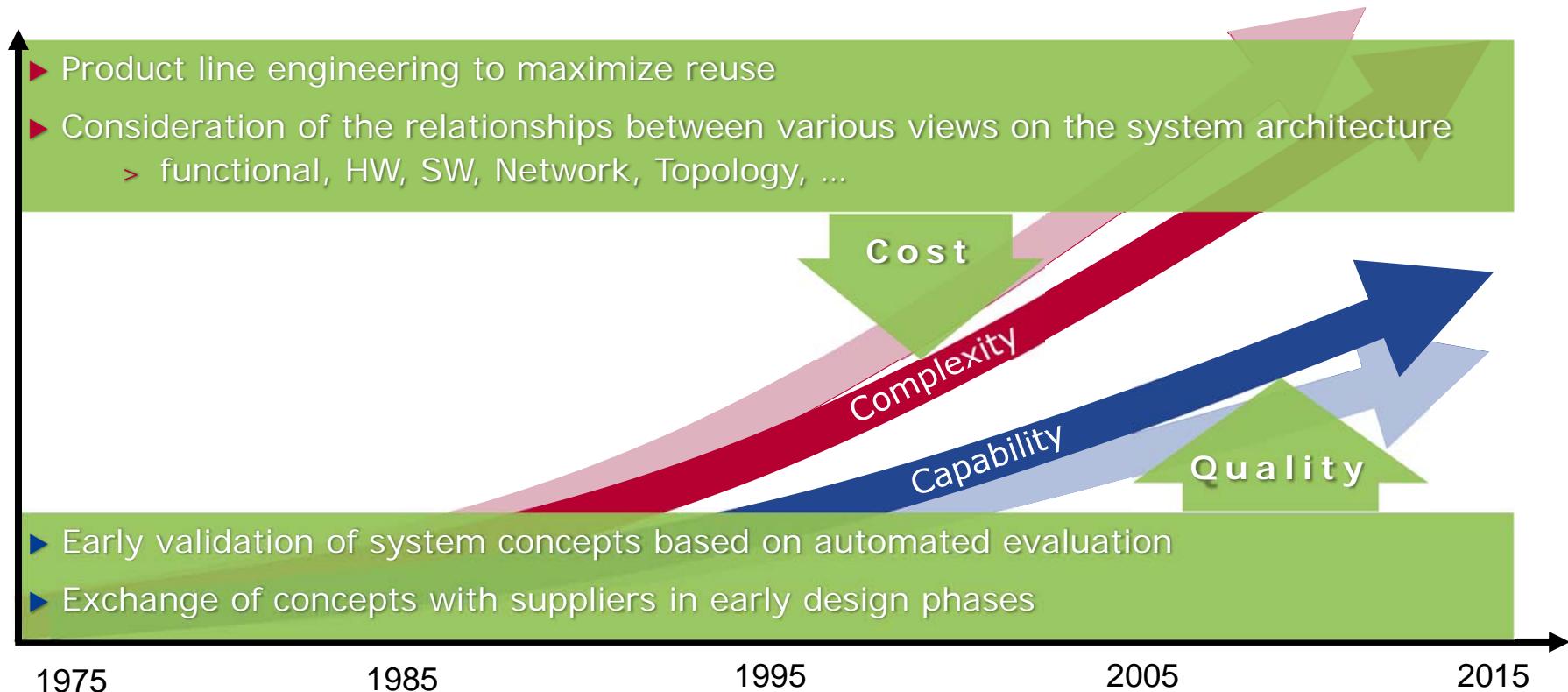
The Complexity/Capability Gap

- ▶ Number, complexity and interaction of **functions**
- ▶ Increasing **authority** over control of the vehicle
- ▶ Increasing **requirements** on certification
- ▶ **Capacity:** Number and rate of change of functions
- ▶ **Discipline:** Inappropriate processes and tools
- ▶ **Talent:** Lack of experience and topic experts

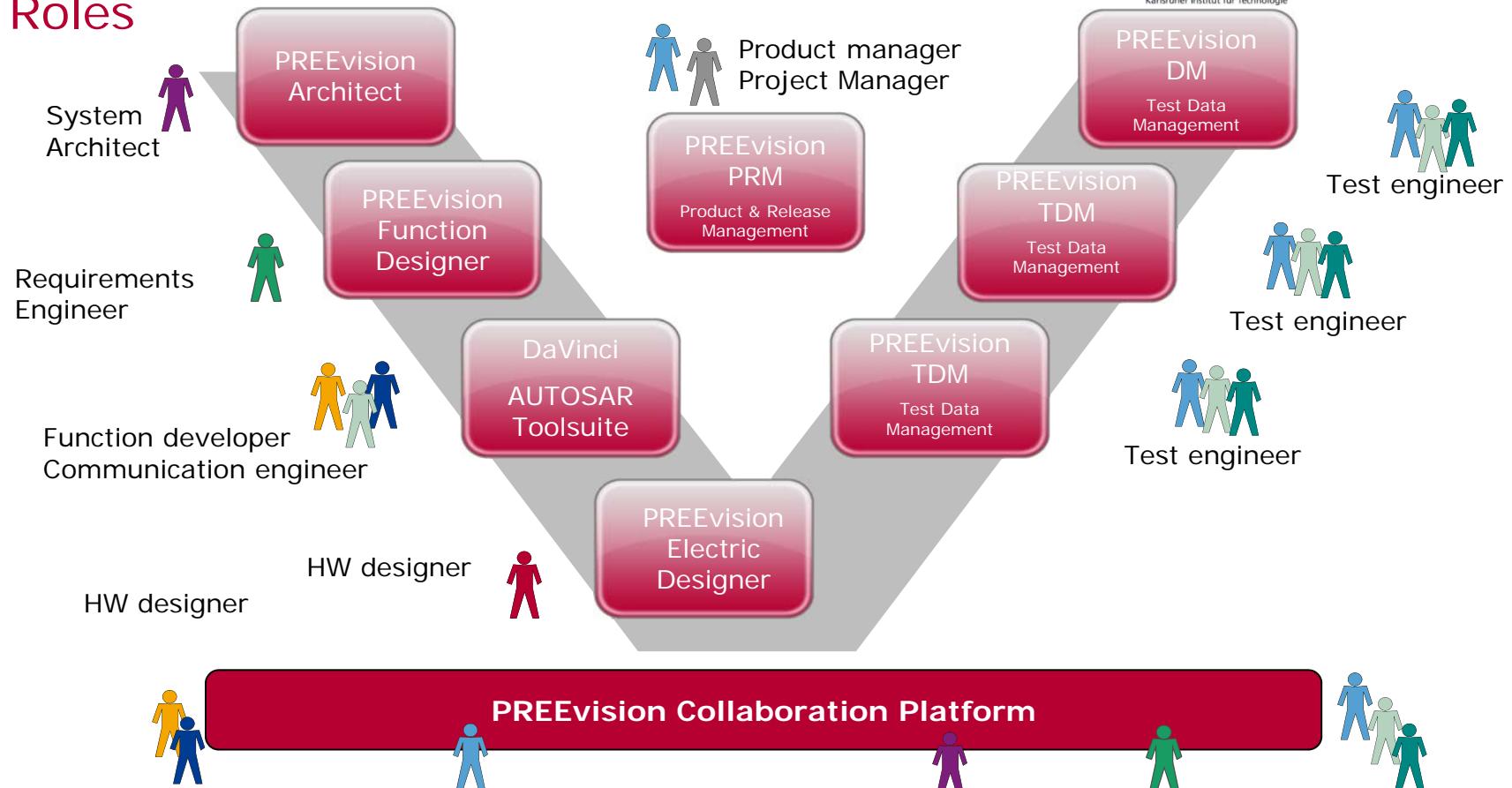


Challenges in E/E Development

The Complexity/Capability Gap

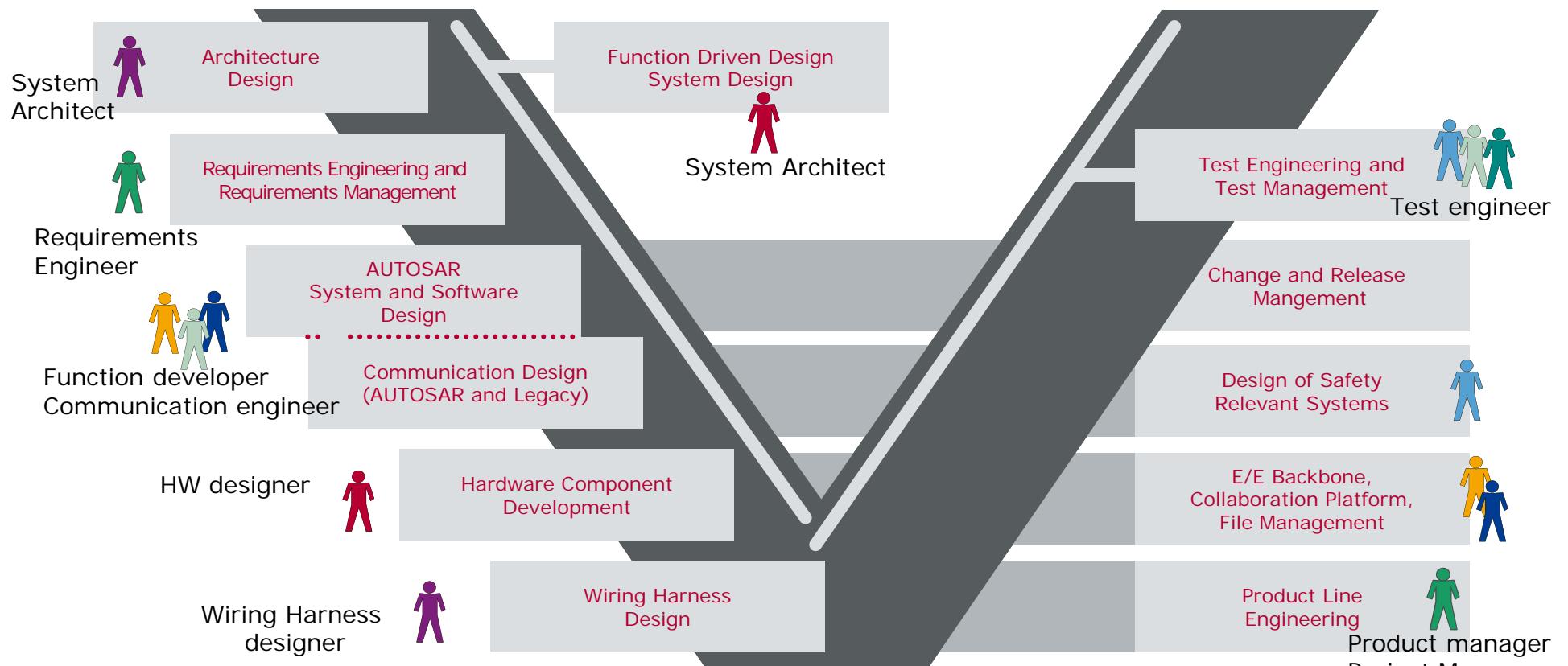


PREEvision Roles



Challenges in E/E Development

Supported E/E Engineering Use Cases and Roles

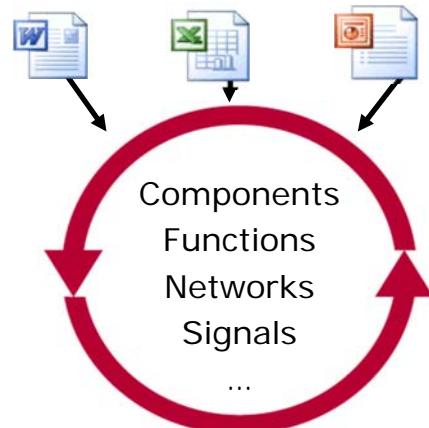


Challenges in E/E Development

Current Situation – Document Based Development Process

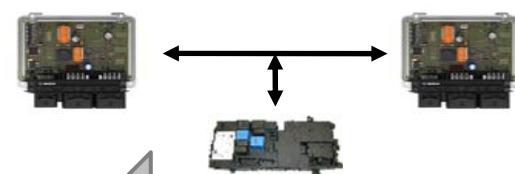
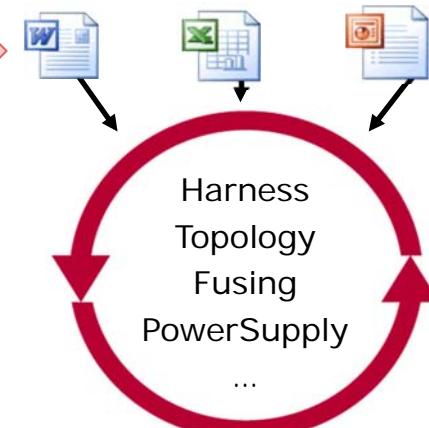


Function & Network Design Process



*Thousands(!) of
Inconsistent Documents*

Electric Design Process

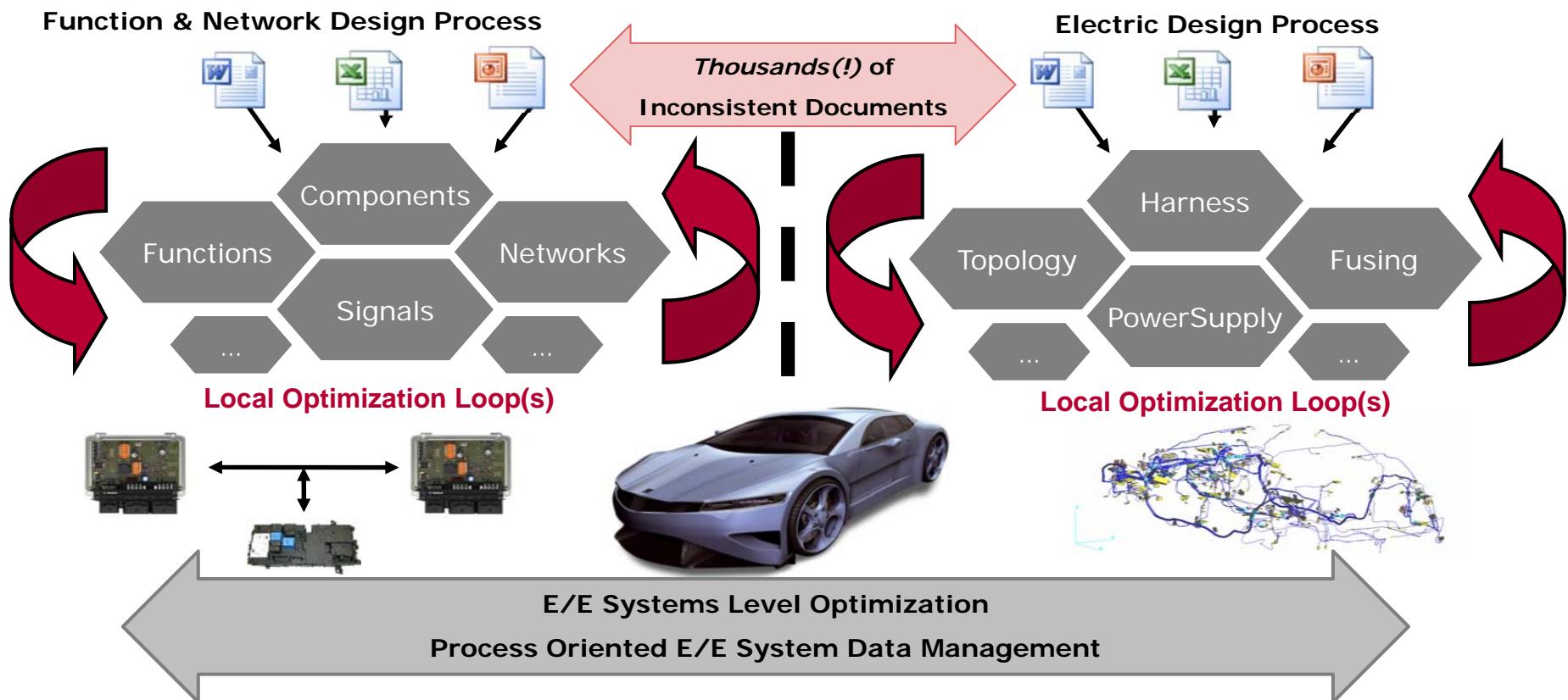


E/E Systems Level Optimization

Process Oriented E/E System Data Management

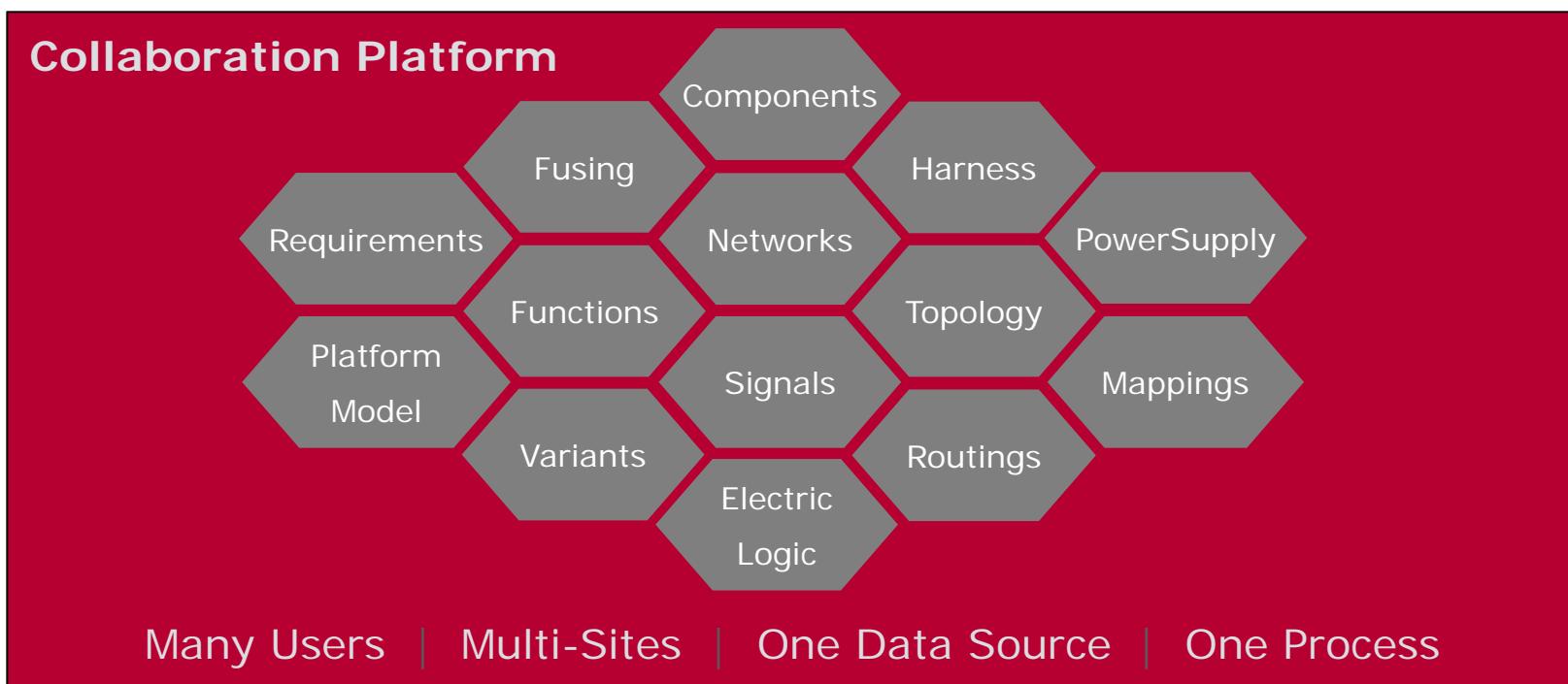
Challenges in E/E Development

E/E Backbone – Central Team Server

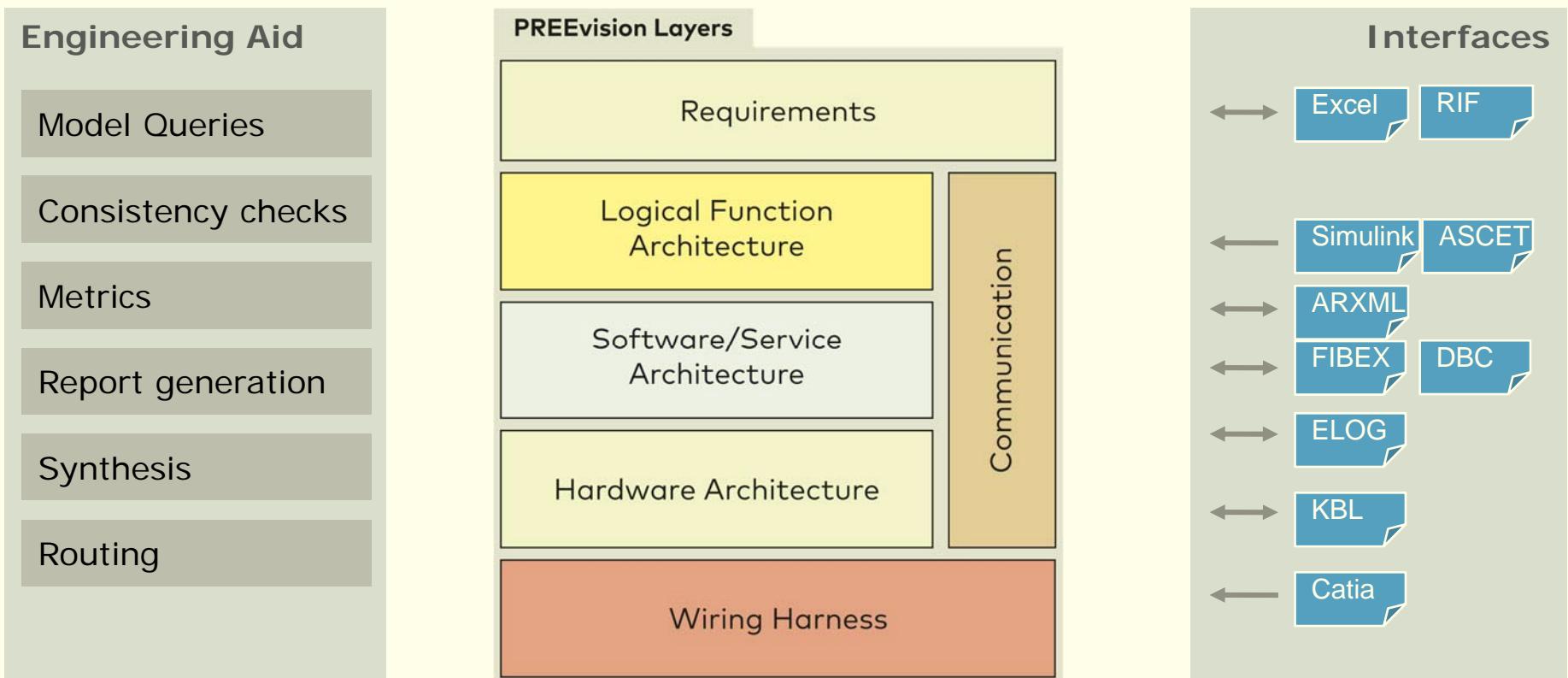


Integrated E/E Development with PREEvision
Data-Oriented E/E Development Process

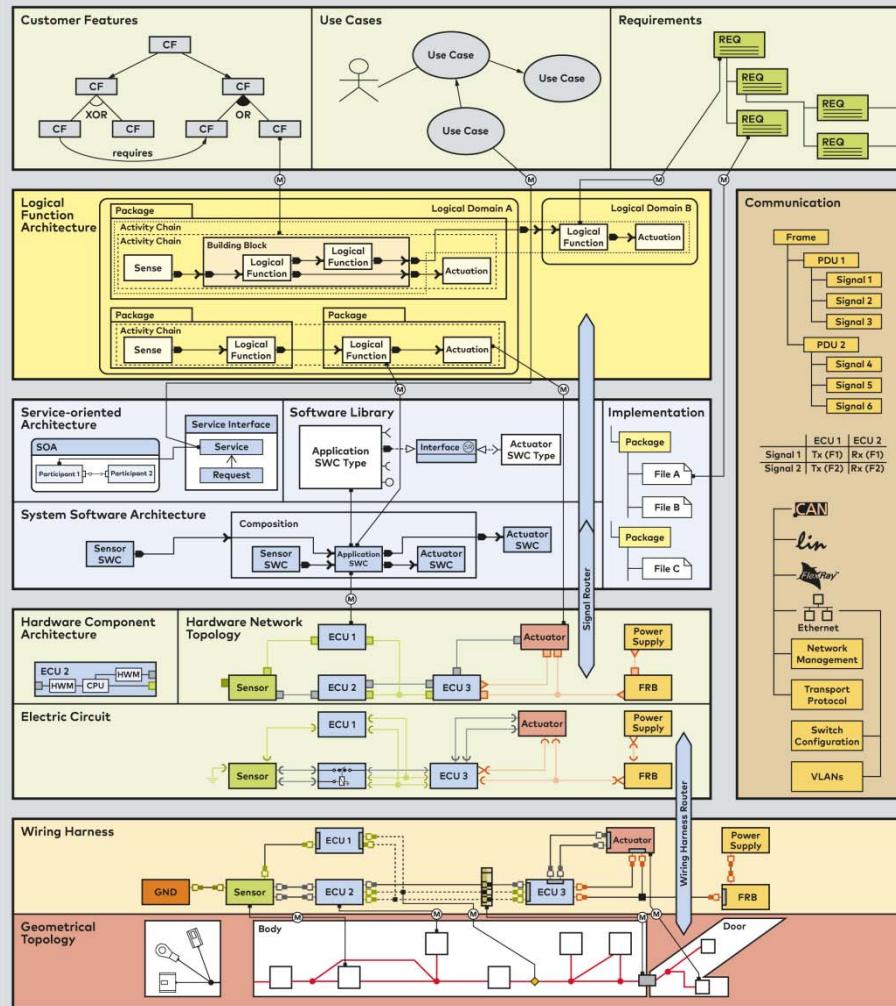
PREEVISION® One Data Model | One GUI | Full Traceability



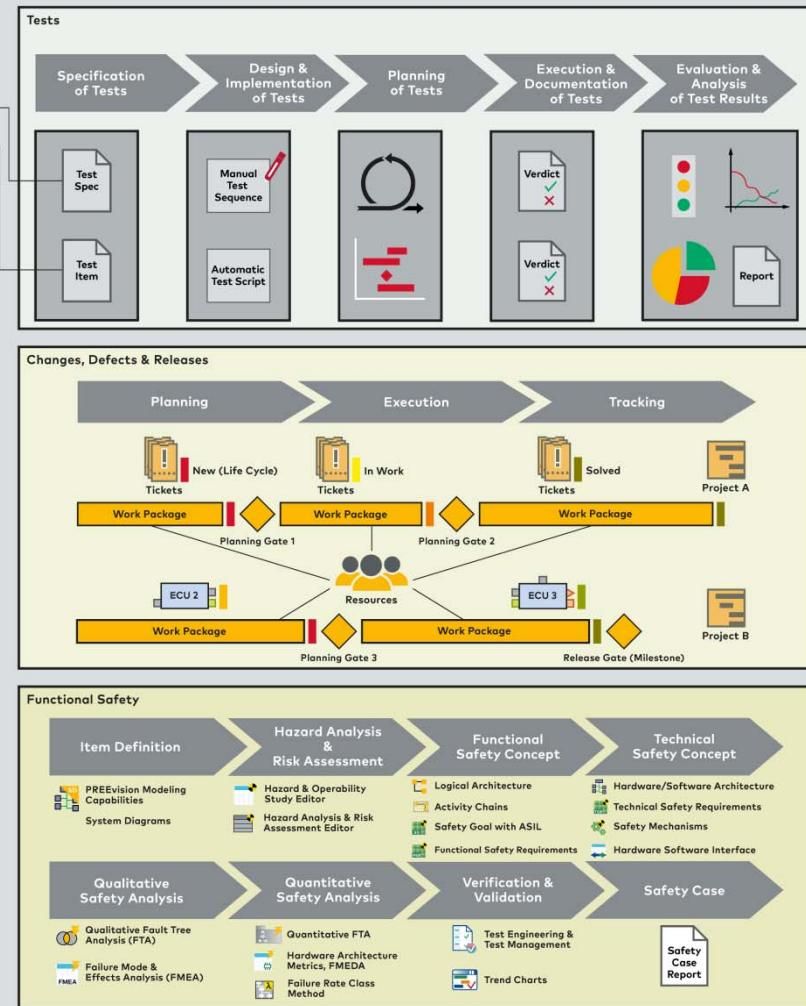
Integrated E/E Development with PREEmber Architecture Layers



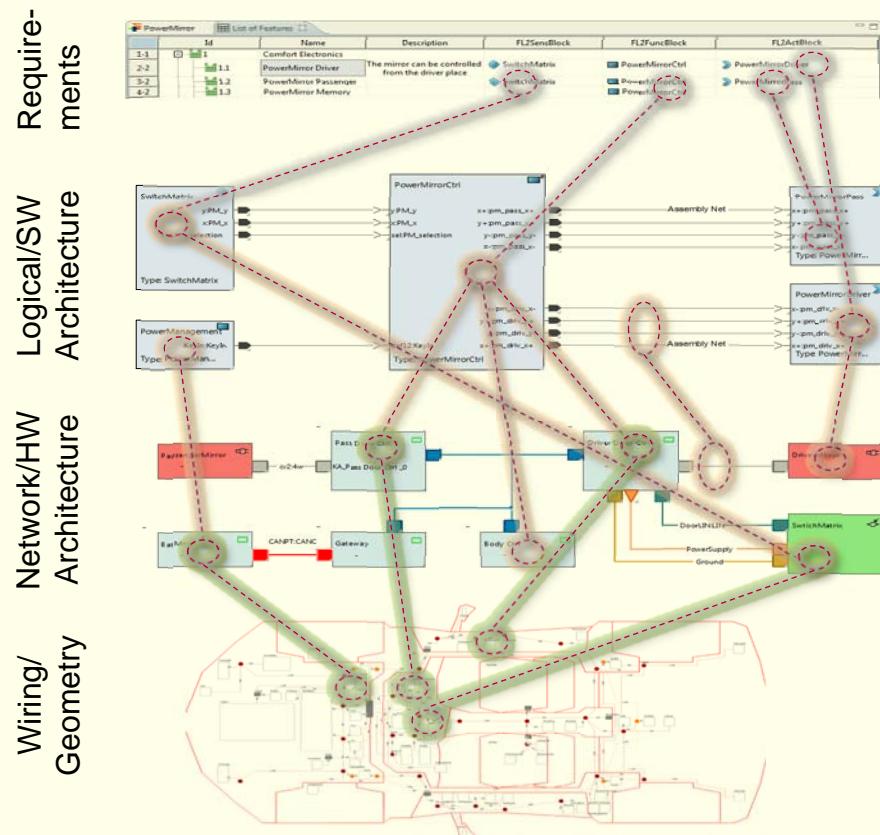
PREEvision Layers



Process & Team Support

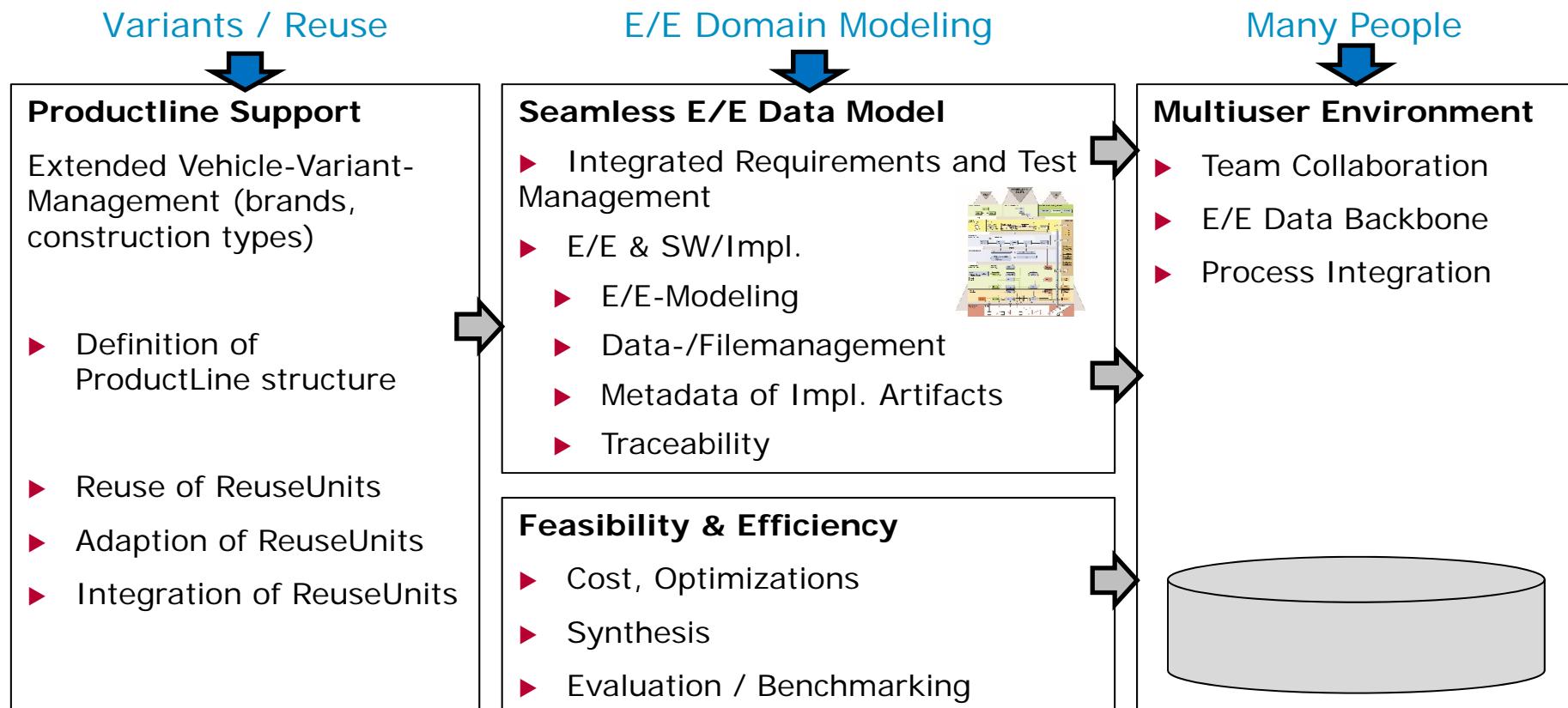


Integrated E/E Development with PREEvision Model-Based Systems Engineering



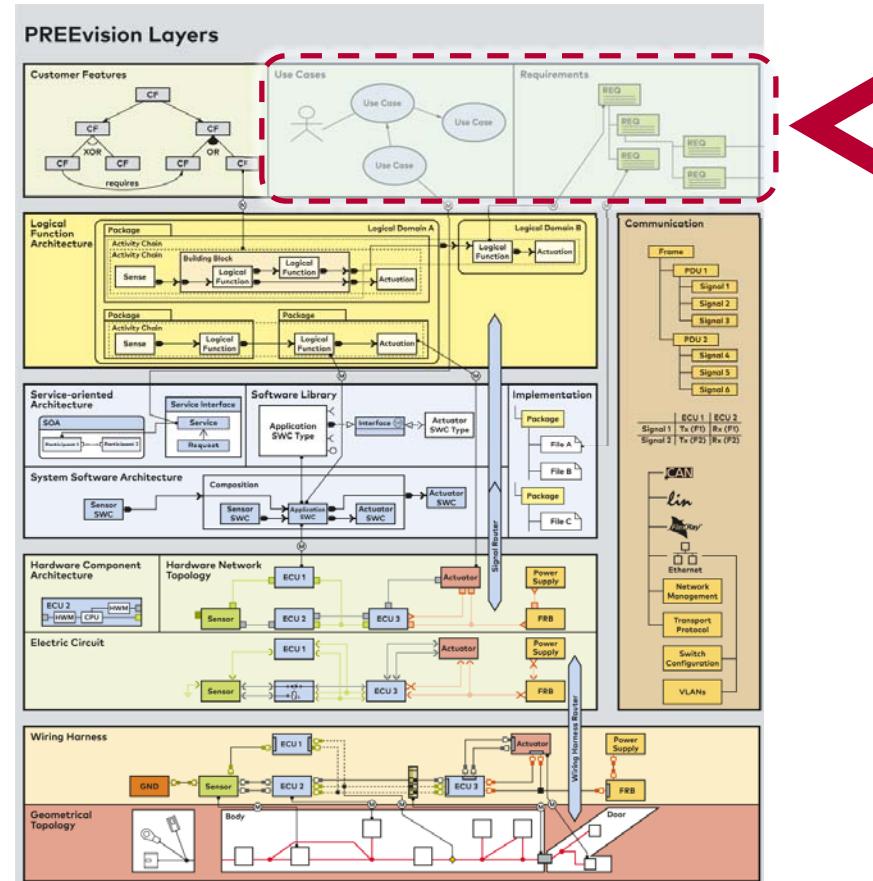
- ▶ **Domain specific** language and data model.
- ▶ **Single source model** across all development levels and disciplines
- ▶ **Collaboration** Platform
- ▶ Support for **reuse** and **product line engineering**.
- ▶ Automated **report generation** and **consistency checks**.
- ▶ **Metrics** for Benchmarking and **Automated algorithms** for scheduling, signal routing, etc.
- ▶ **Import and export** (e.g. AUTOSAR, KBL, FIBEX...)

Improvements in Handling of Complex Systems Managing Complexity



PREEvision Collaboration Platform (I)

Requirements Management



Requirements Structuring and Editing

The screenshot shows the PREVision V3.0.0 interface. On the left, the 'Model Tree' pane displays a hierarchical structure of requirements packages and requirements. On the right, the 'Requirements Table' pane shows a detailed view of requirements with columns for Level, ID, Name, and Description. A blue arrow points from the 'Model Tree' to the 'Requirements Table'.

| LEVEL | ID | Name | Description |
|-------|-----------|-----------------------------------|---|
| 2-2 | 4.1.1 | Stop | This is a description of the software requirements for automatically turning off an idle motor. |
| 3-3 | 4.1.1.1 | Conditions for turning engine off | The engine shall only be turned off under the following conditions: 1. Vehicle has been idle for more than 5 seconds 2. The last automatic turn off of the engine through stop and go is at least 2 minutes ago. ... |
| 4-3 | 4.1.1.2 | Sequence for turning engine off | The following sequence shall be followed for turning off the engine: ... |
| 5-2 | 4.1.2 | Go | This is a description of the software requirements for automatically starting an idle motor. |
| 6-3 | 4.1.2.1 | Sequence for starting | This is a description of the events required for automatically starting an idle motor. |
| 7-4 | 4.1.2.1.1 | Apply clutch | Apply the clutch before selecting gear. |
| 8-4 | 4.1.2.1.2 | Select gear | Select the first gear. |
| 9-4 | 4.1.2.1.3 | Release clutch | Release the clutch. |
| 10-4 | 4.1.2.1.4 | Give gas | Notify the motor management unit to accept acceleration commands. |

- ▶ Requirements can be **grouped** into requirements packages and **structured hierarchically**.
- ▶ **Tables** provide an efficient overview and editing capability.

Requirements Attributes and tables

The screenshot shows two windows side-by-side. The left window is titled 'Software Requirements Table' and displays a hierarchical requirements structure. The right window is titled 'Requirements Status' and lists the status of requirements mapped to a specific module.

| LEVEL | ID | Name | Description | Status | ReqMappedTo |
|-------|-----------|-----------------------------------|---|----------|------------------|
| 1-1 | 4.1 | Stop And Go | This is a description of the software requirements for automatically turning off an idle motor. | In work | PowerTrainModule |
| 2-2 | 4.1.1 | Stop | The engine shall only be turned off under the following conditions: 1. Vehicle has been idle for more than 5 seconds 2. The last automatic turn off of the engine through stop and go is at least 2 minutes ago. ... | Obsolete | PowerTrainModule |
| 3-3 | 4.1.1.1 | Conditions for turning engine off | | Released | PowerTrainModule |
| 4-3 | 4.1.1.2 | Sequence for turning engine off | The following sequence shall be followed for turning off the engine: | Reviewed | PowerTrainModule |
| 5-2 | 4.1.2 | Go | Requirements for... | In work | PowerTrainModule |
| 6-3 | 4.1.2.1 | Sequence | Required for... | In work | PowerTrainModule |
| 7-4 | 4.1.2.1.1 | Apply | ear. | In work | PowerTrainModule |
| 8-4 | 4.1.2.1.2 | Selected | | In work | PowerTrainModule |
| 9-4 | 4.1.2.1.3 | Release clutch | Release the clutch | In work | PowerTrainModule |
| 10-4 | 4.1.2.1.4 | Give gas | Notify the motor management unit to accept acceleration commands. | Obsolete | PowerTrainModule |

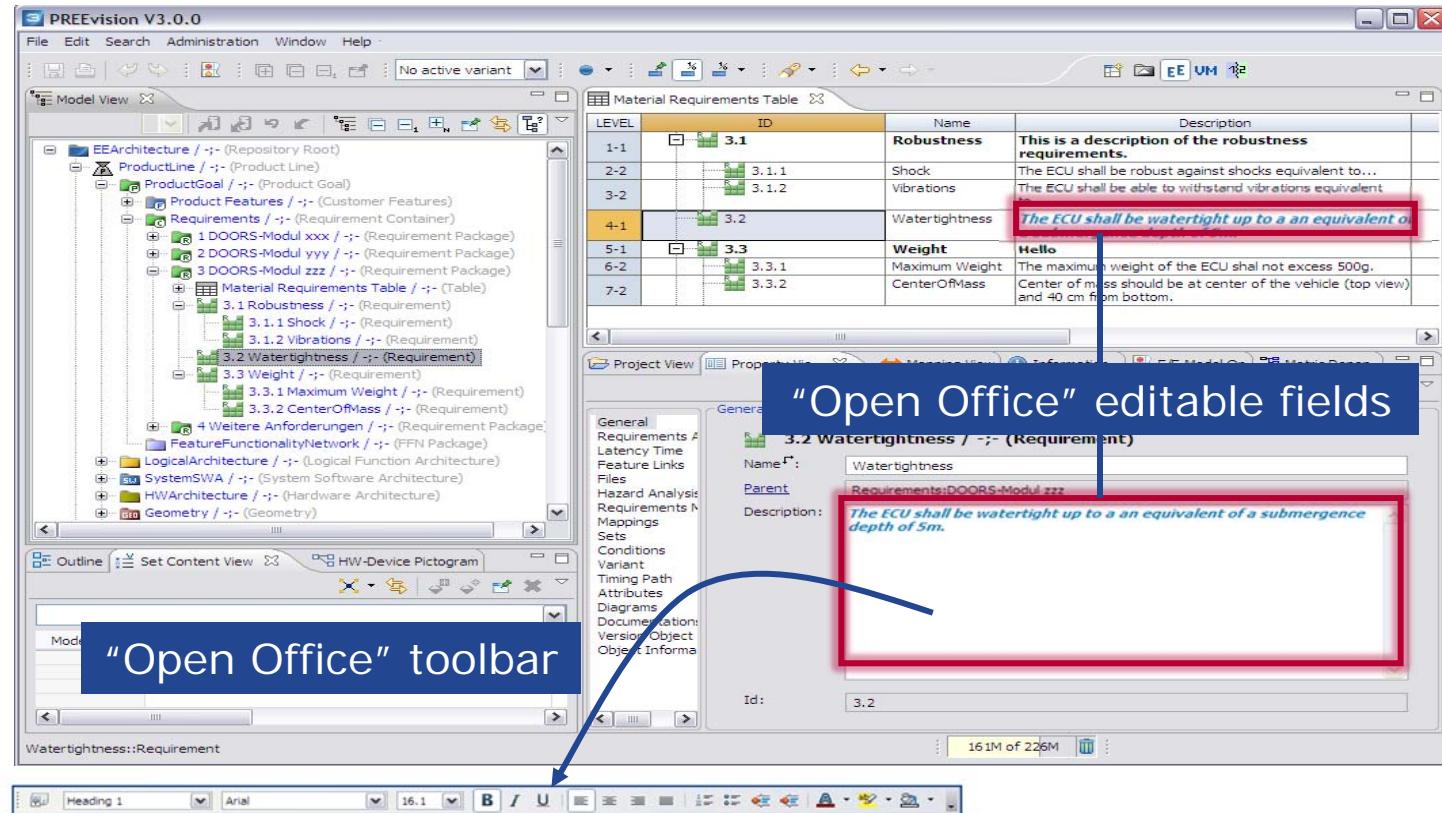
Enumeration Requirements Attribute

Boolean Requirements Attribute

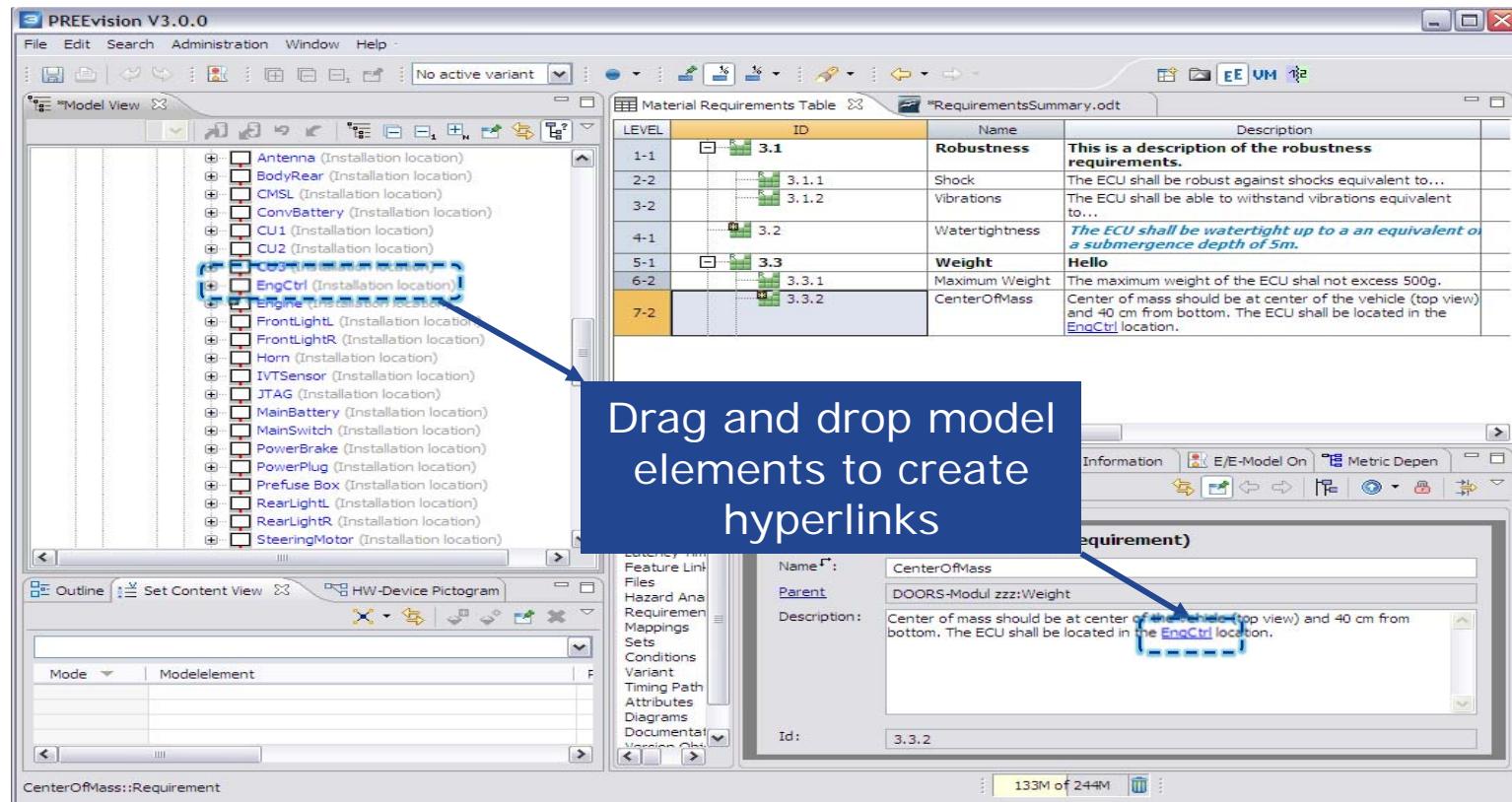
Model Queries

- ▶ Requirements can be extended with **user-defined attributes** that can be directly edited in the requirements tables.
- ▶ Attributes can be typed, e.g. Boolean, Enumeration, Integers with **valid value ranges**,..., and are handled accordingly in tables.
- ▶ Requirements tables can display the results of **model queries**. (E.g. ECUs to which the requirements are mapped.)

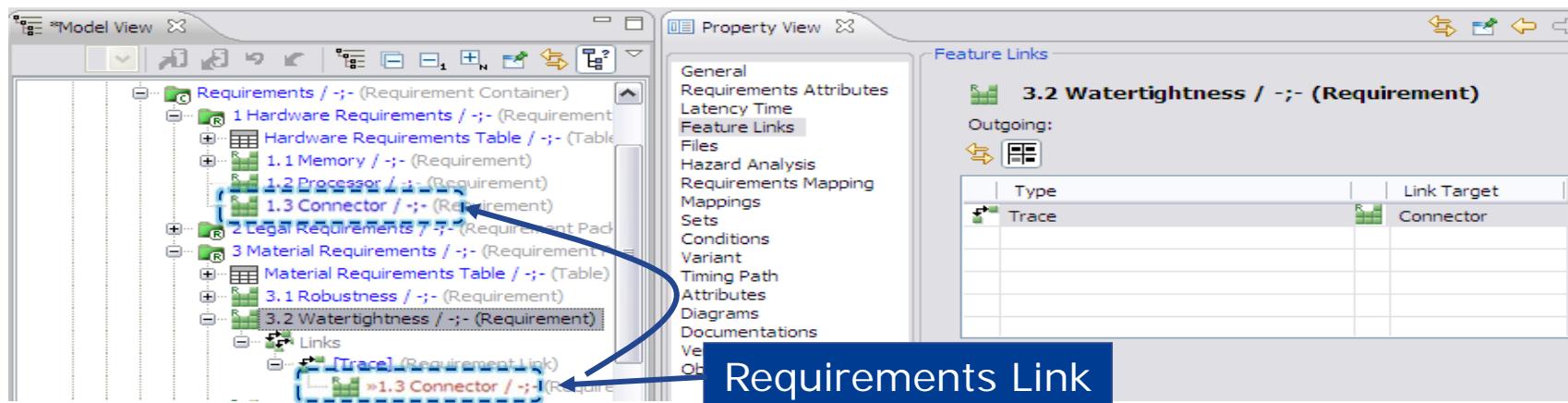
Open Office Integration Editable Fields



Open Office Integration Hyperlinks

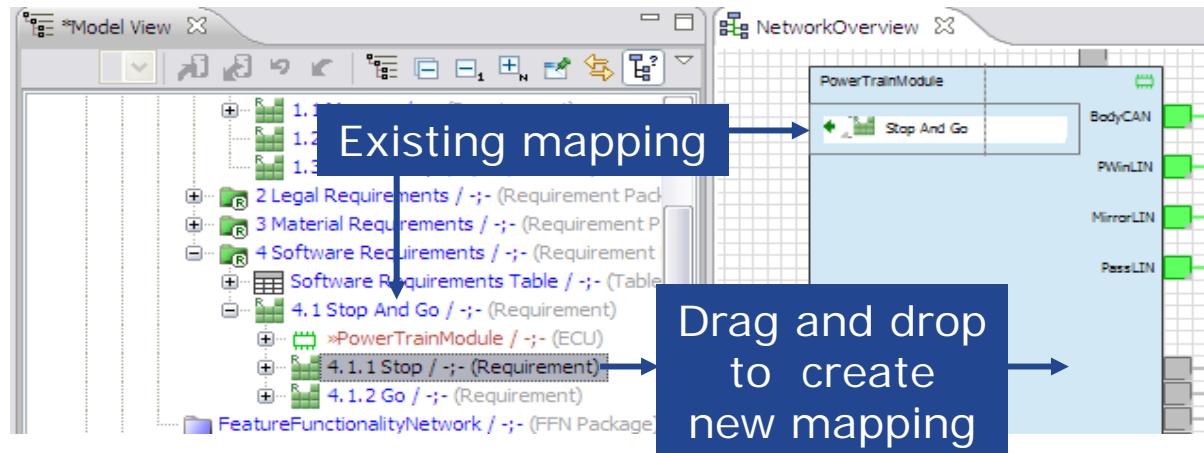


Requirements Linking



- ▶ Requirements can be **linked** to other requirements to **maintain traceability**.
- ▶ Linked requirements are shown in the **model tree** and **property view**.
- ▶ User can **directly navigate** to linked requirements by pressing the space bar.

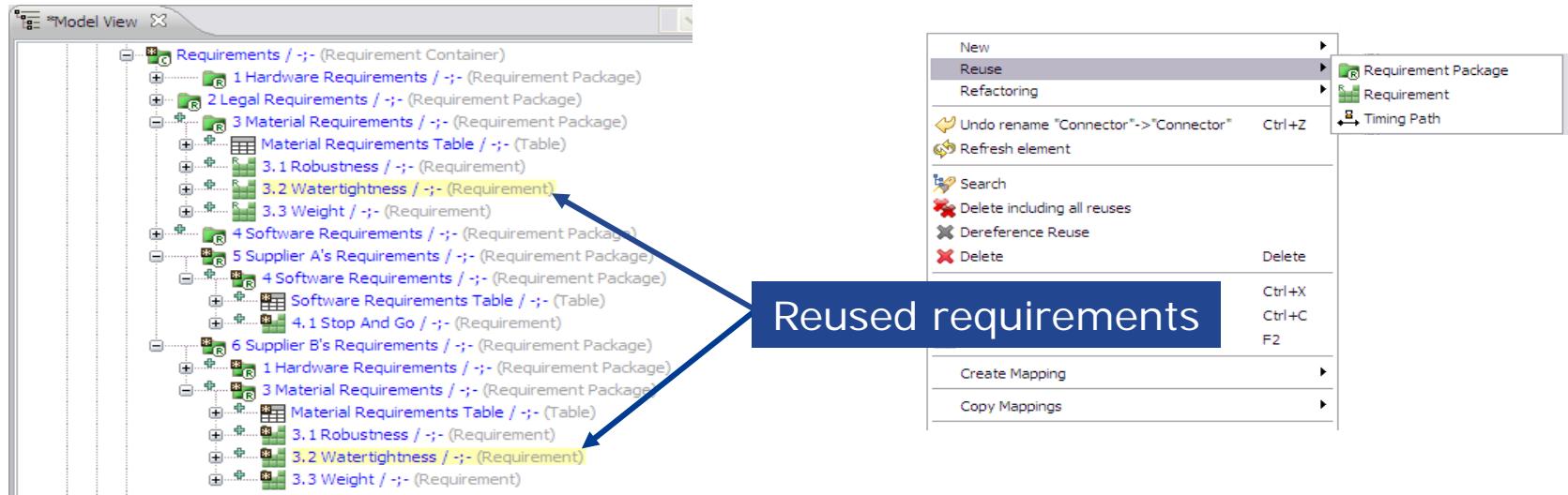
Requirements Mapping



- ▶ Requirements can be **directly mapped** to architecture artefacts
E.g. Logical functions, SW Functions, ECUs, Hardware modules,...
- ▶ Mappings can be created via **drag and drop** between requirements in the model tree and graphical elements.
- ▶ The **results of the mappings** can be **displayed** in the diagrams and model tree.

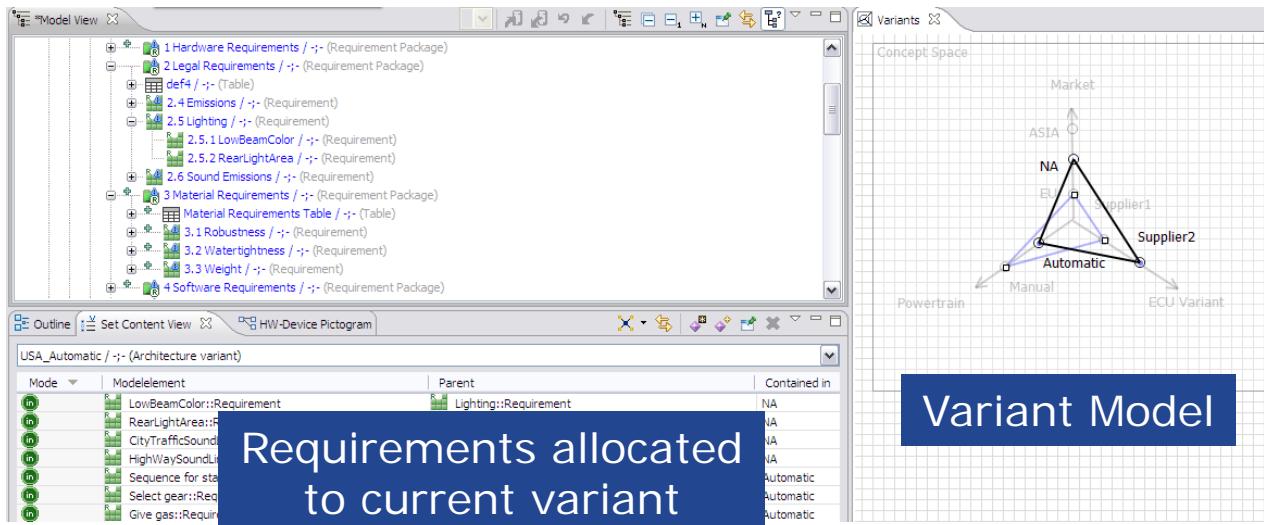
Requirements

Reuse of Requirements



- ▶ Requirements, Requirements Packages and Timings can be **reused** in a number of contexts.
- ▶ **Changes** in the description are **copied across all reuses**.
- ▶ Reuse can be used to efficiently **manage different groupings** of the same requirements, e.g. for different suppliers.

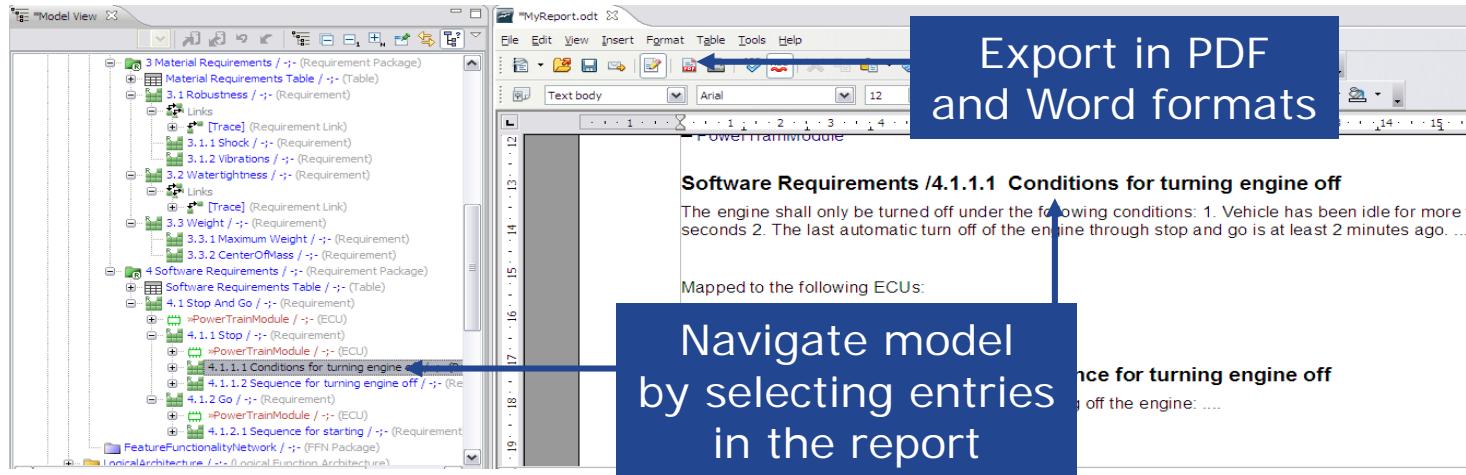
Requirements Variant Management



Variant Model

- ▶ Requirements can be allocated to sets and alternatives defined in the variant model.
- ▶ Requirements **not allocated** to the current selected variant are indicated in the model tree (green)
- ▶ Can be used to **manage variant-specific requirements**.

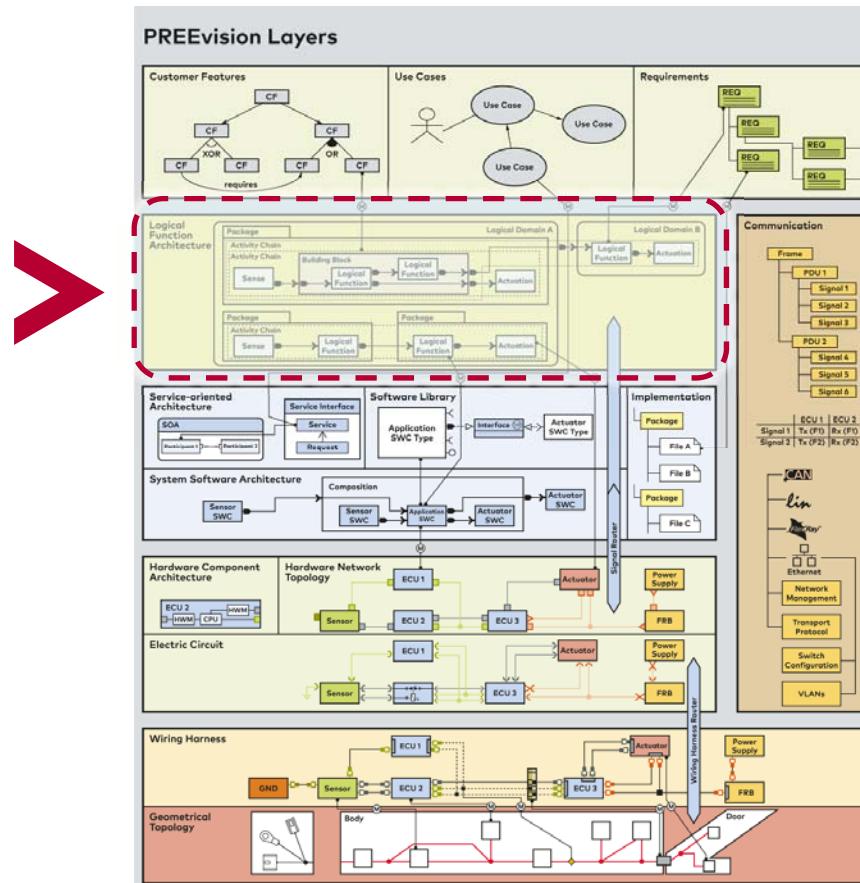
Report Generation



- ▶ Reports can be generated based on **user defined templates**.
- ▶ High level of **flexibility** in the report format including the use of tables, diagrams and complex **model queries**.
- ▶ Generation of **variant-specific** reports possible (e.g. to create supplier specific specifications in PDF).

PREEvision Collaboration Platform (II)

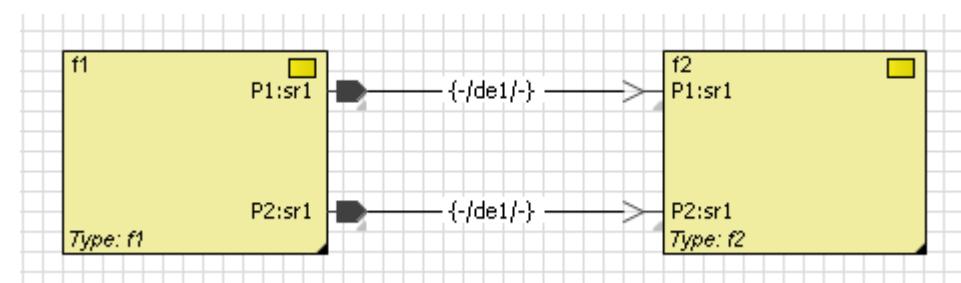
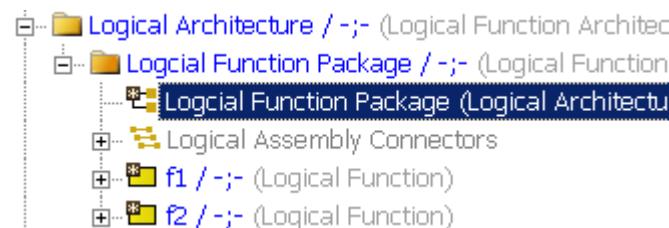
Logische Architektur



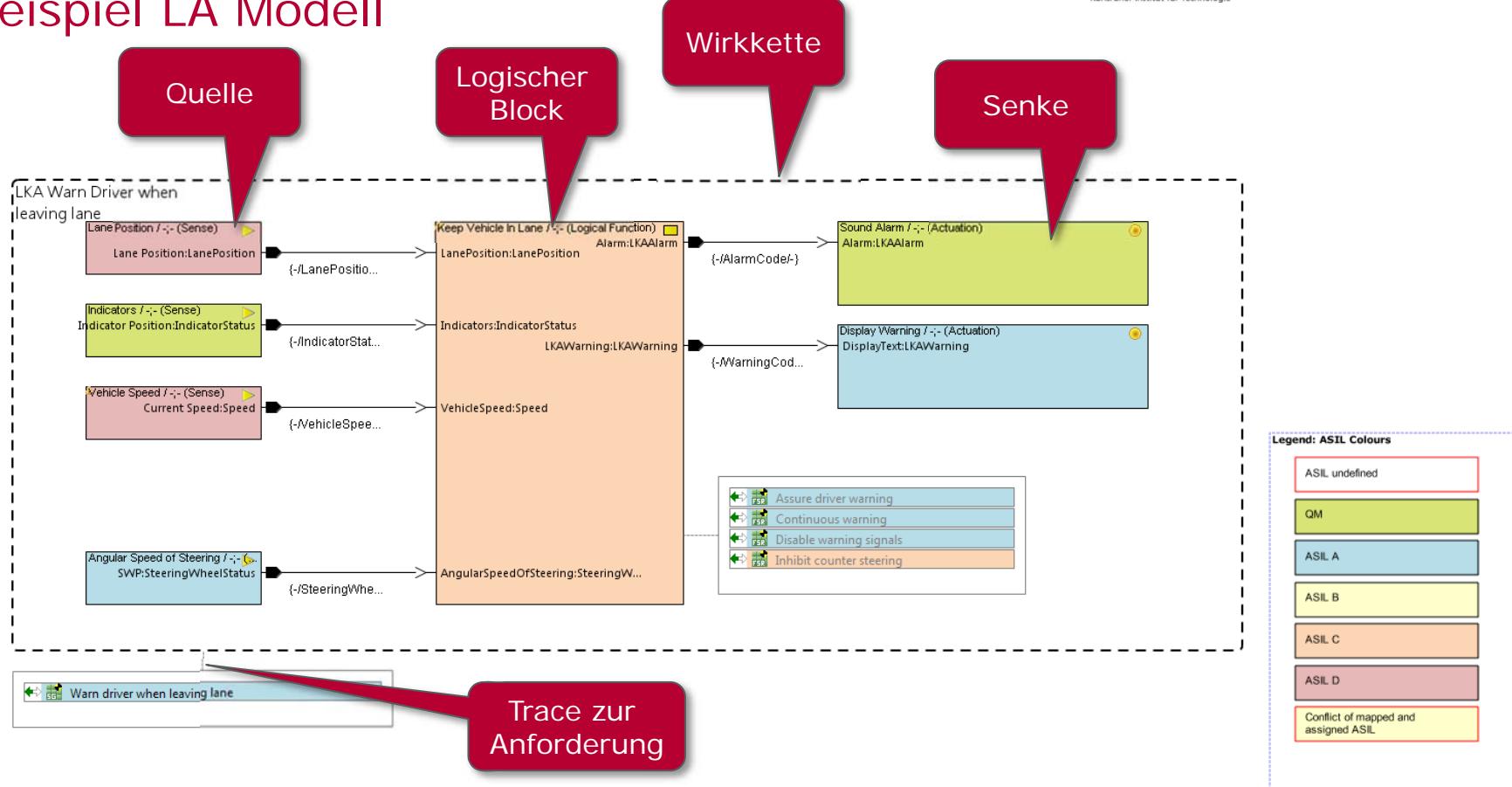
Logical Architecture Layer

Logical Architecture

- ▶ The network communication specification for distributed systems is typically driven by the **Logical Architecture** and its mapping to the **Hardware Architecture**
- ▶ The **Logical Architecture** specification is supported by graphical block diagrams
- ▶ The communication between logical functions in the **Logical Architecture** is specified by ports and connections (in a similar way as AUTOSAR)

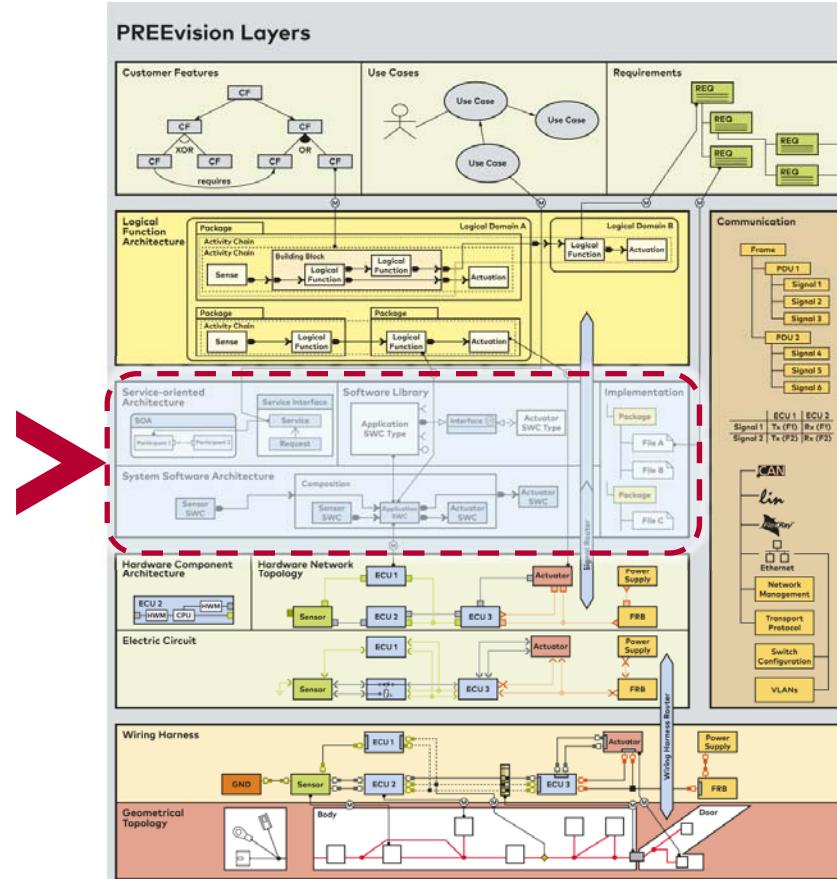


Logical Architecture Layer Beispiel LA Modell

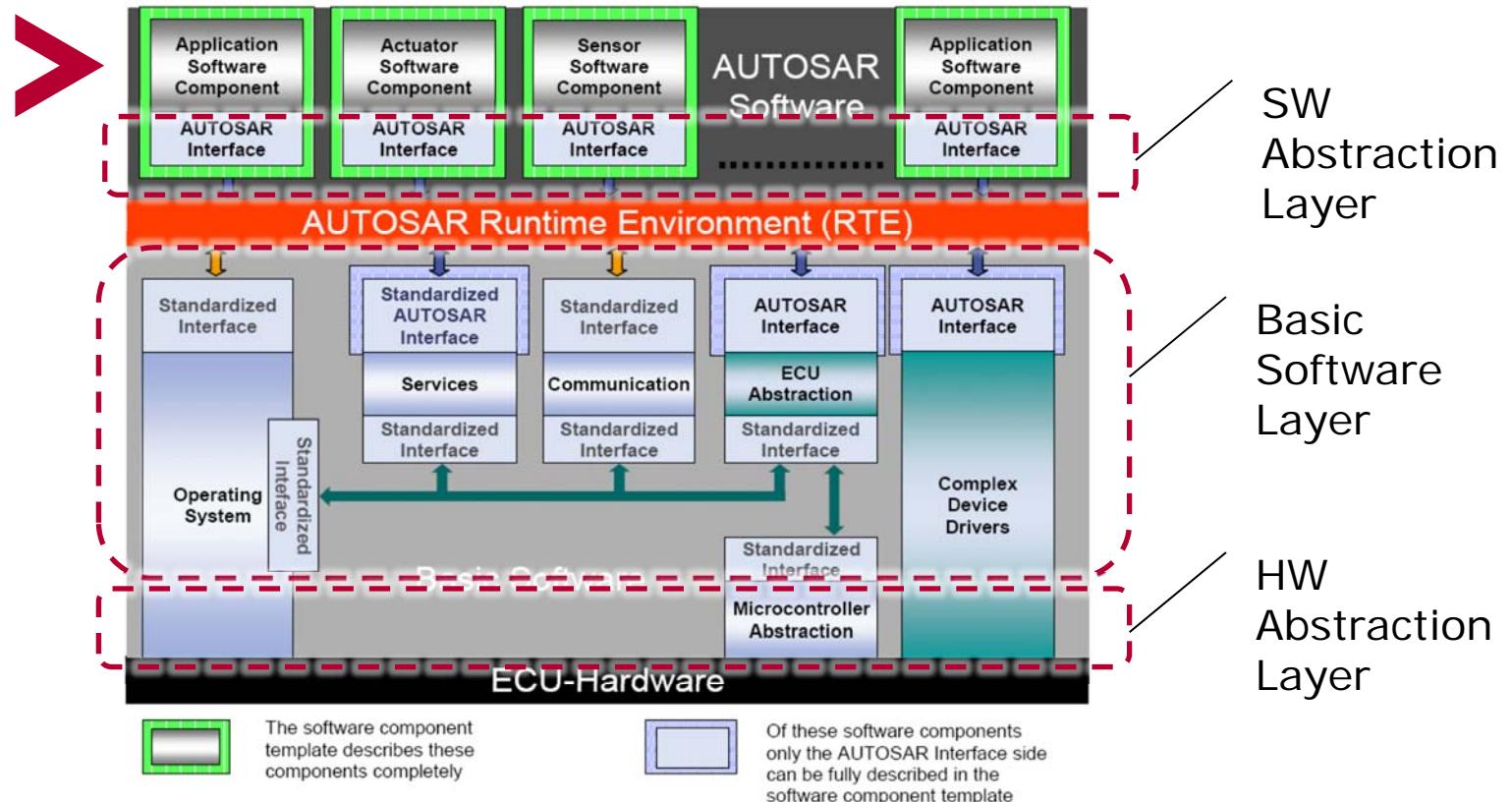


PREEvision Collaboration Platform (II)

SW Architektur

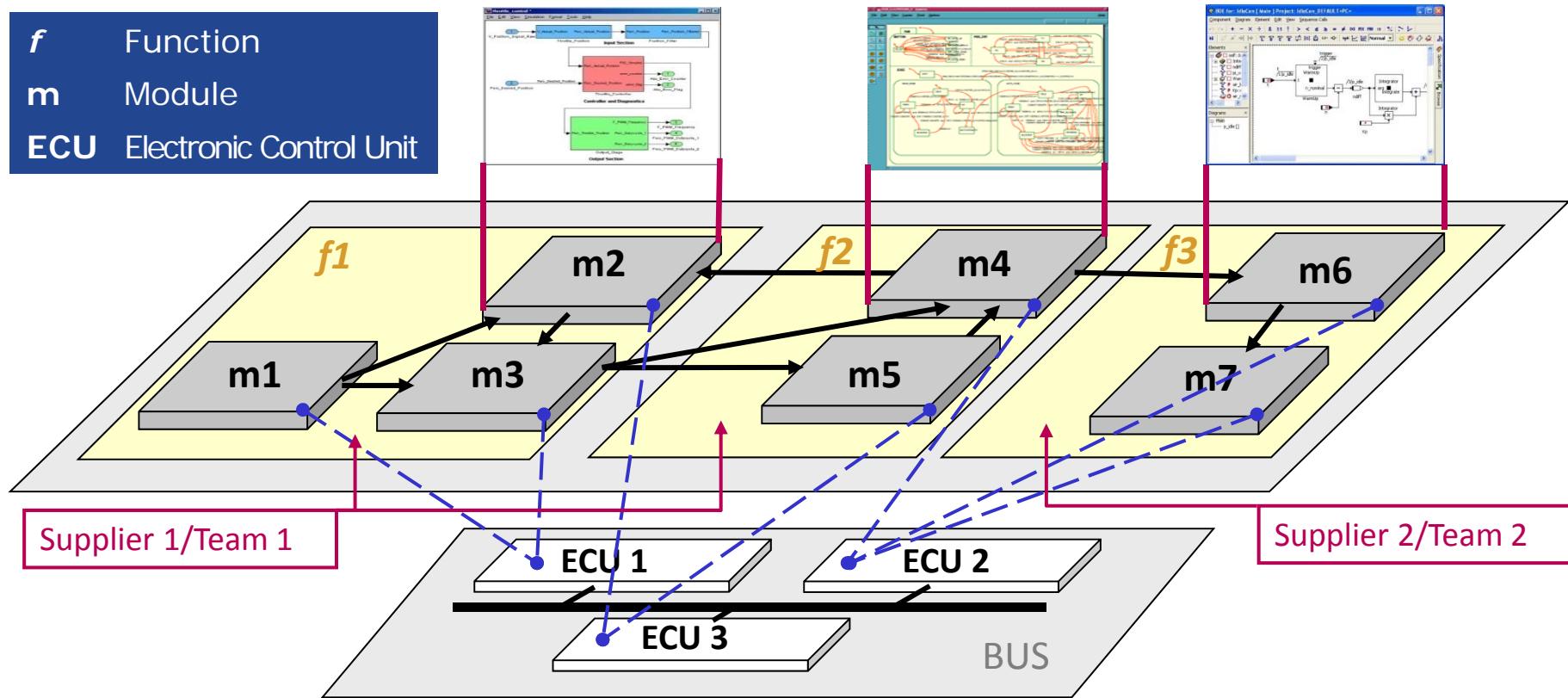


AUTOSAR (ECU and Software) Scope in the ECU SW Architecture

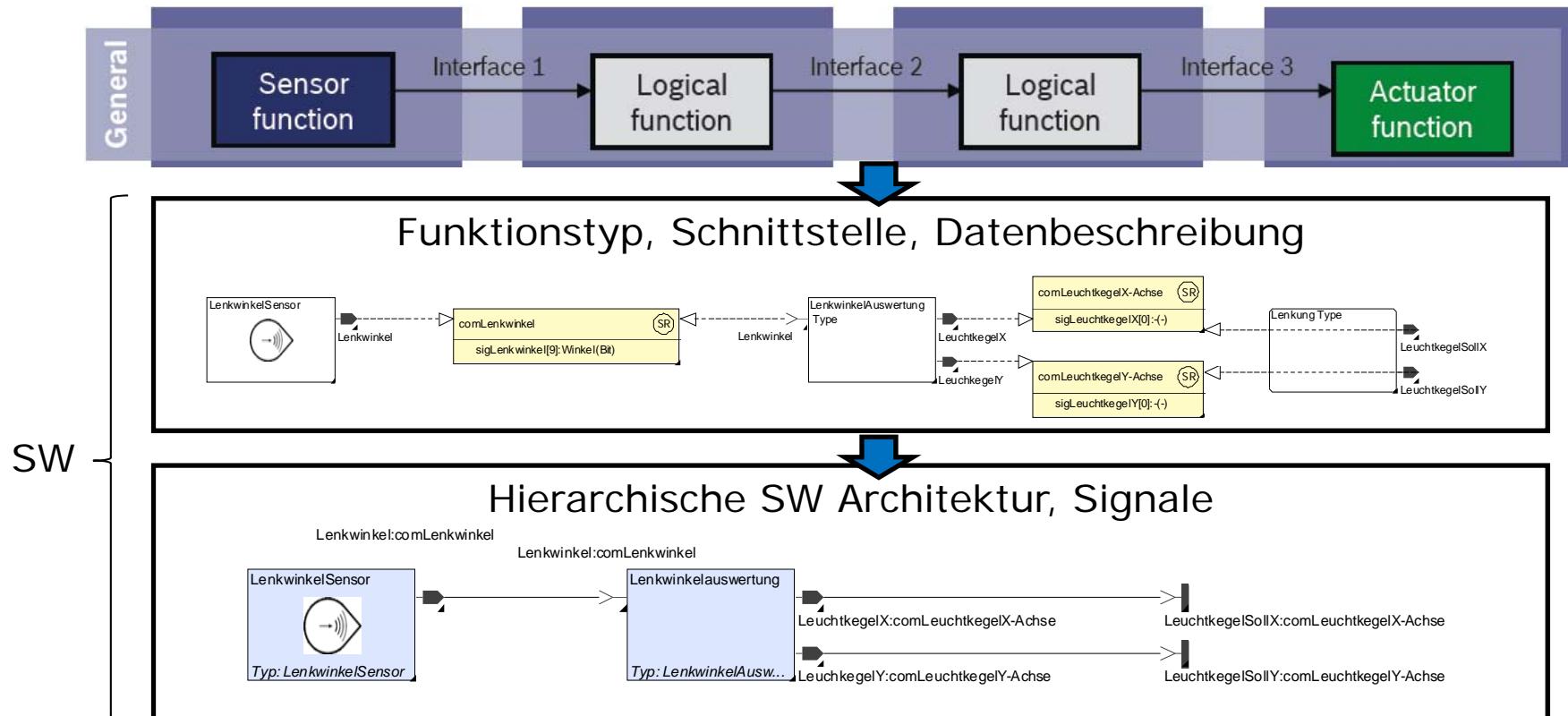


Function Oriented System Development

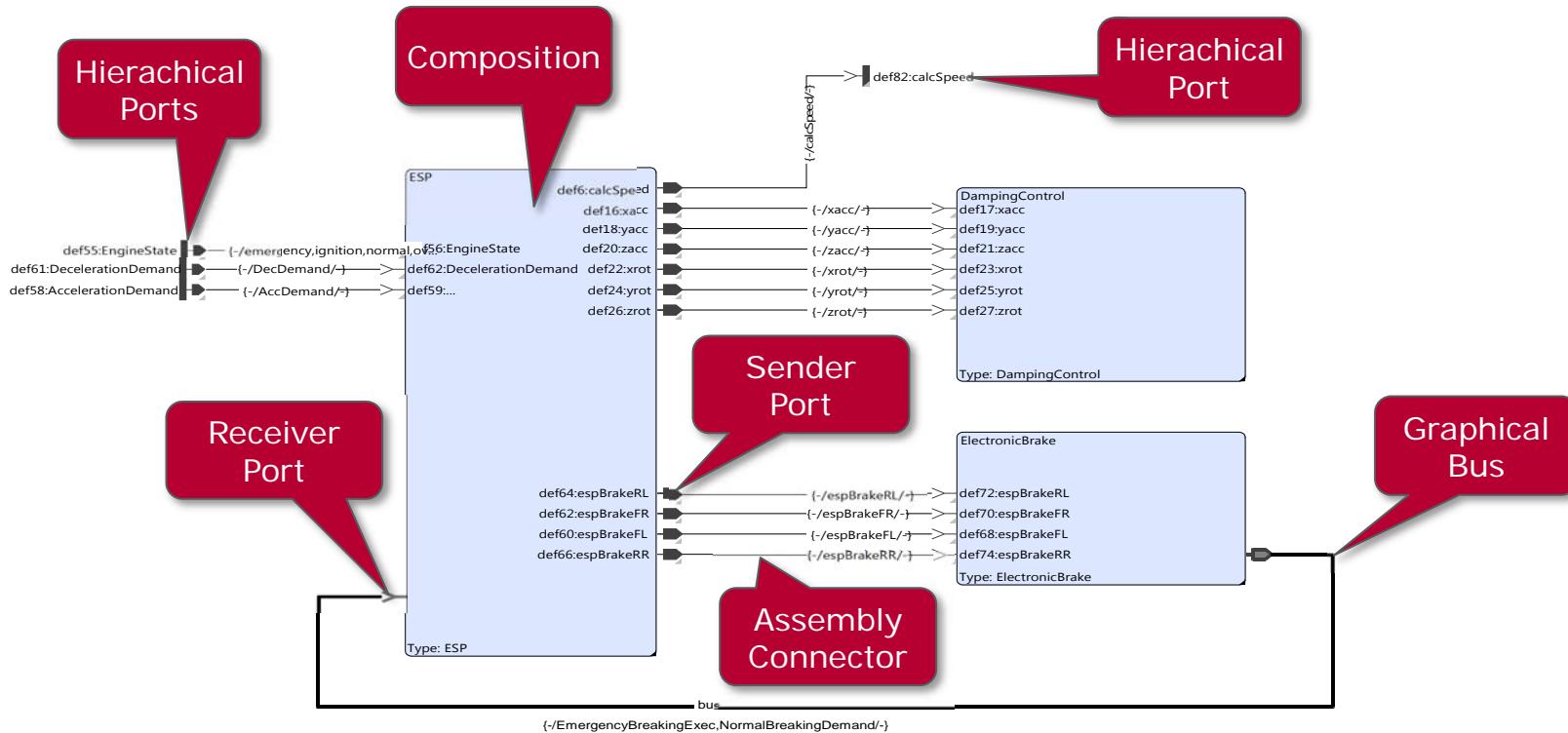
f Function
m Module
ECU Electronic Control Unit



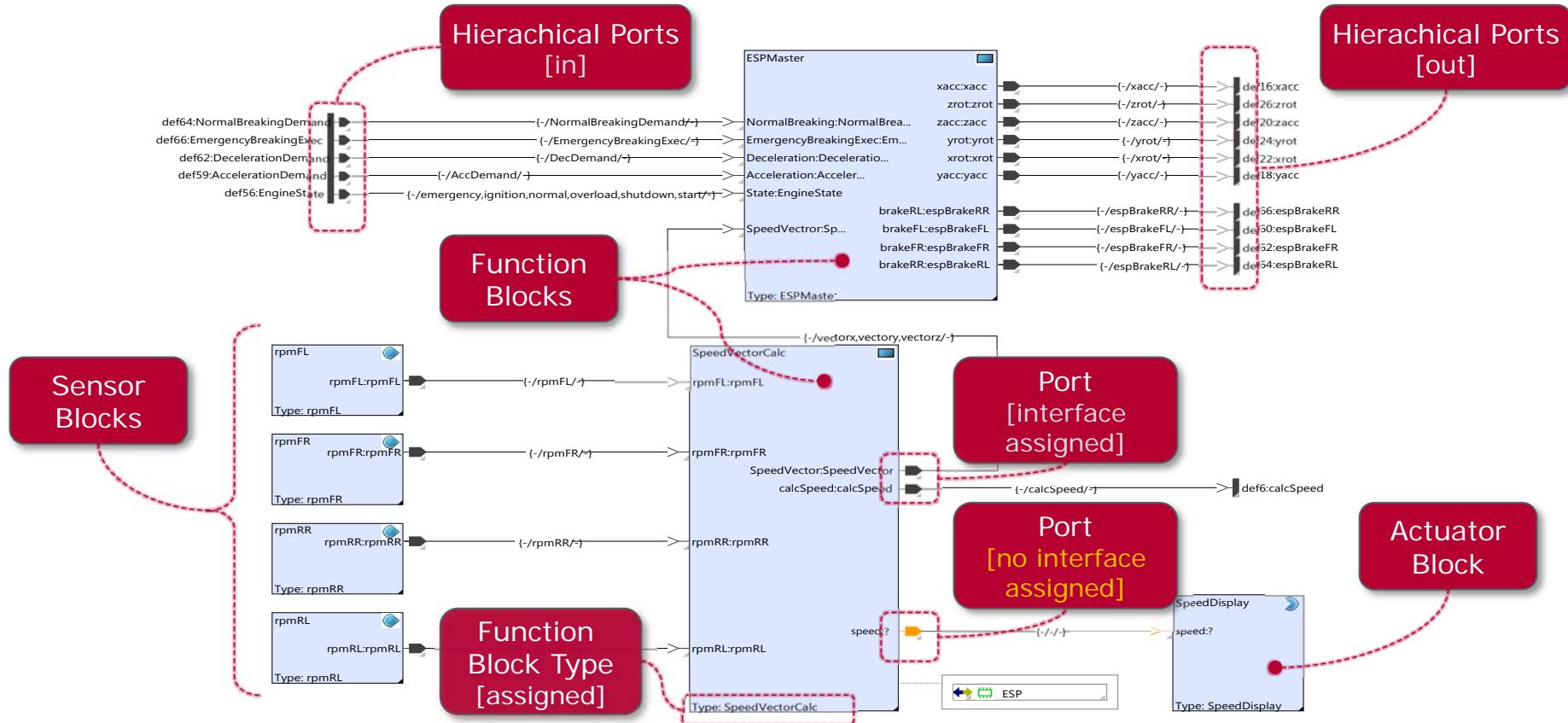
SW Architektur



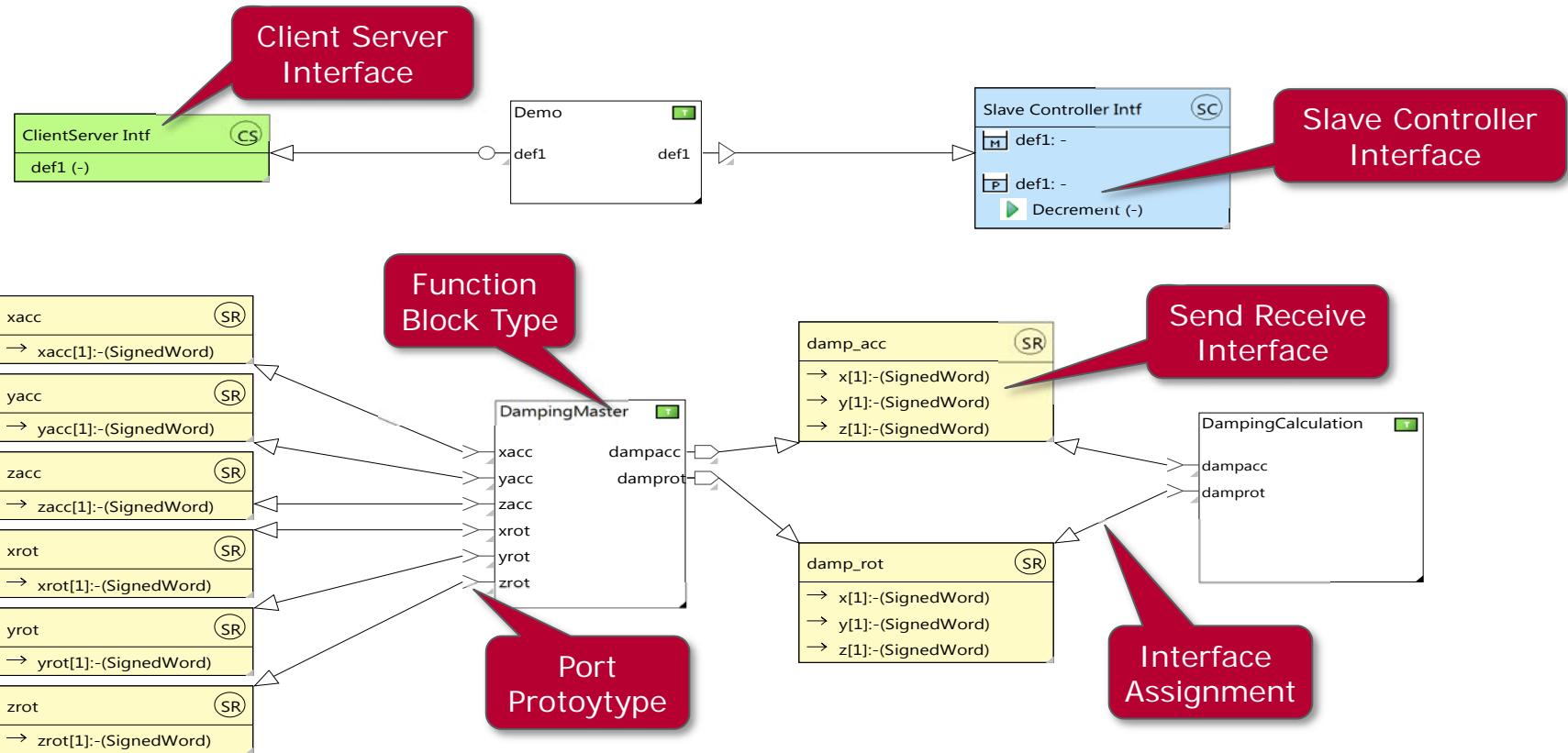
Function / Function Network Layer (2) Ports and Connectors



Function / Function Network Layer (3) Ports and Blocks



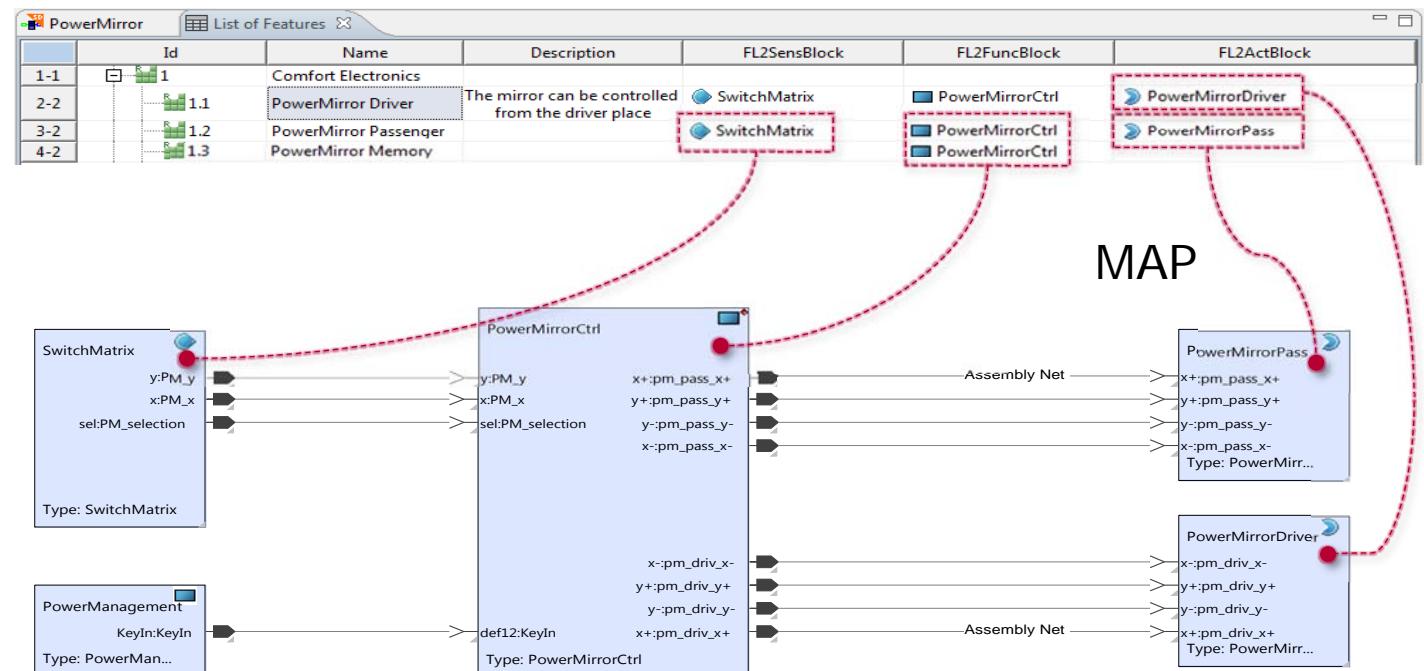
Function / Function Network Layer (4) Interfaces



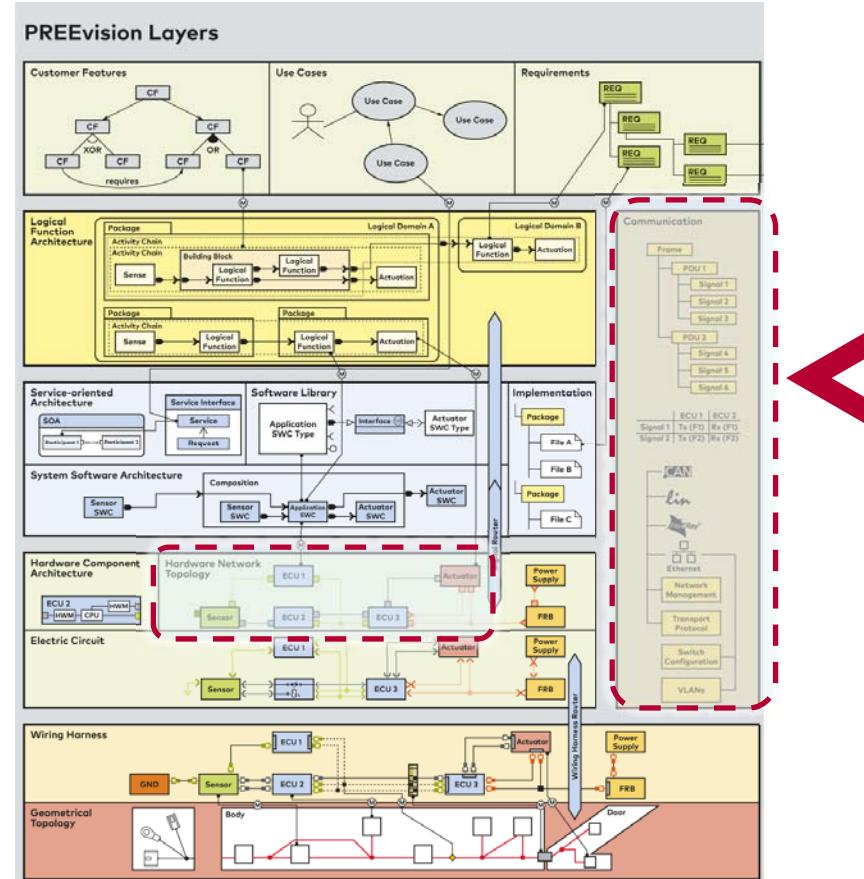
Requirements
Feature List



Function
Network

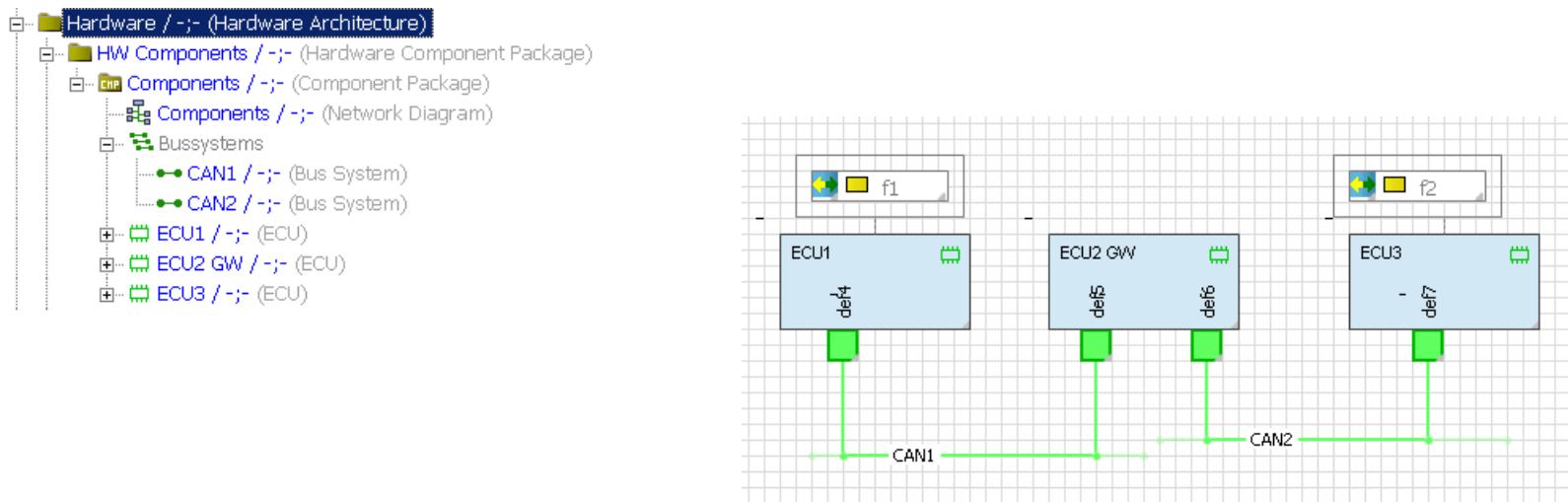


PREEvision Collaboration Platform (III) Vernetzungs-Architektur

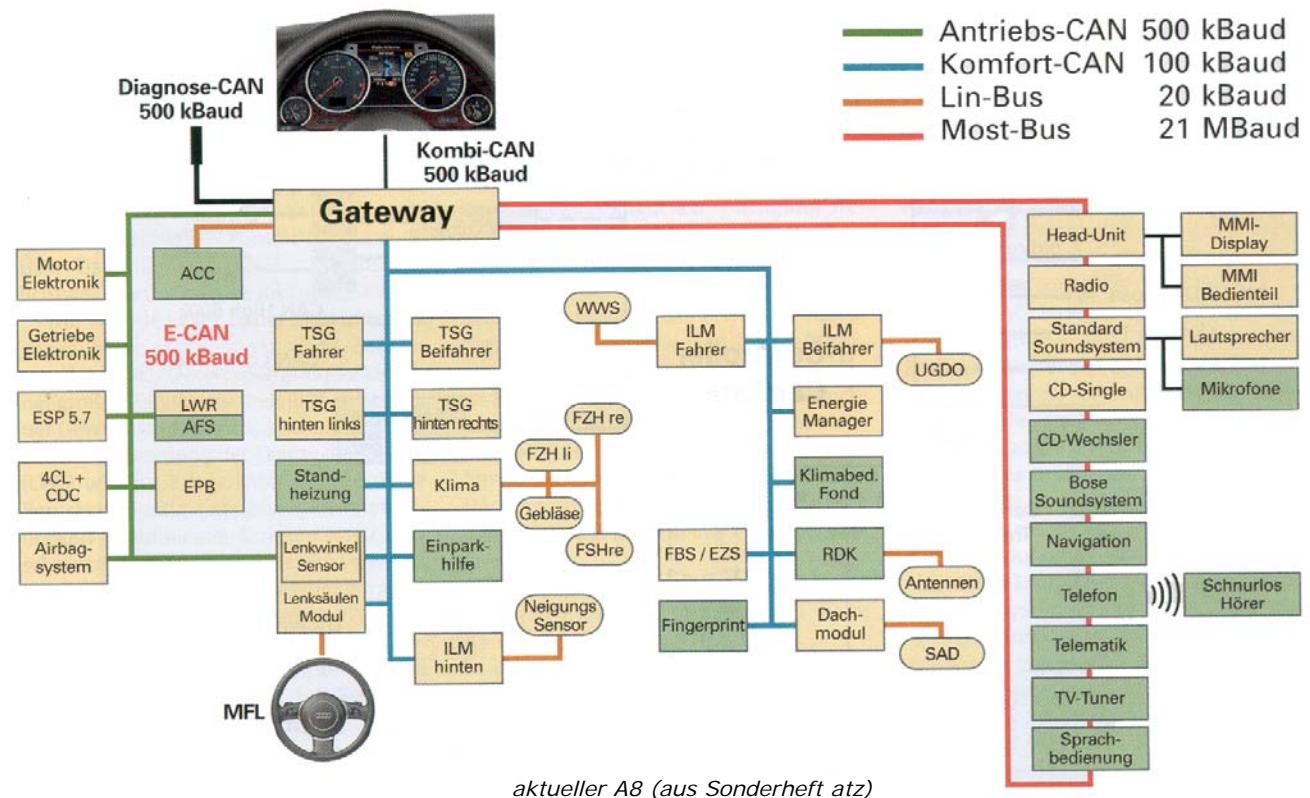


Communication Layer in PREEvision Hardware Architecture

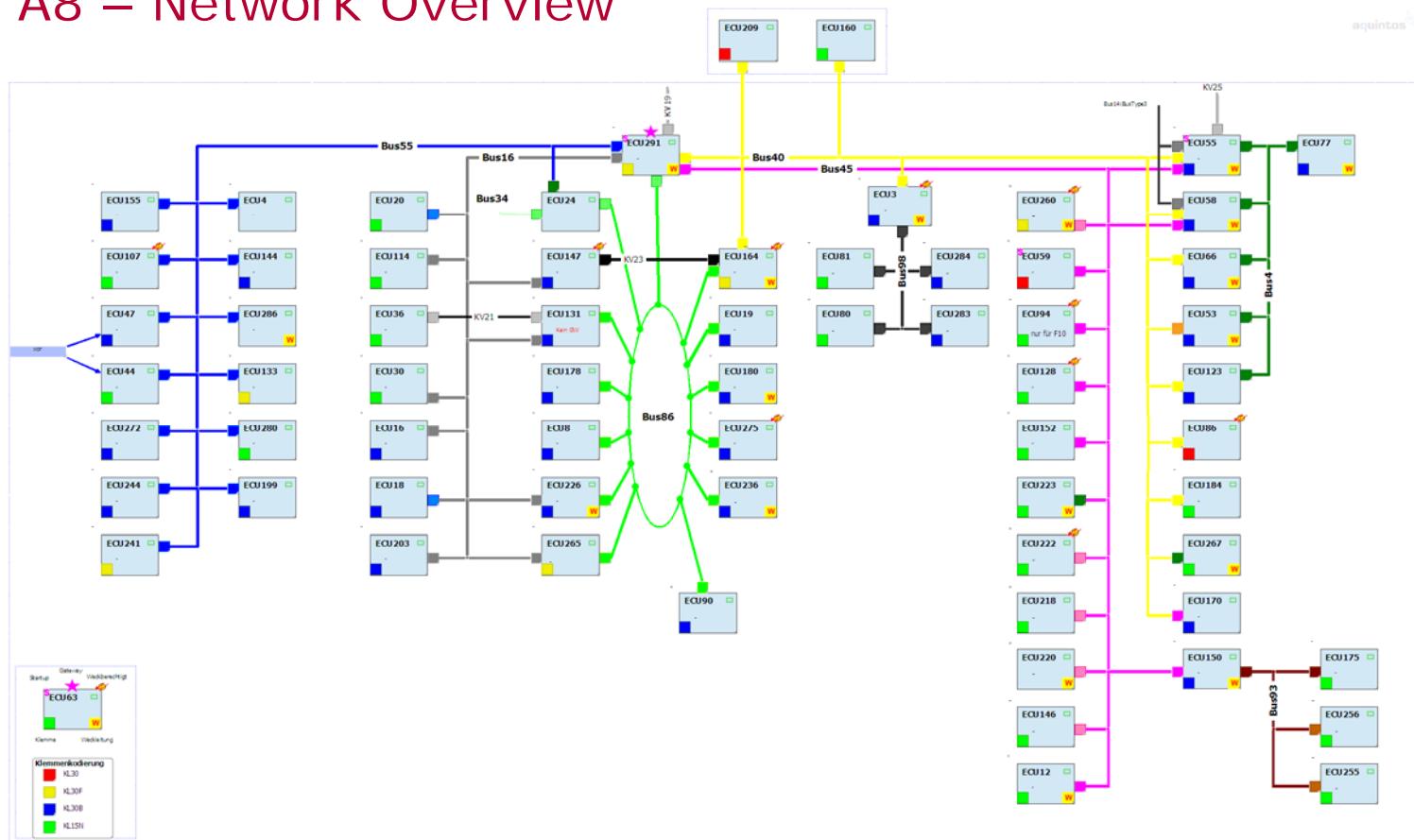
- ▶ The **Hardware Architecture** specification is also supported by graphical block diagrams.
- ▶ The mapping of logical functions can be displayed directly in the graphics, shown over the ECU block



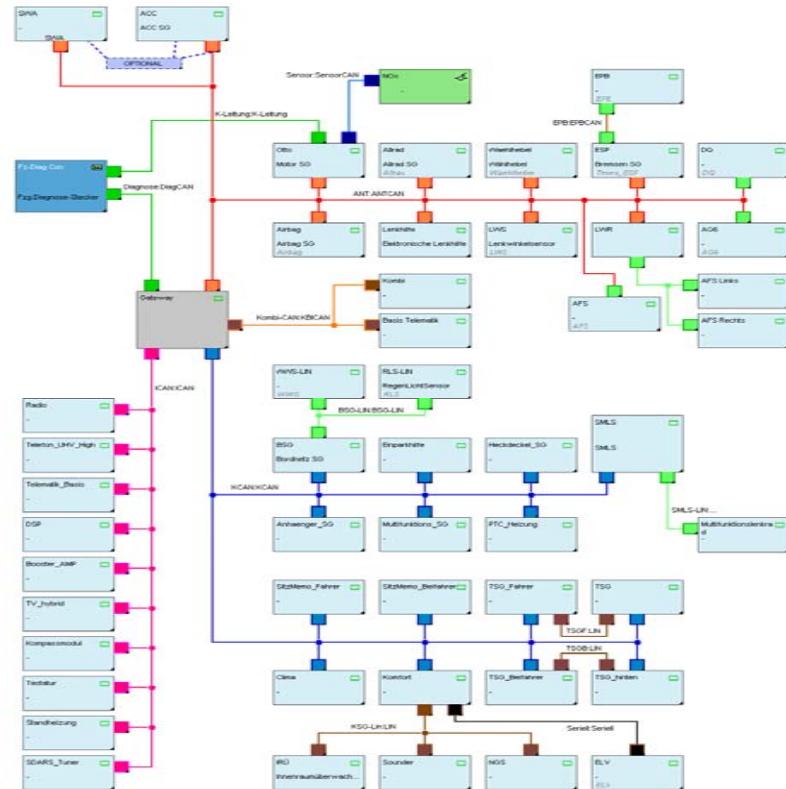
Kommunikationsarchitektur Audi A8



Kommunikationsarchitektur Audi A8 – Network Overview



Typisches Vernetzungskonzept PKW Domänen



E/E-Architekturen im KFZ werden meistens in die folgenden **Domänen** unterteilt:

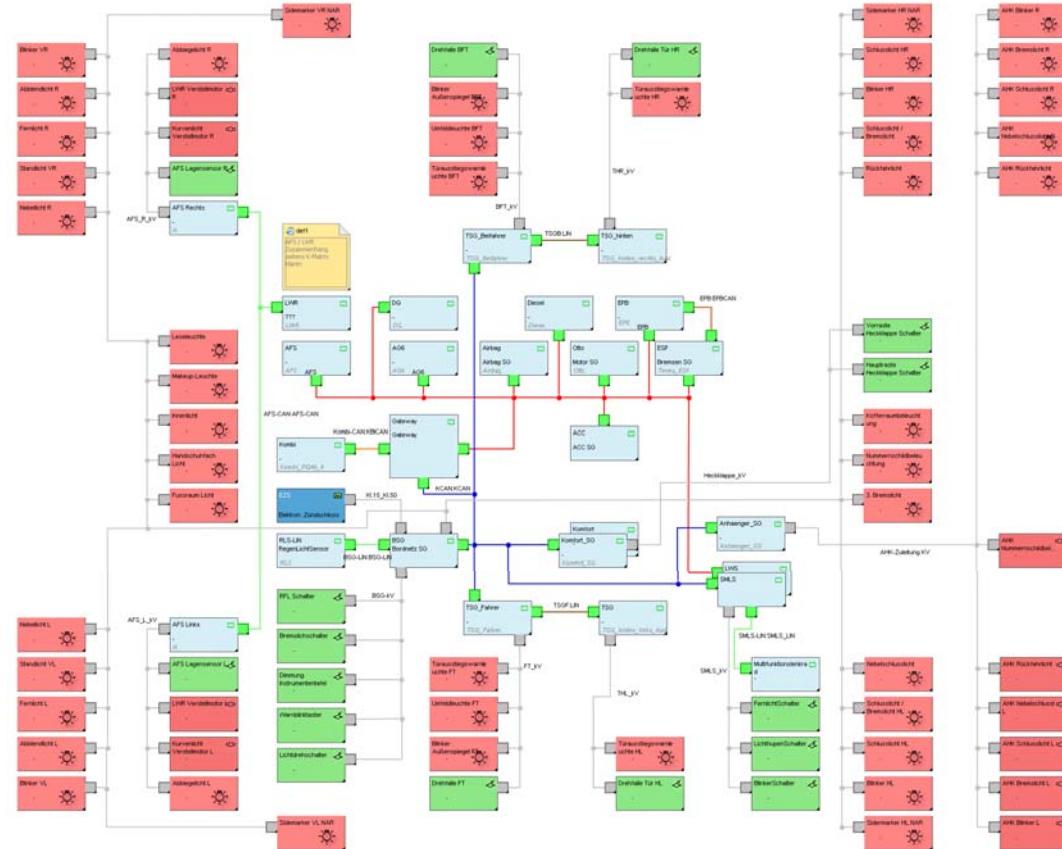
- Antriebsstrang
- Komfort (Innenraum)
- Chassis
- Telematik oder Infotainment

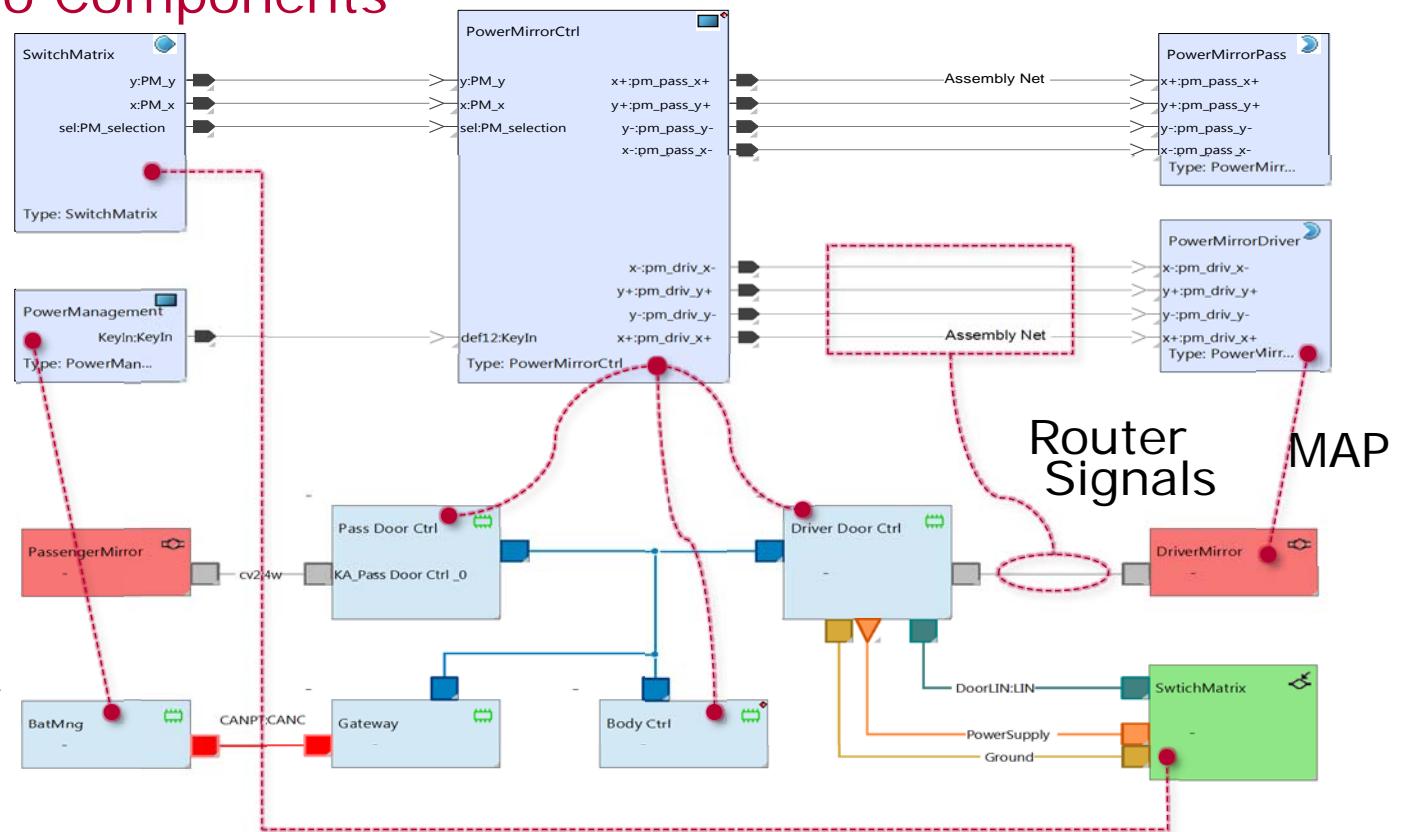
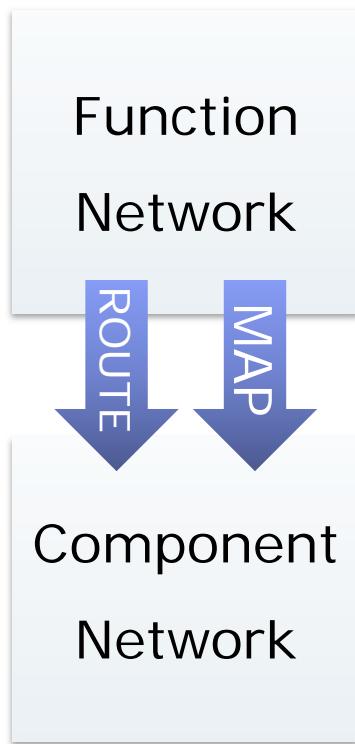
Das E/E-System eines Fahrzeugs ist **stark verteilt**. Die Kommunikation der Systeme erfolgt über standardisierte Bussysteme (CAN, LIN, MOST, FlexRay).

Der modellbasierte Architekturentwurf unterstützt den **Entwurfsprozess** stark verteilter Elektroniksysteme und erlaubt eine **Optimierung** z.B. nach Kosten, Gewicht, Bauraumbedarf etc.

Typisches Vernetzungskonzept PKW

Aufbau eines Lichtsystems



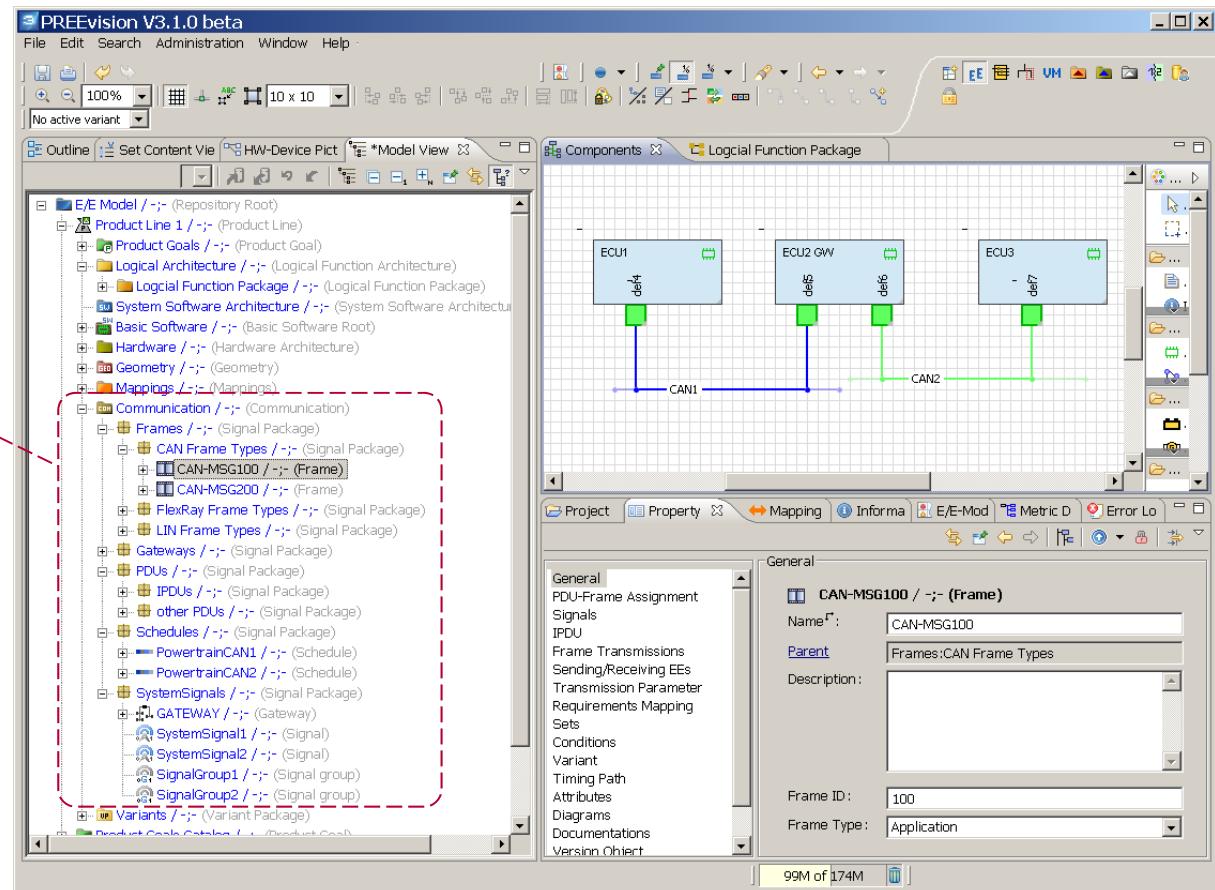


Communication Layer in PREEvision

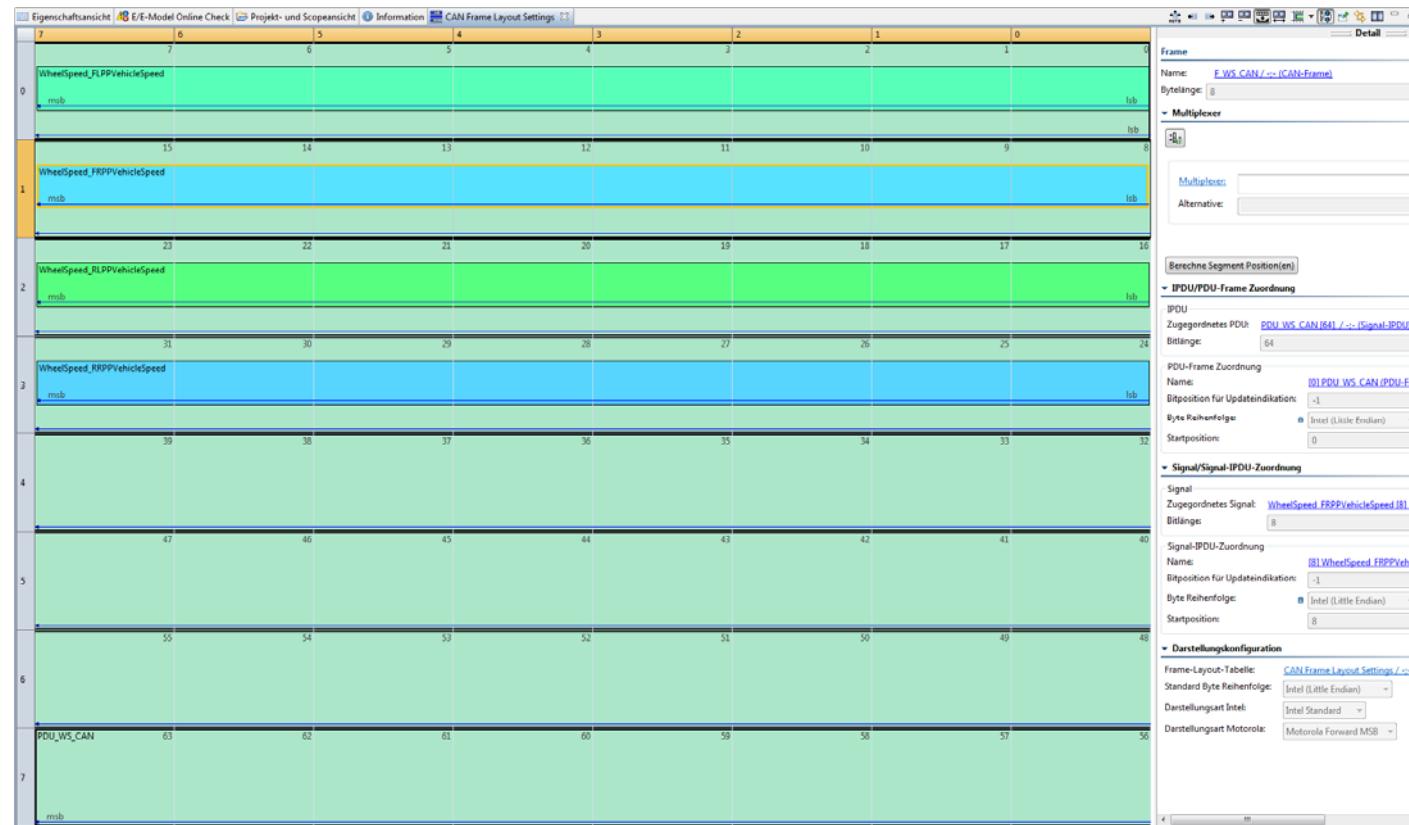


Communication Layer to specify

- ▶ Signals
- ▶ PDUs
- ▶ Frames
- ▶ Schedules

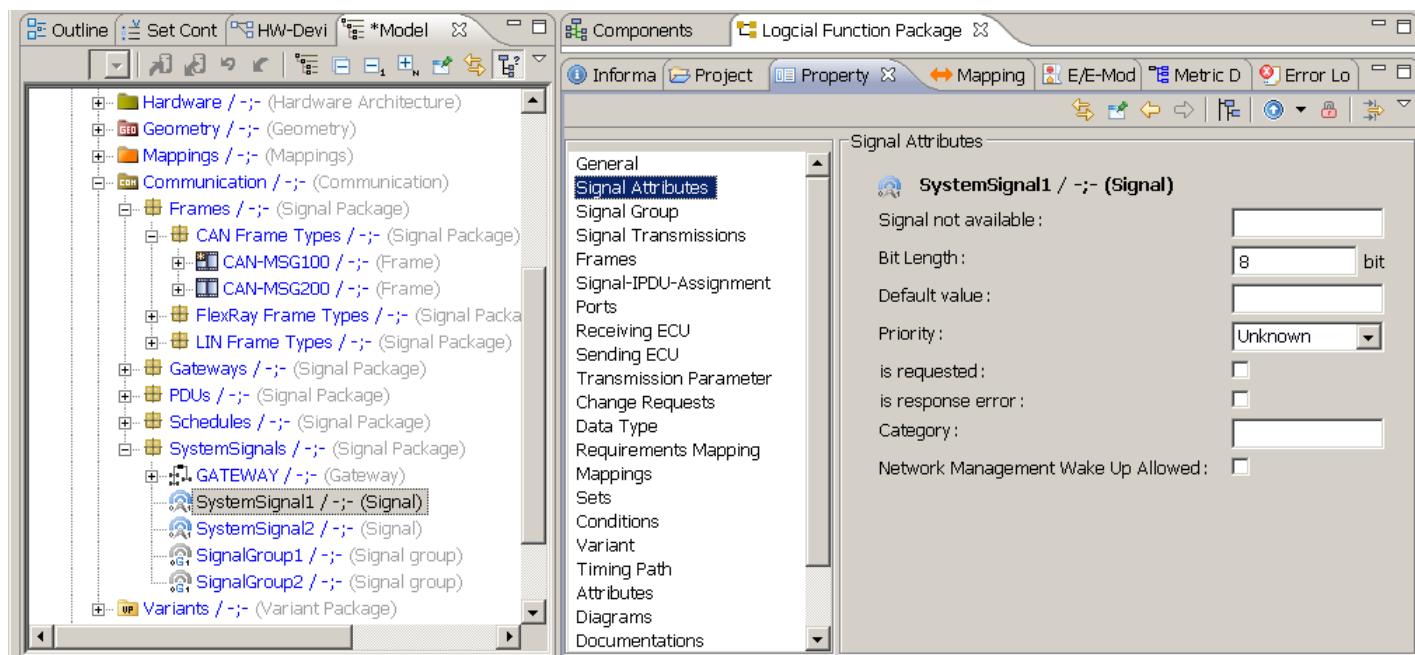


Communication Layer in PREEvision Frame Layout Design



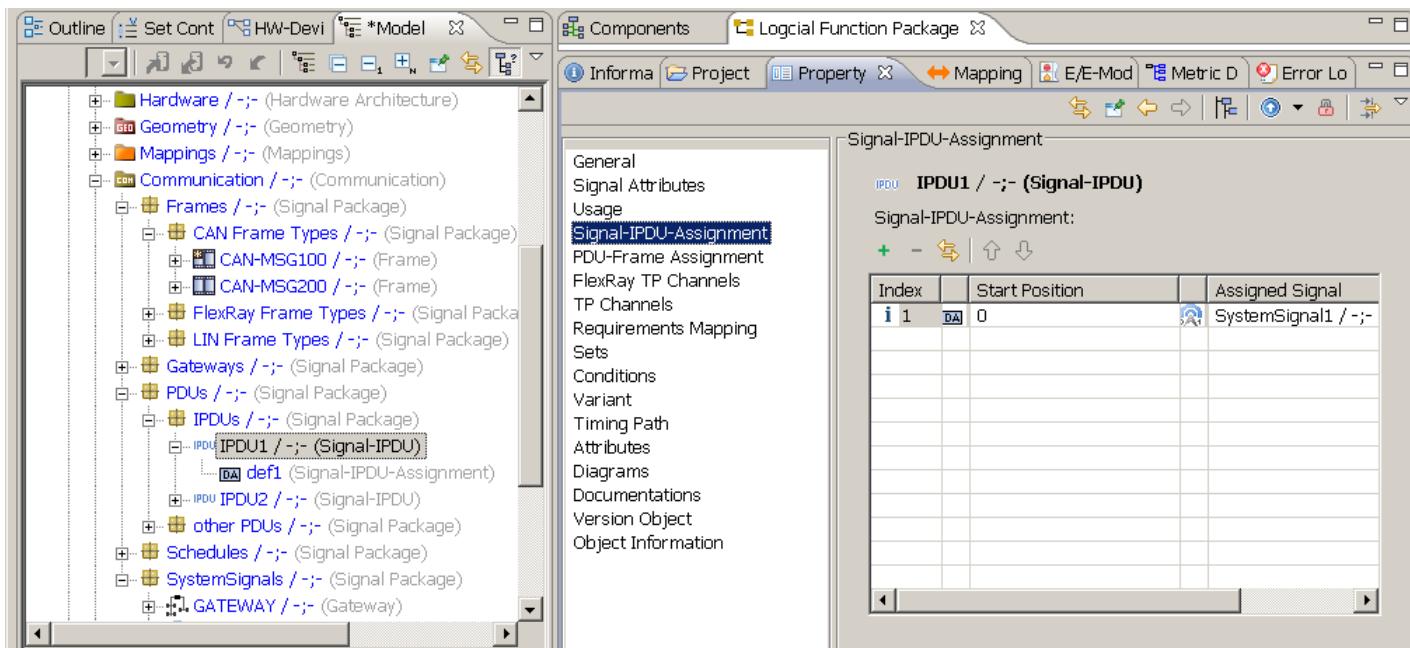
Communication Layer in PREEvision System Signal Specification

The specification of **System Signals** can be done by the Property Editor..



Communication Layer in PREEvision PDU Specification

PDUs are specified interactively based on *System Signals*



The main window displays the "Signal-IPDU-Assignment" tab for the "IPDU1 / -;-(Signal-IPDU)" object. A table lists the assigned signals:

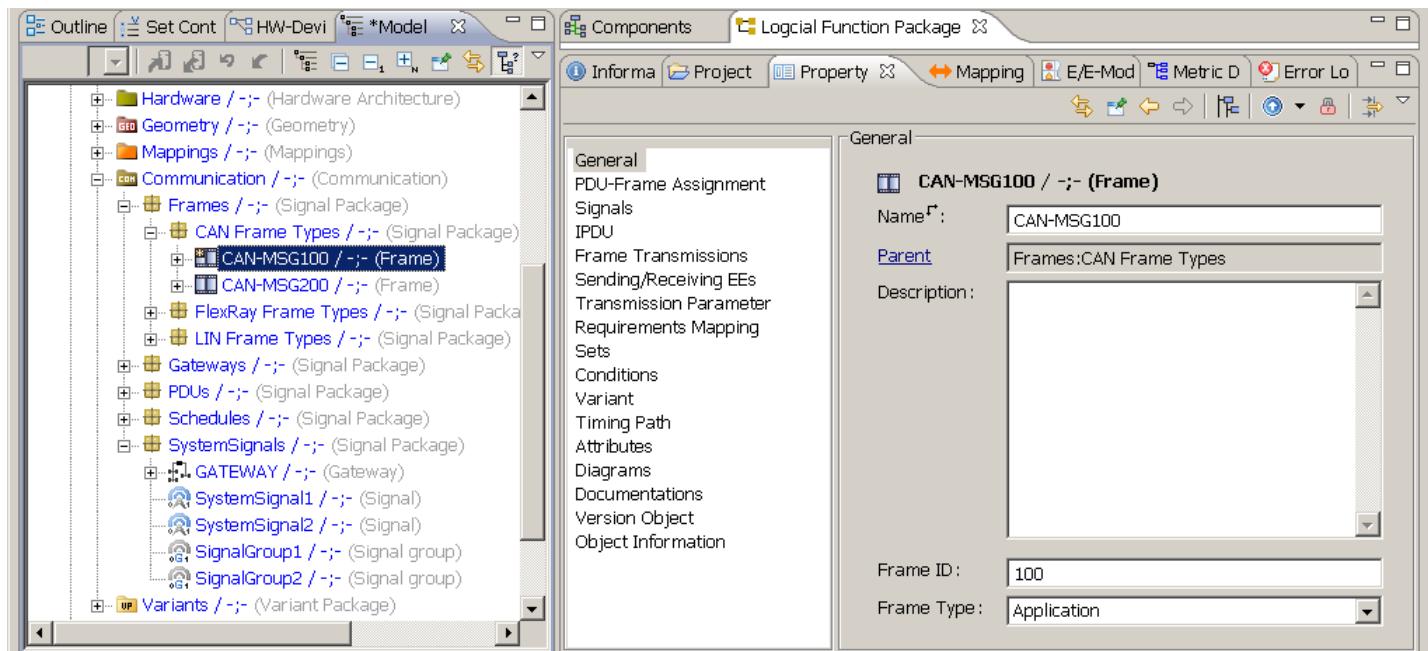
| Index | Start Position | Assigned Signal |
|-------|----------------|---------------------|
| i 1 | DA 0 | SystemSignal1 / -;- |

PDU: Protocol-Data-Unit

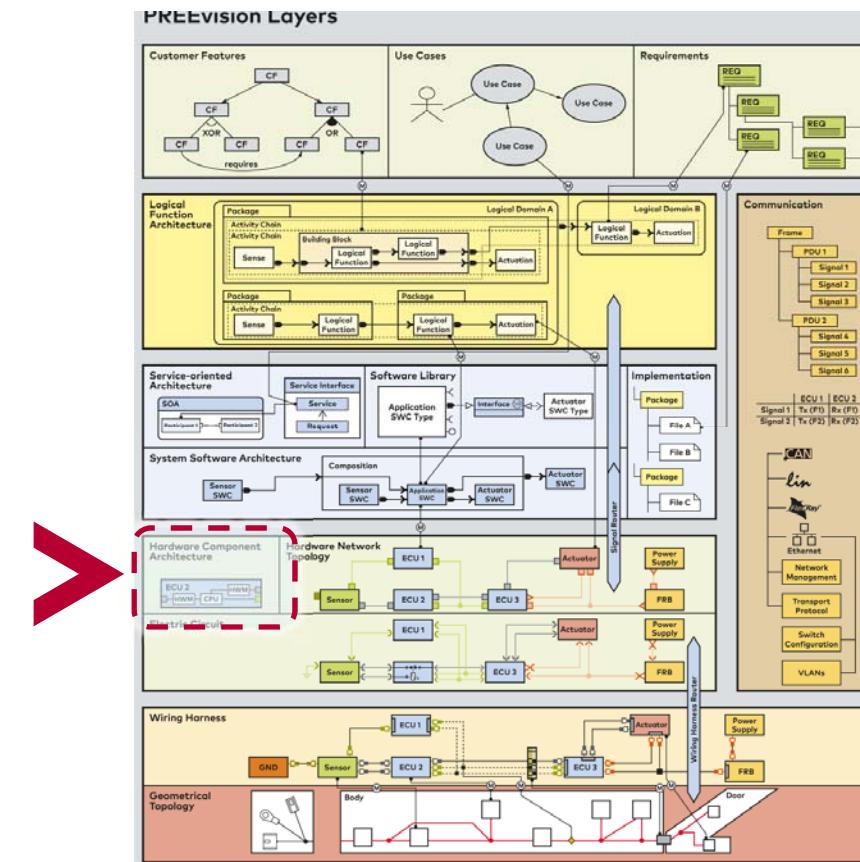
Communication Layer in PREEvision Frame Specification

Frames are specified interactively based on PDU^s and are available for

- ▶ CAN
- ▶ LIN
- ▶ FlexRay



PREEvision Collaboration Platform (IV) Bauteil/Steuergerät



Aufbau eines Steuergerätes (Komponenteneditor)

Aufbau einer ECU

- ▶ CPU, FPGA, RAM, etc.
- ▶ Gateway-Struktur

Kommunikation zwischen
Busanbindung und CPU

ECU: Electronic Control Unit

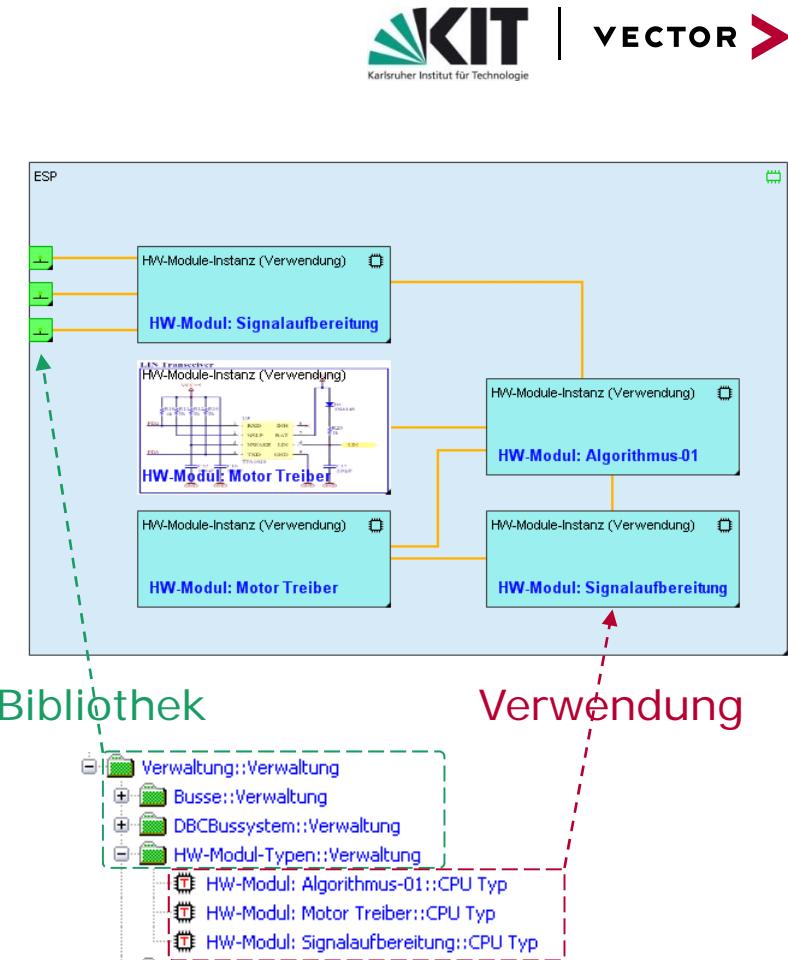


HW-Modulmodellierung Aus Zulieferer-Sicht

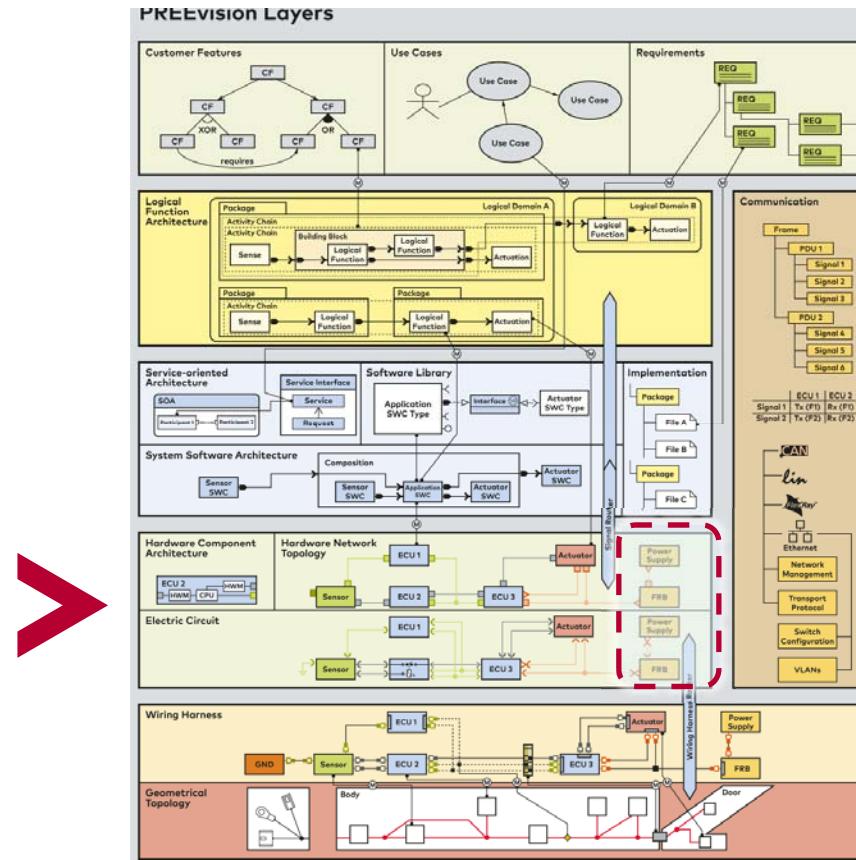
(a) zur Verwendung | (b) als Bibliothek

Attribute eines HW Moduls

- ▶ Stückliste
 - ▶ Bauteilbezeichner
 - ▶ Teilenummer
 - ▶ Gehäuse
 - ▶ Fläche
 - ▶ Anzahl
 - ▶ Einzelkosten (aus DB) Import
 - ▶ Bild
- ▶ Kumulierte Attribute
 - ▶ Kosten
 - ▶ Fläche (aus Bauform,
Overhead Anteil, Offset)
 - ▶ Gewicht

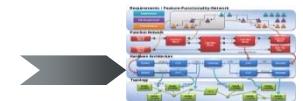


PREEvision Collaboration Platform (IV) Leistungsversorgungs-Architektur



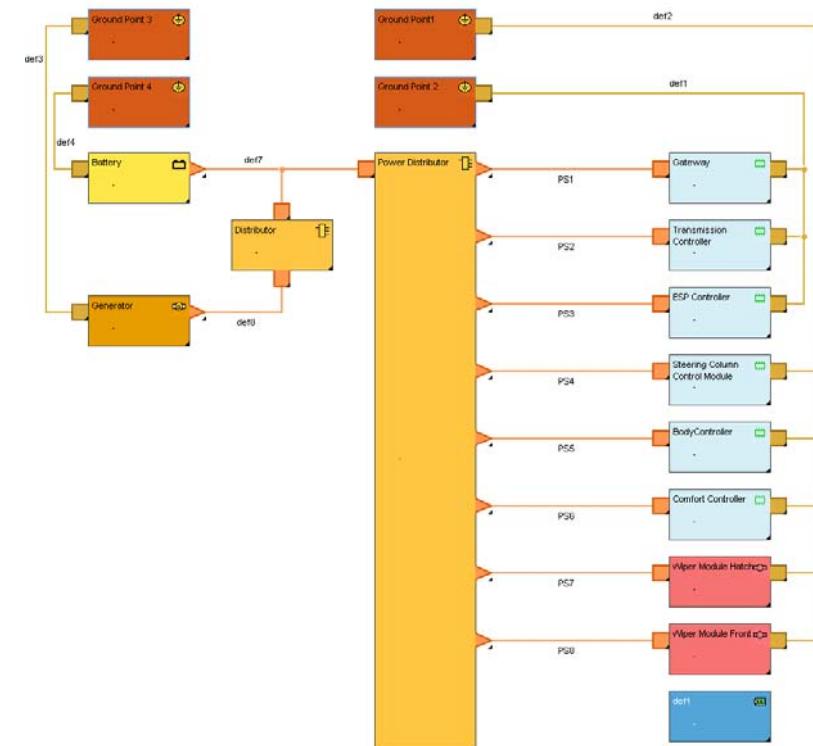
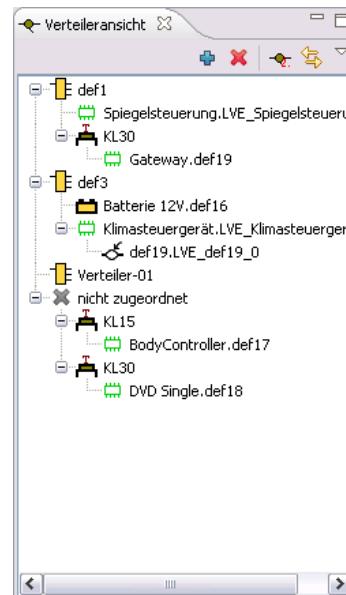
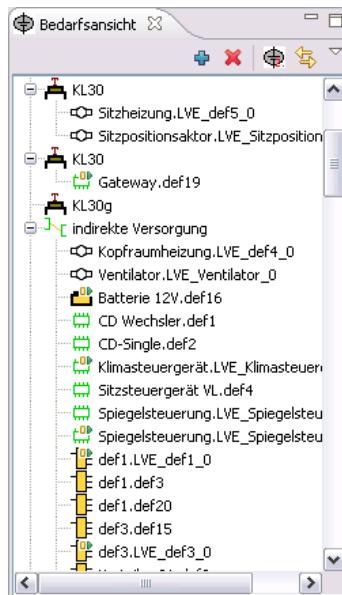
Stromversorgung

Stromverteilungsmodellierung



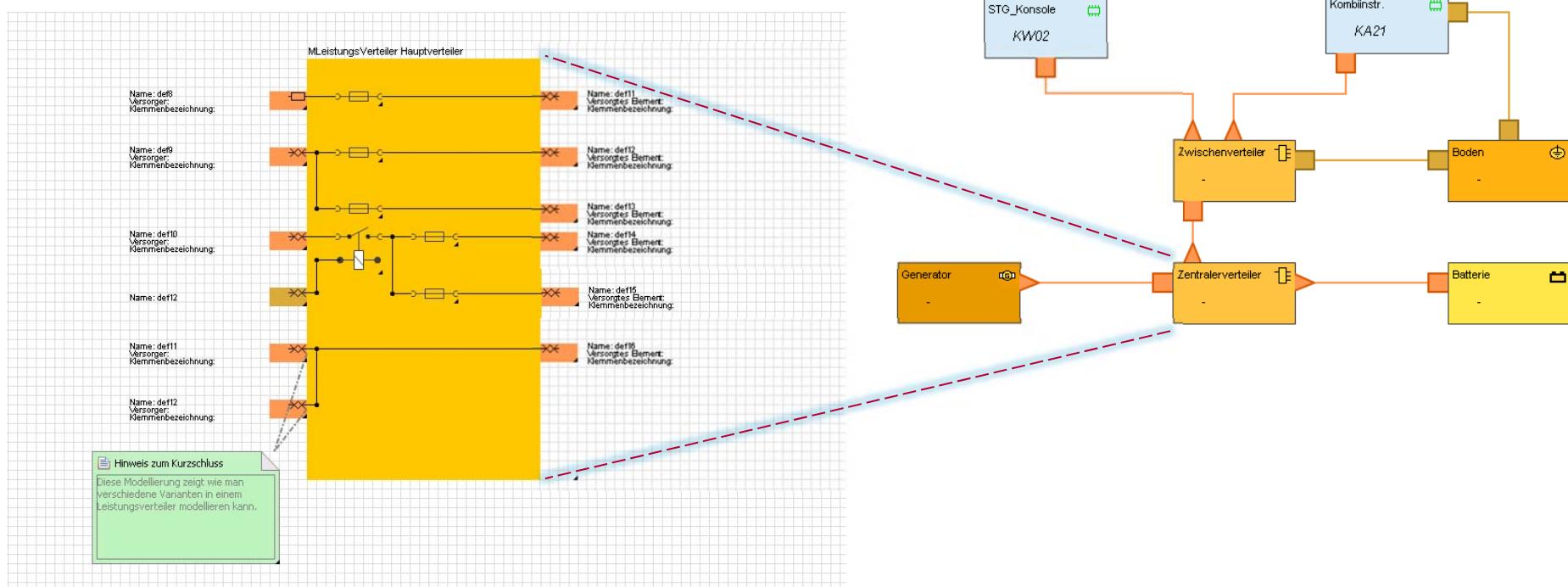
Schnelle Modellerzeugung durch zwei Ansichten:

- ▶ Bedarfsansicht (Klemmenbedarf)
- ▶ Verteilungsansicht



Sicherungs-Relais-Box

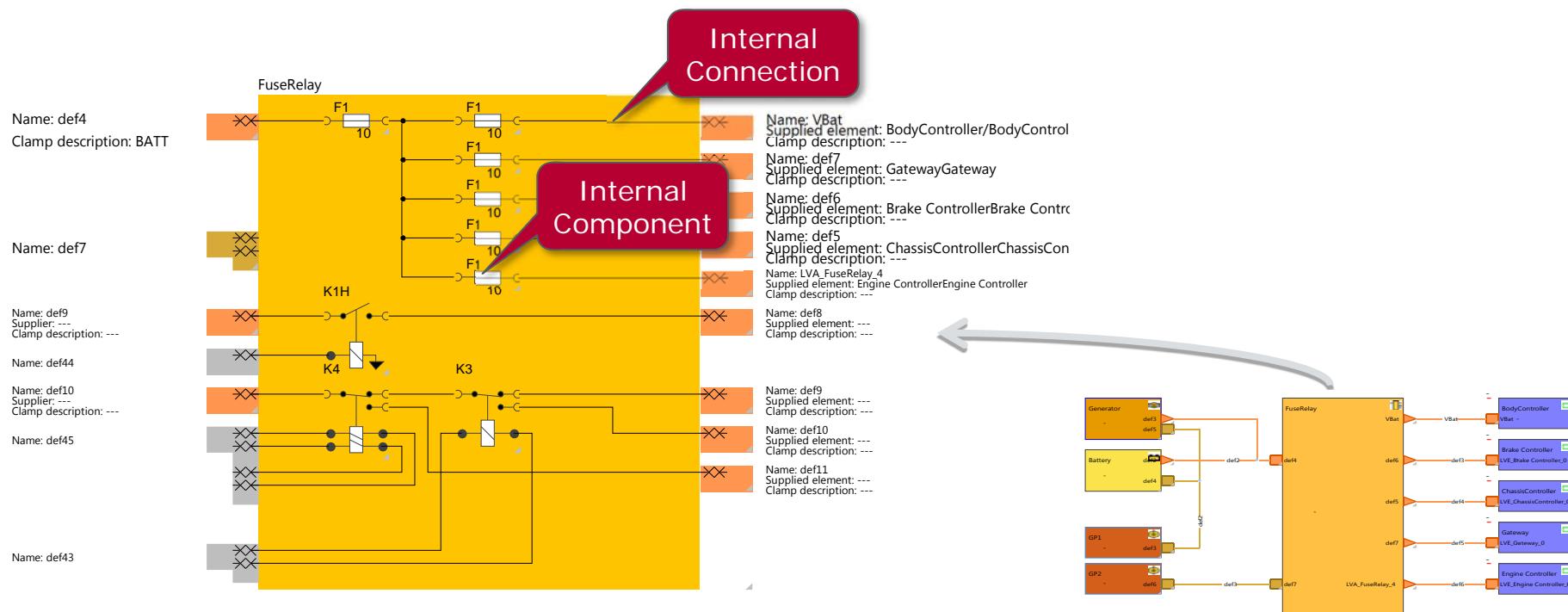
Spezifikation von Sicherungen und Relais



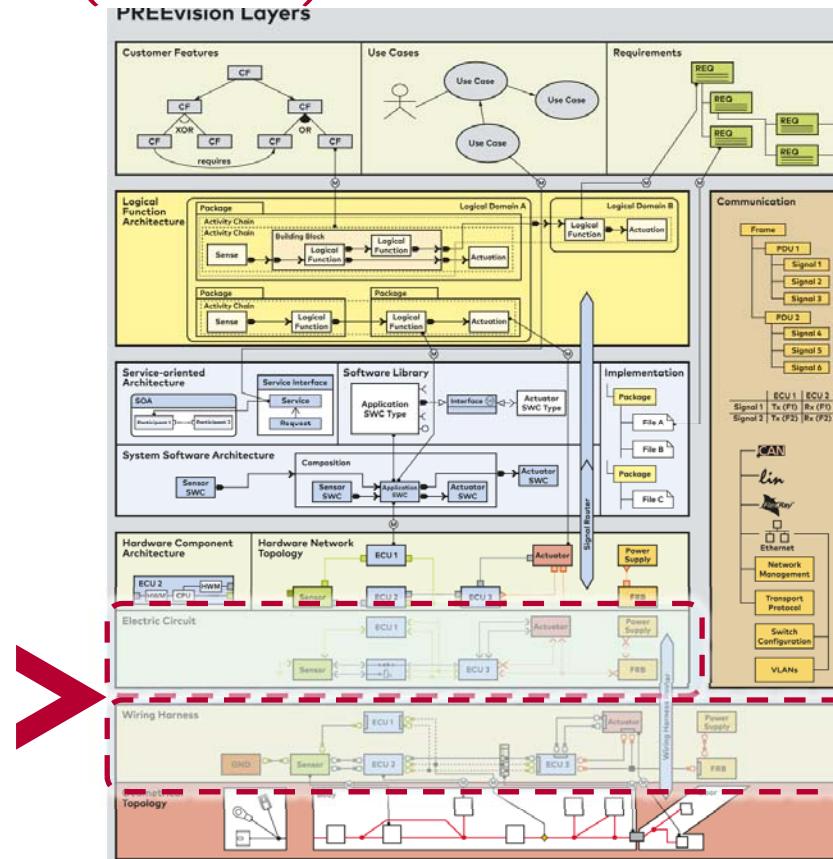
Internal Schematic Diagram

Internal Schematic Diagram is auto-layouthed

Quick Conceptualization of e.g. Fuse-Relay Systems



PREEvision Collaboration Platform (V) Stromlaufplan Leitungssatz (Elektrik)

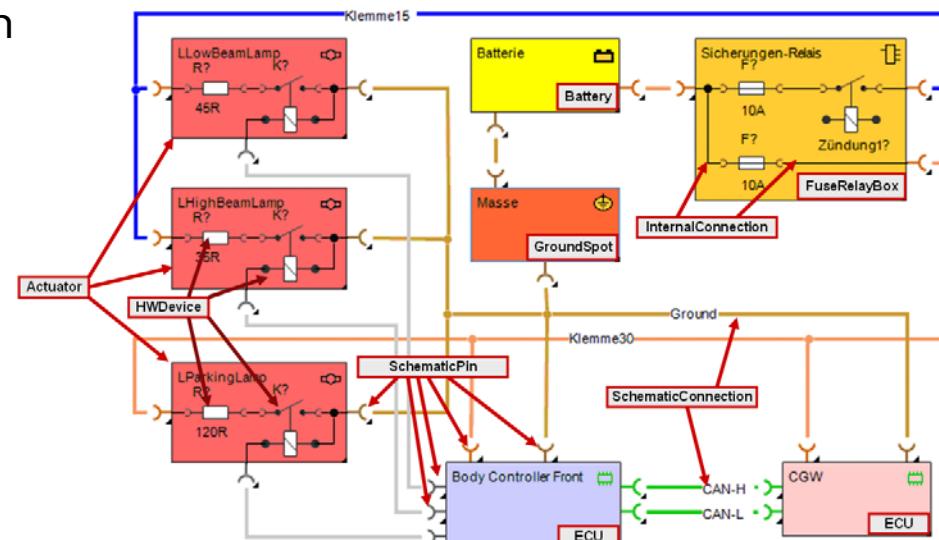


Darstellung der **elektrischen Verbindungen** zwischen Komponenten

- ▶ Keine Trennstellen
- ▶ Keine Splices
- ▶ Keine Stecker
- ▶ Vereinfachte Darstellung von Pins

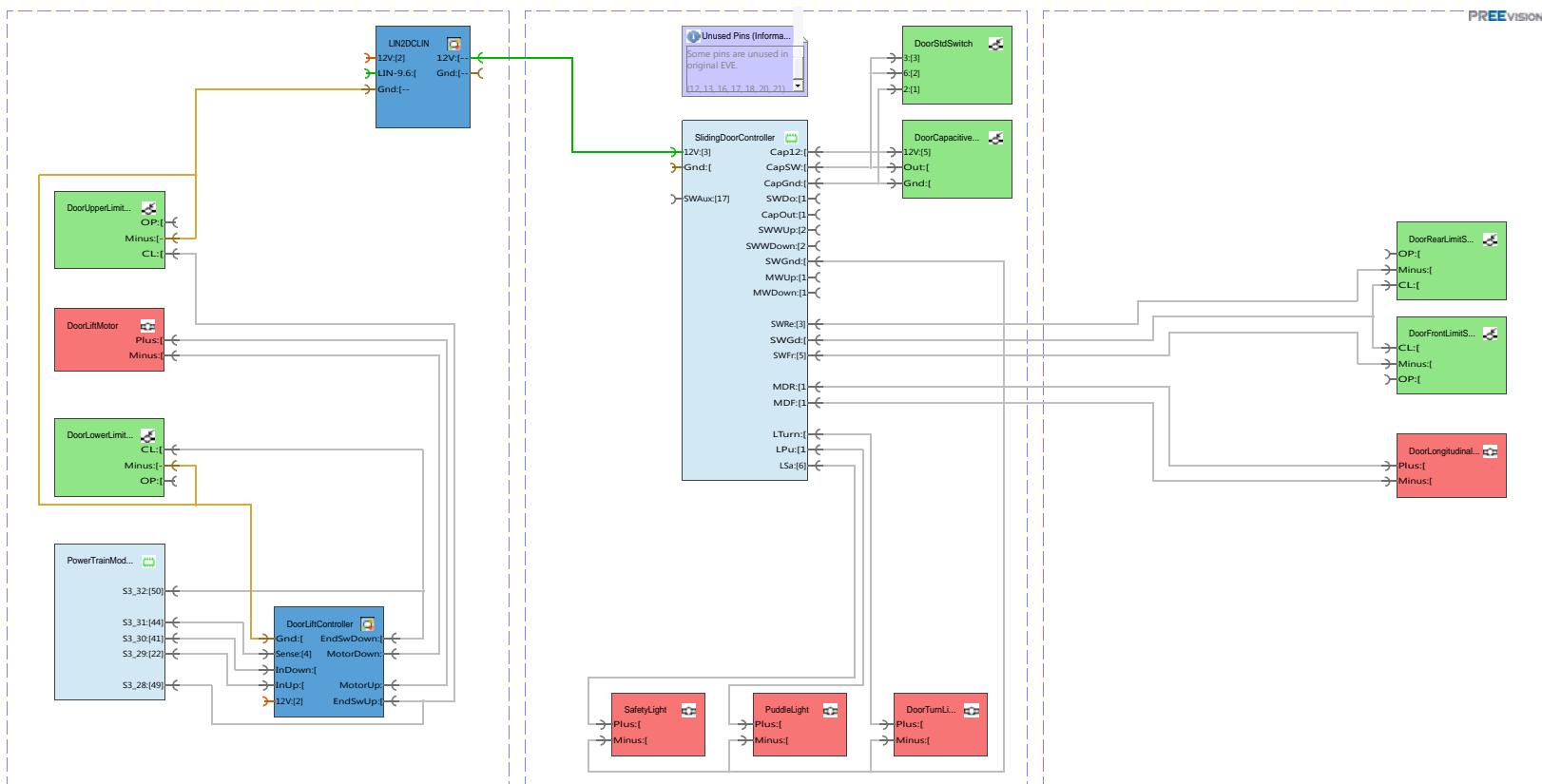
Komponenten können mit „Ersatzschaltbild“ hinterlegt werden

- ✓ Leichteres Verständnis der elektrischen Zusammenhänge
- ✓ Automatisierte Generierung von Werkstattmaterial



Stromlaufplan

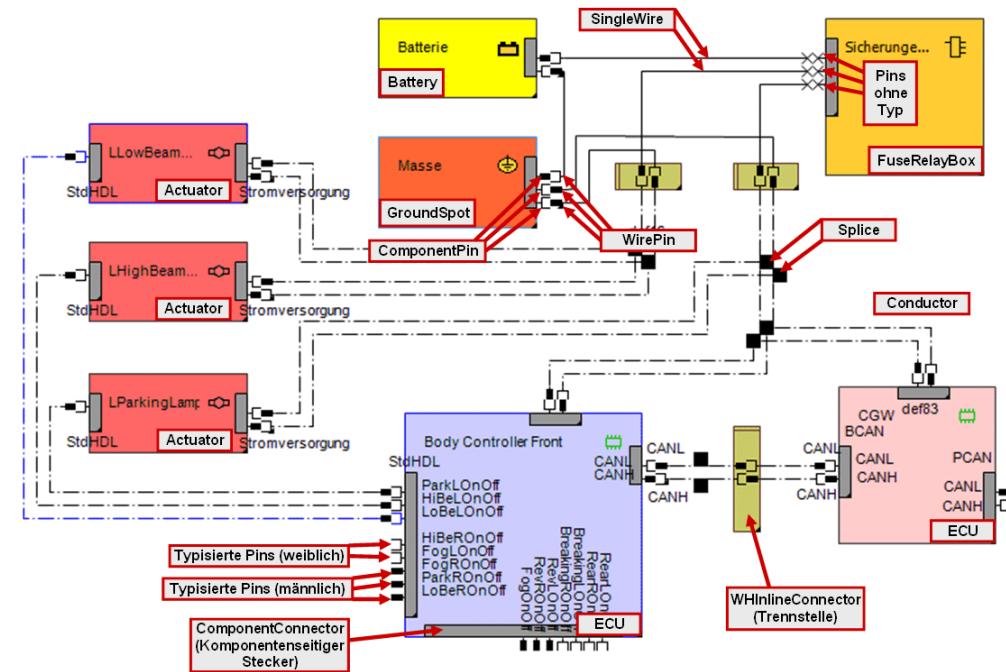
Beispiel Stromlaufplan



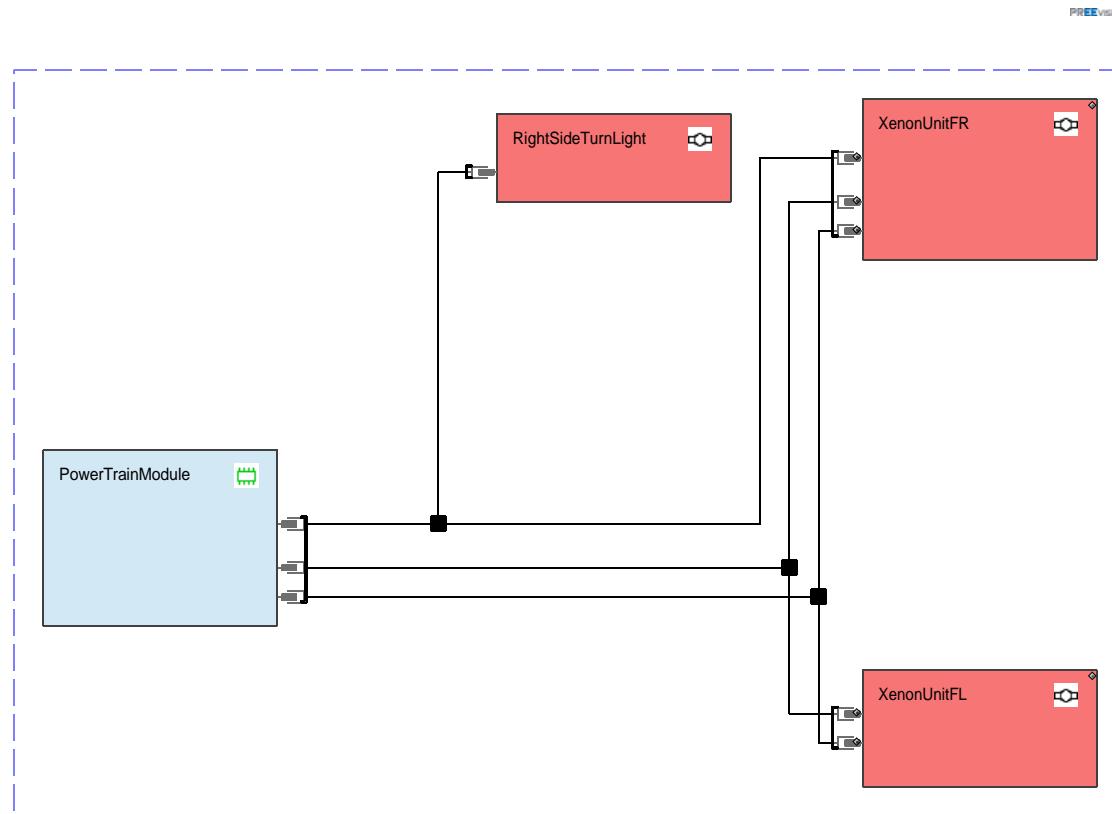
Darstellung aller Leitungssatzelemente

- ▶ Trennstellen
- ▶ Splices
- ▶ Detaillierte Pins
- ▶ Stecker-Partitionierung

- ✓ Dokumentation des Leitungssatzes



Leitungssatz Beispiel Leitungssatzdiagramm

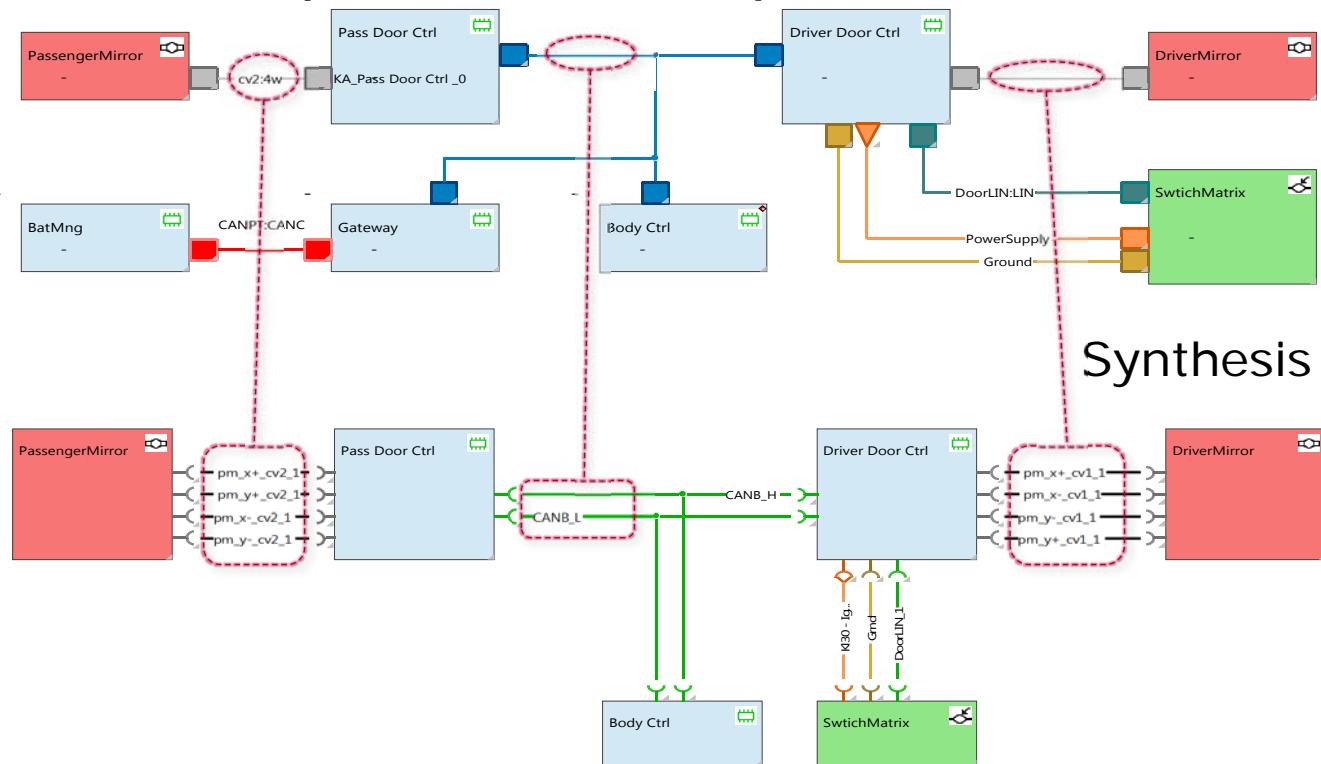


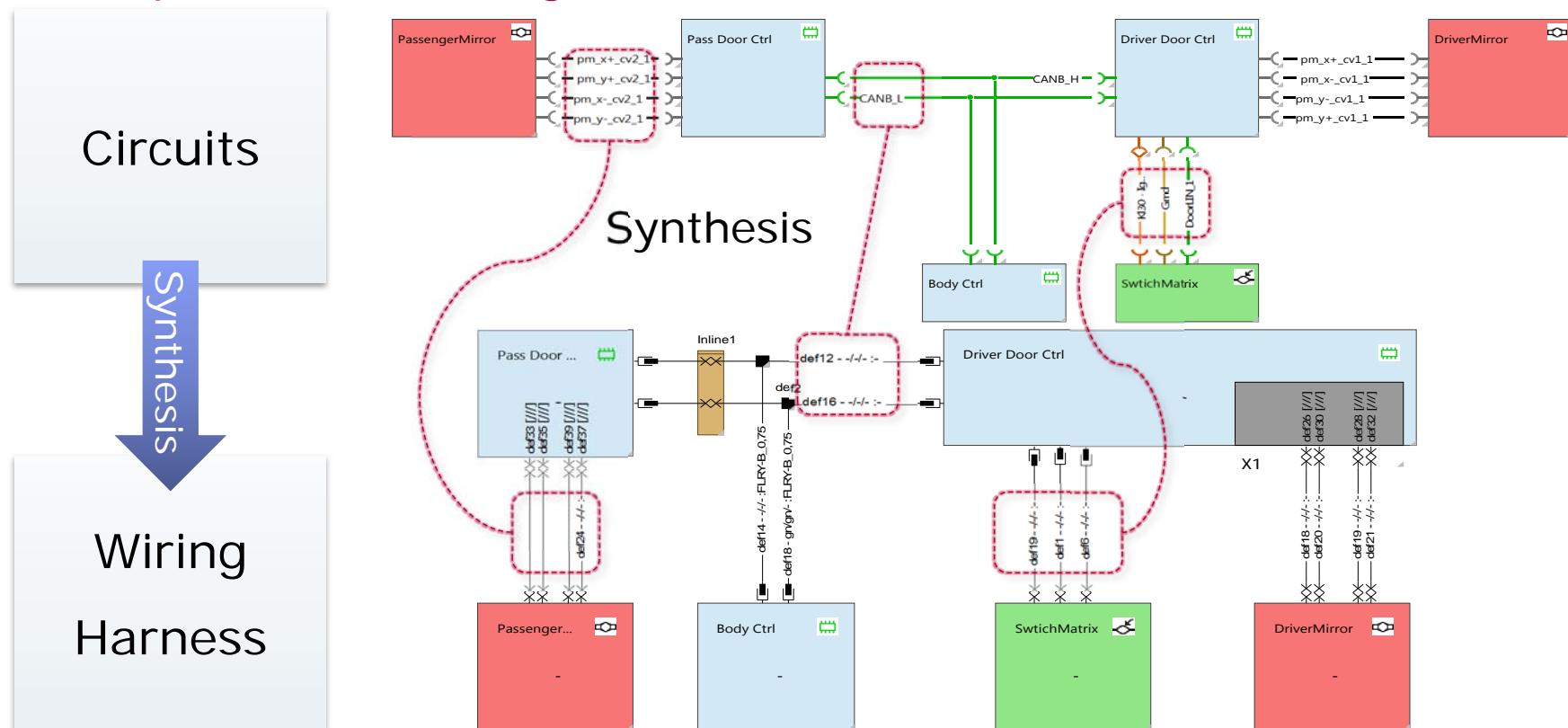
PREEvision Layers Components & Circuits

Component
Network

Synthesis
↓

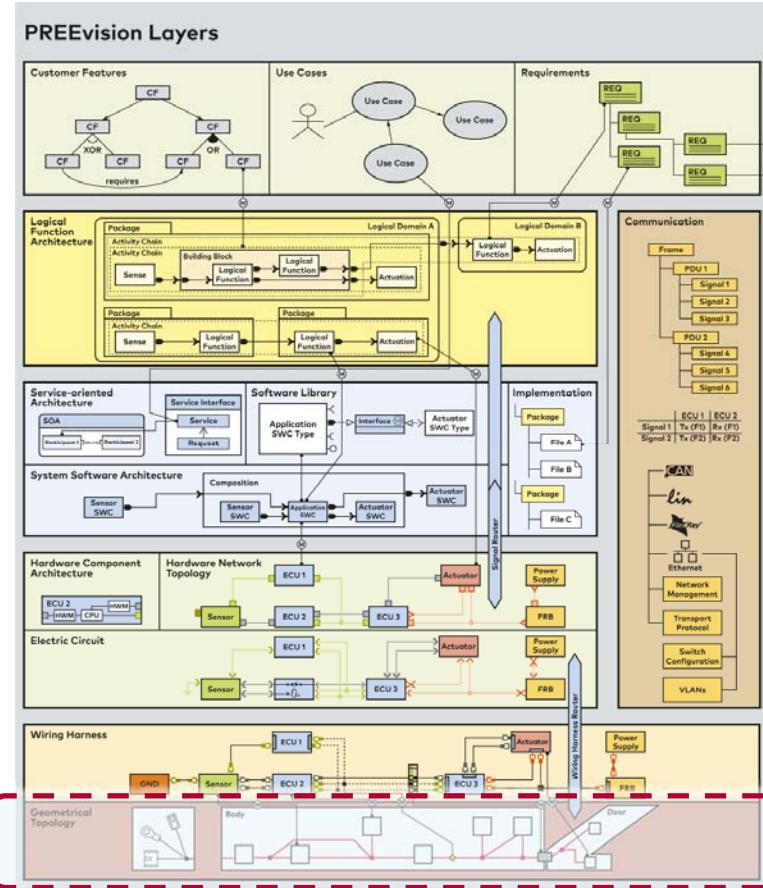
Circuits



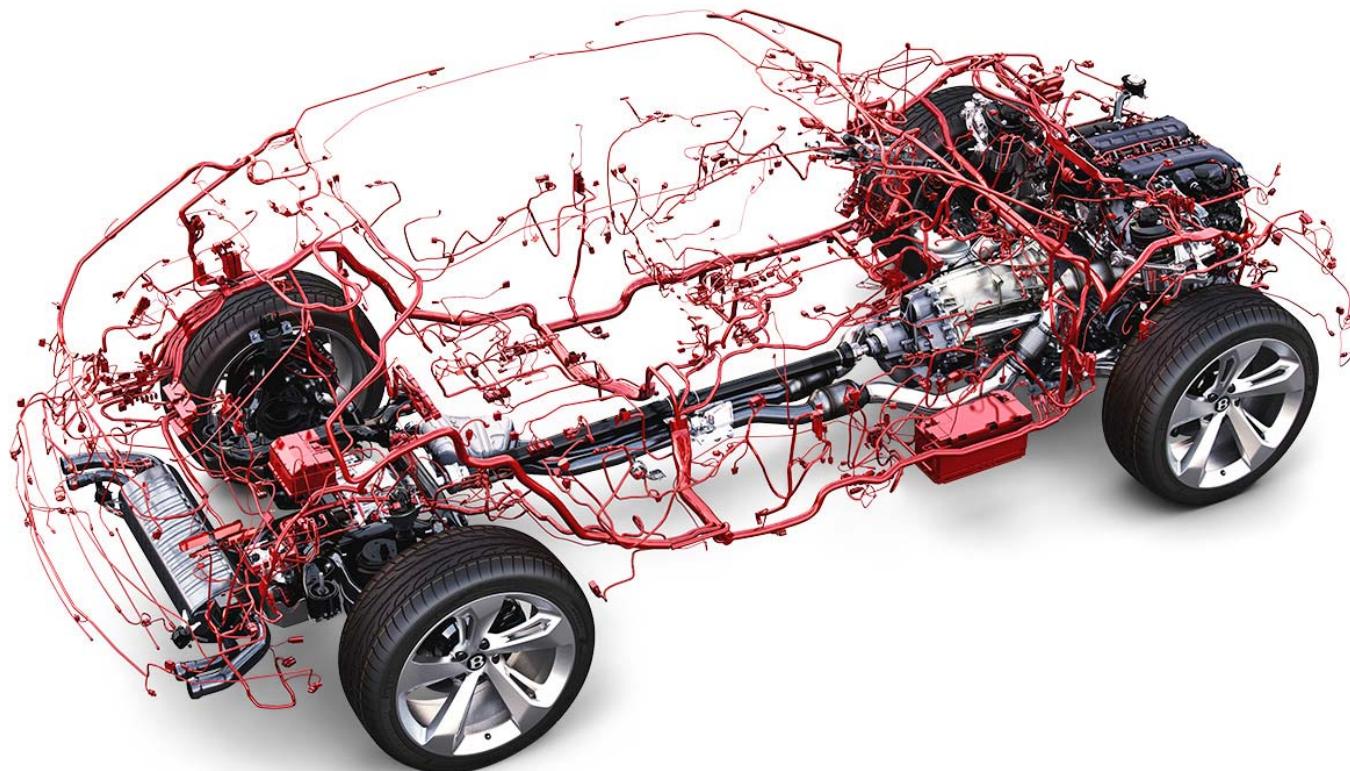


PREEvision Collaboration Platform (VI)

Geometrie

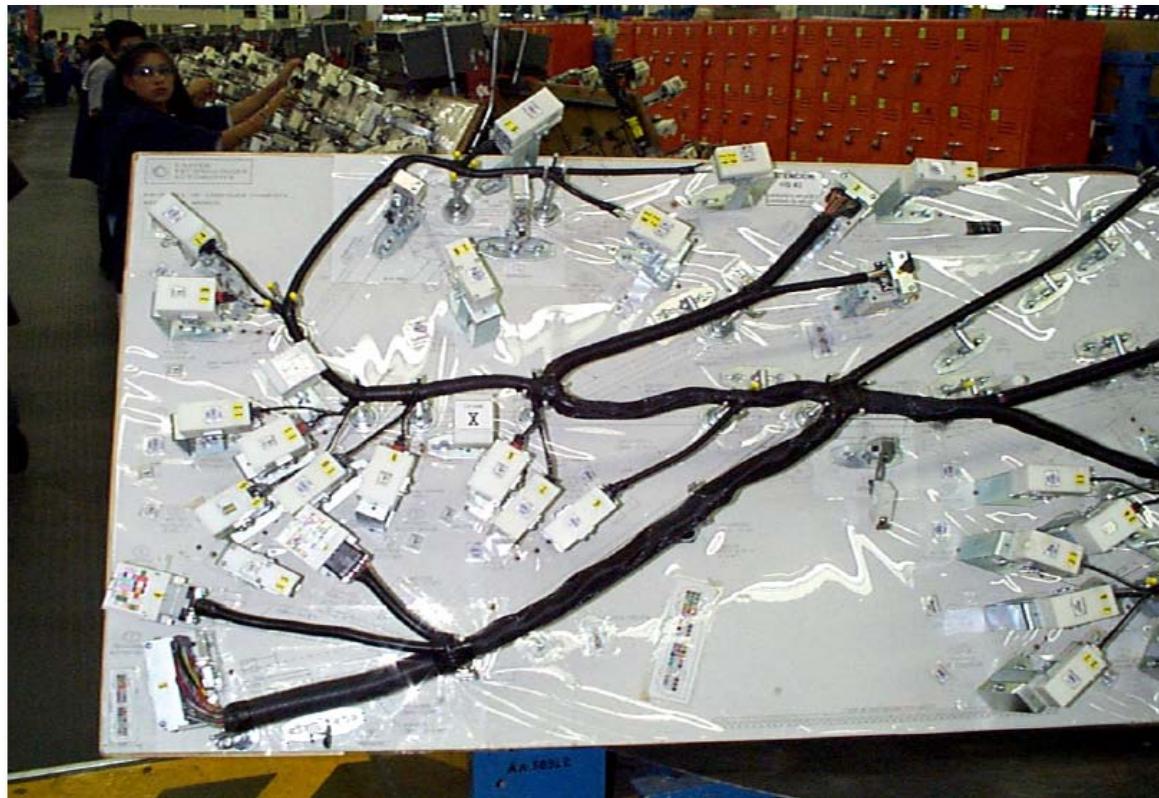


Geometrie Zentralleitungssatz eines PKWs

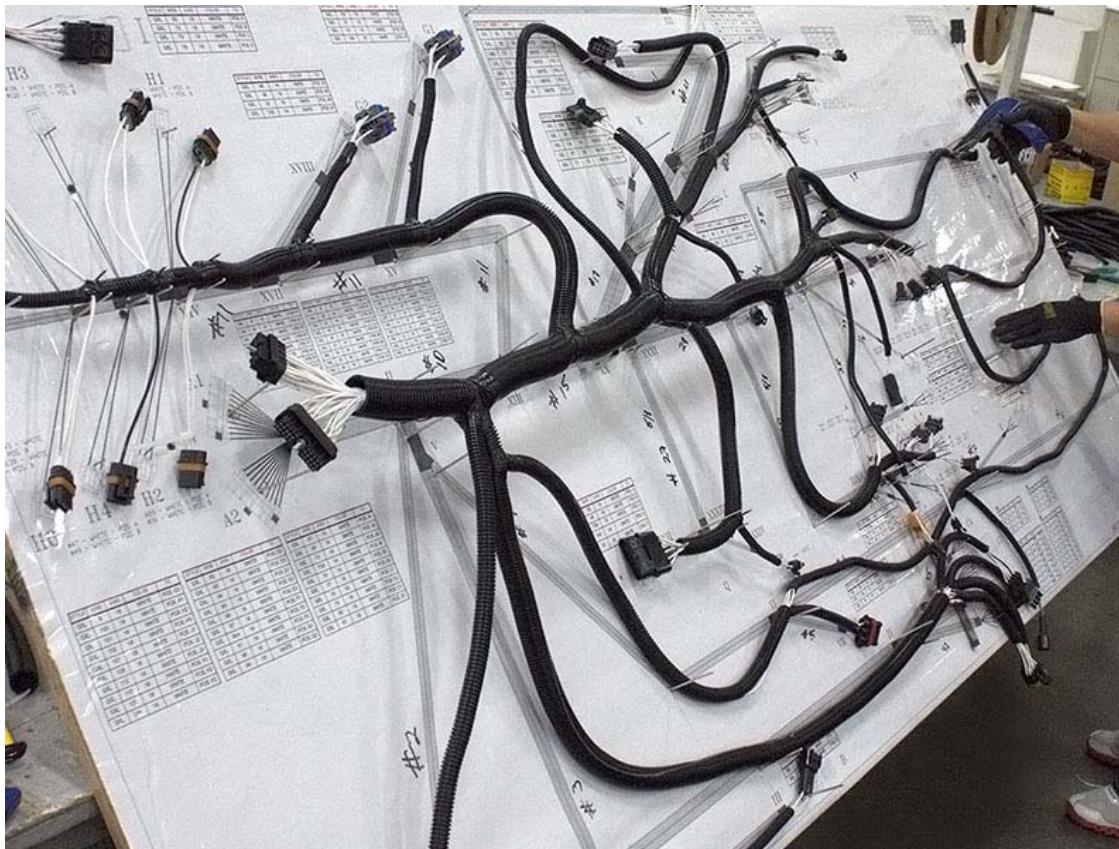


Ca. 2500 verlegte Leitungen mit Querschnitten zwischen 1qmm und 16qmm Kupfer

Beispiel Leitungssatz mit Steuergeräten

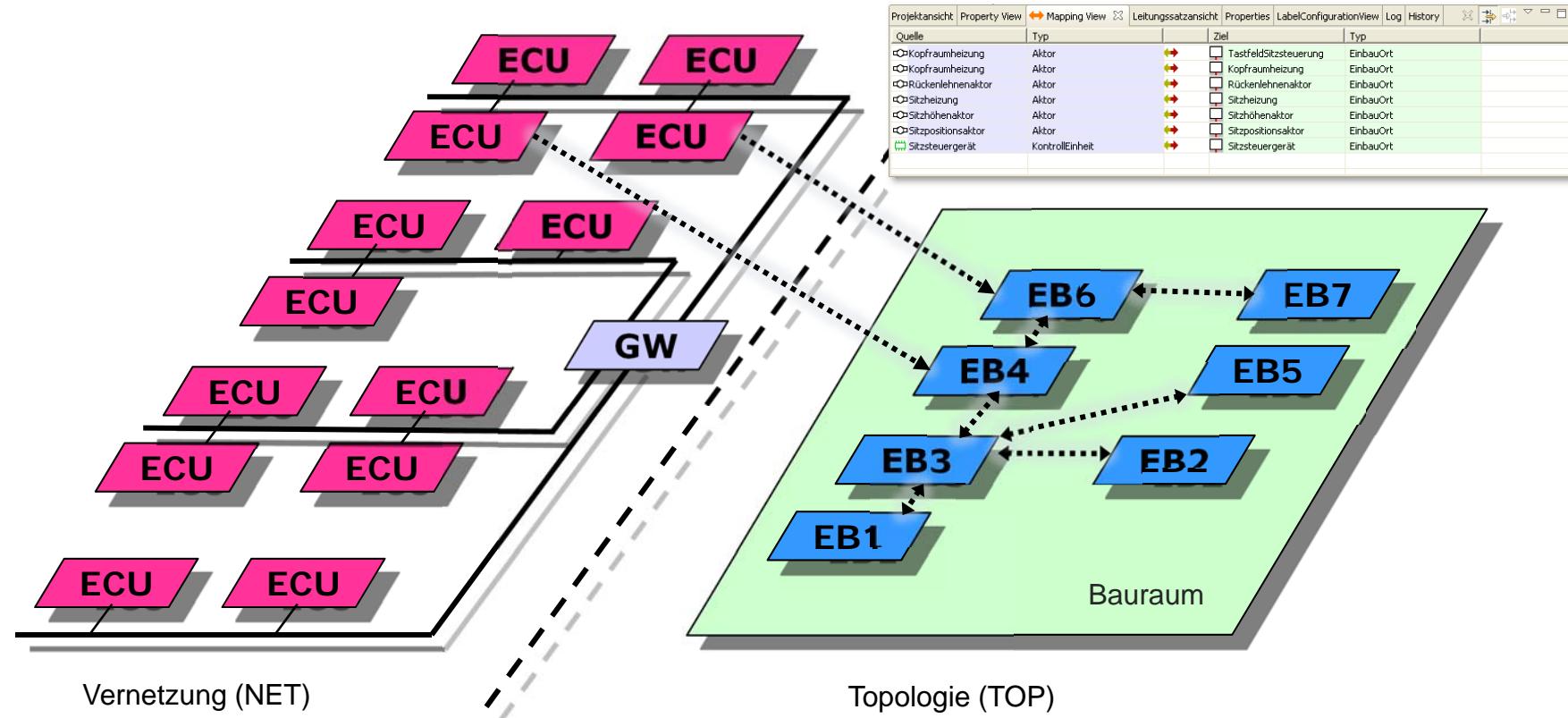


Geometrie Beispiel Leitungssatz

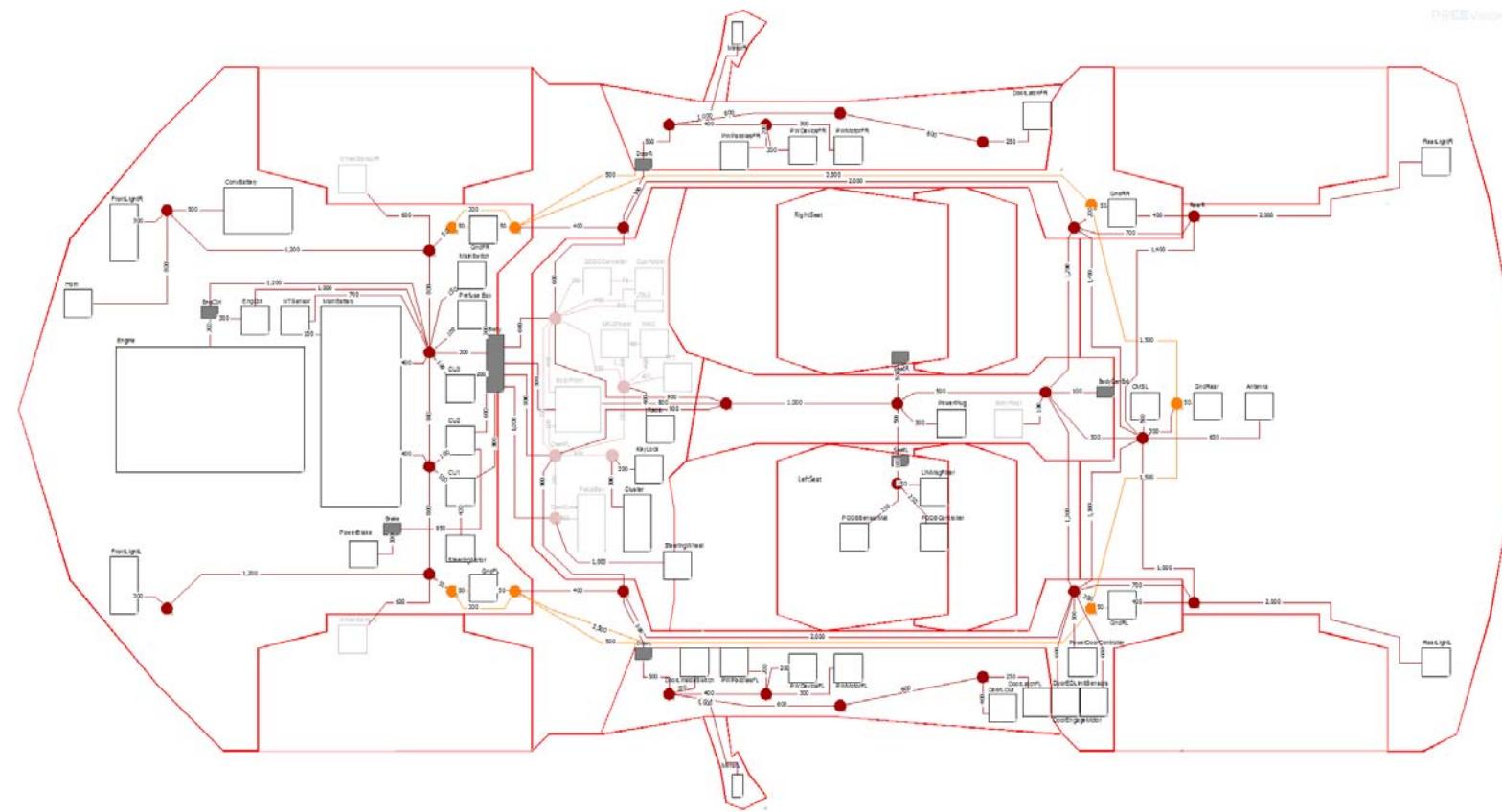


Verknüpfung NET → GEO durch Mappings

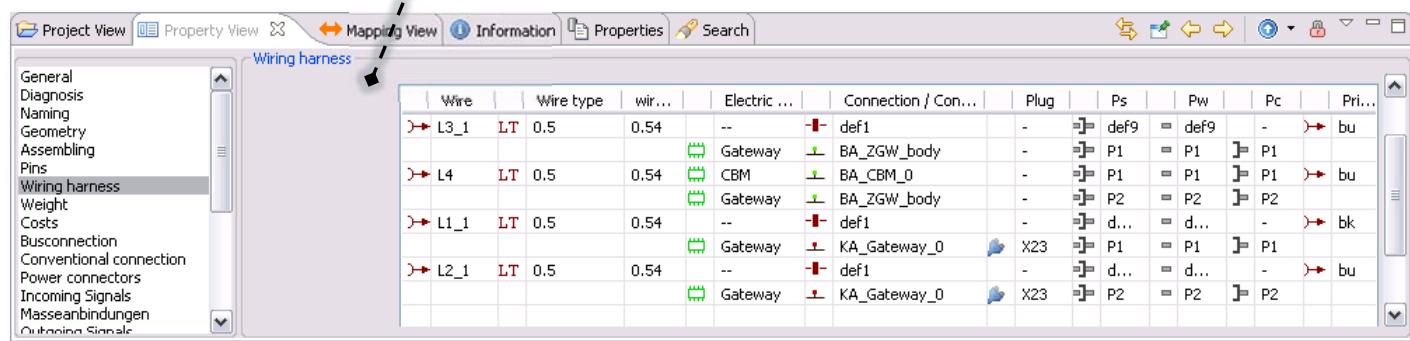
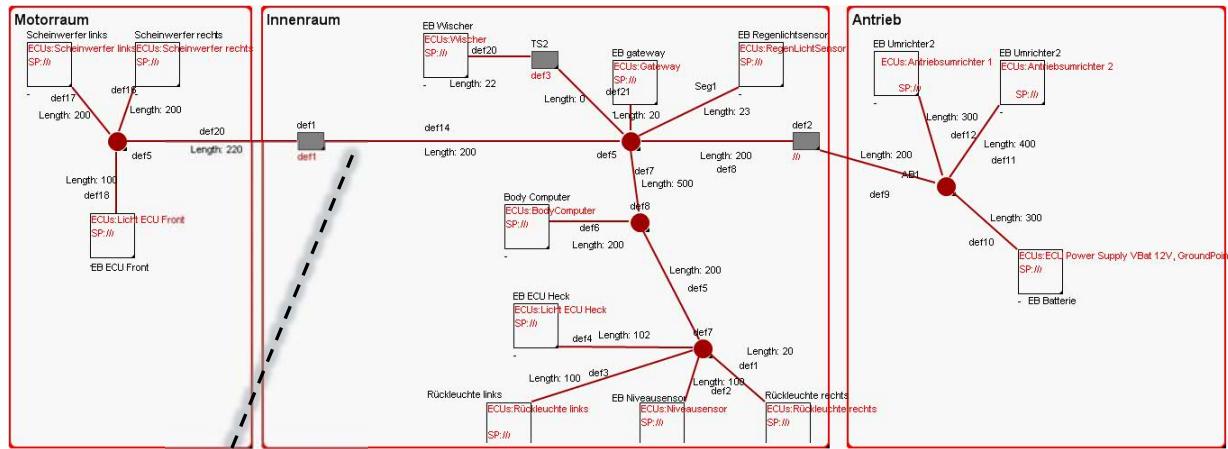
Mapping der Steuergeräte auf Topologie-Ebene



Topologie des Fahrzeugs



Topologie Sitzmodul Leitungssatz

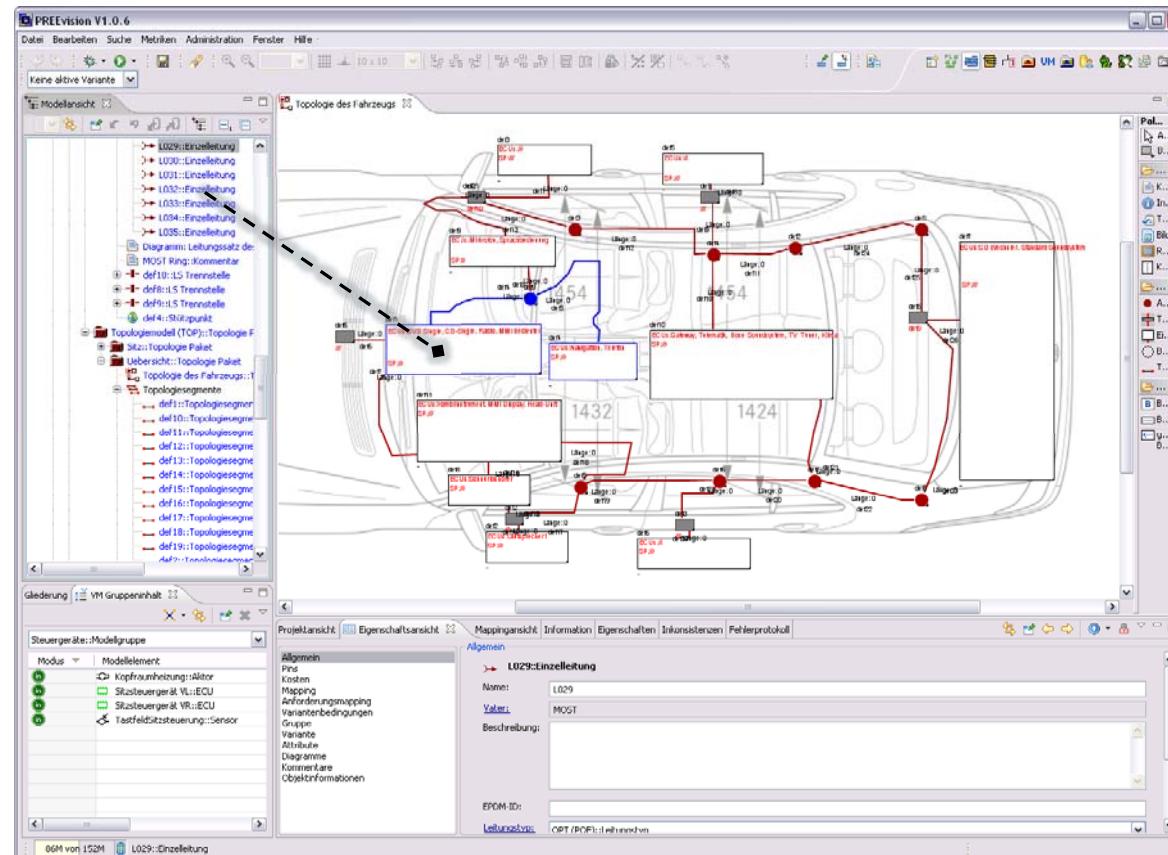


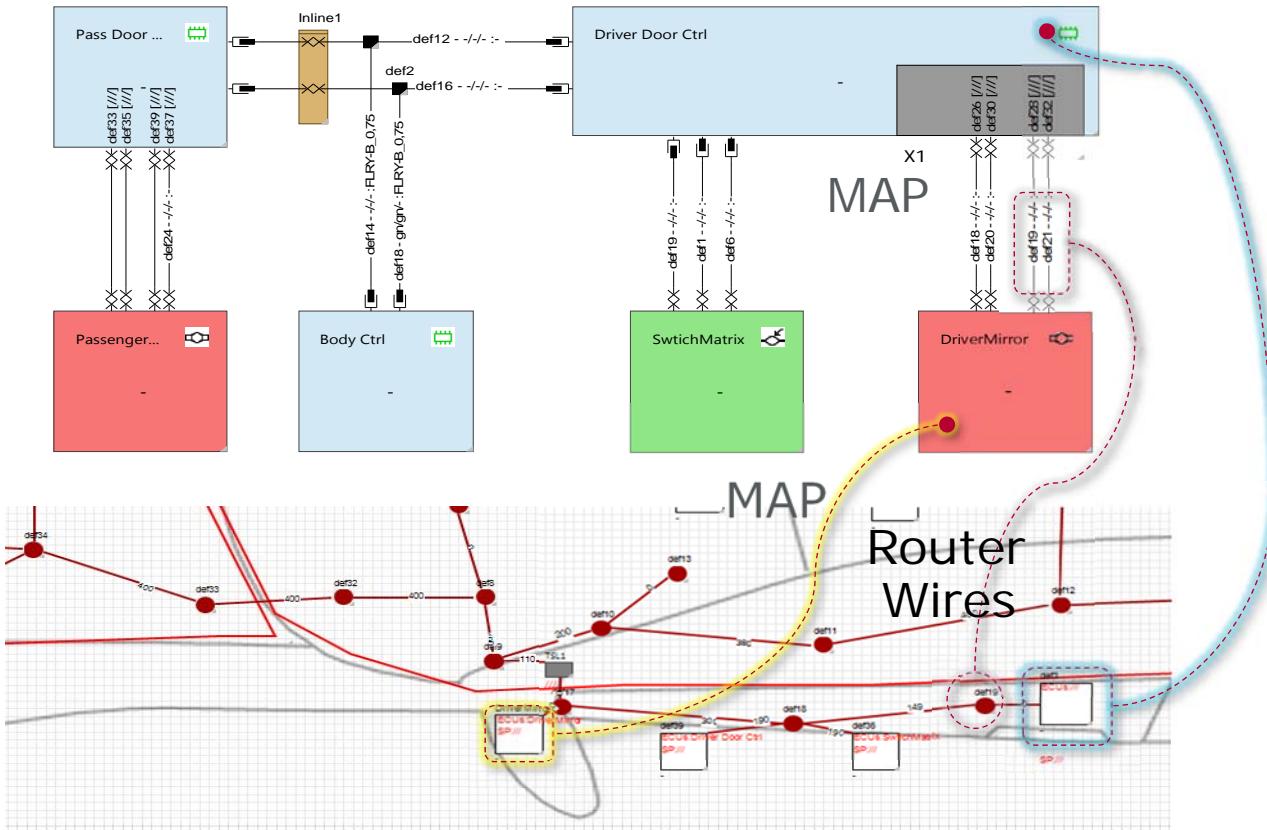
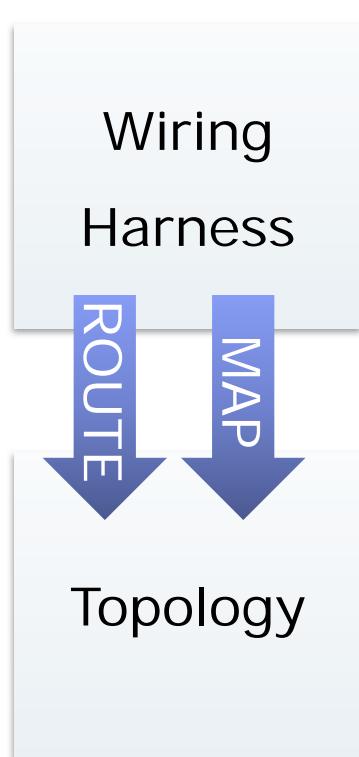


2D Topologie in PREEvision

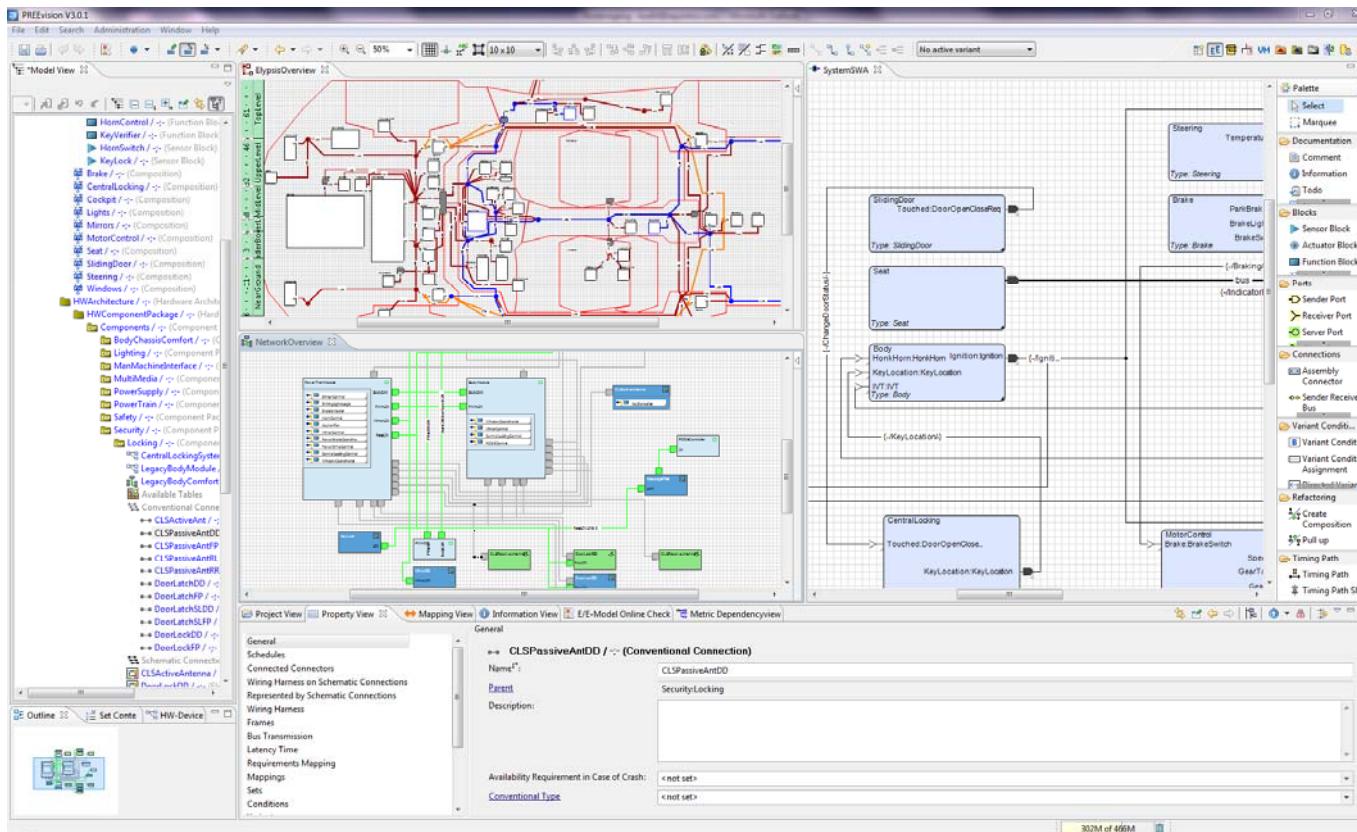


vector >



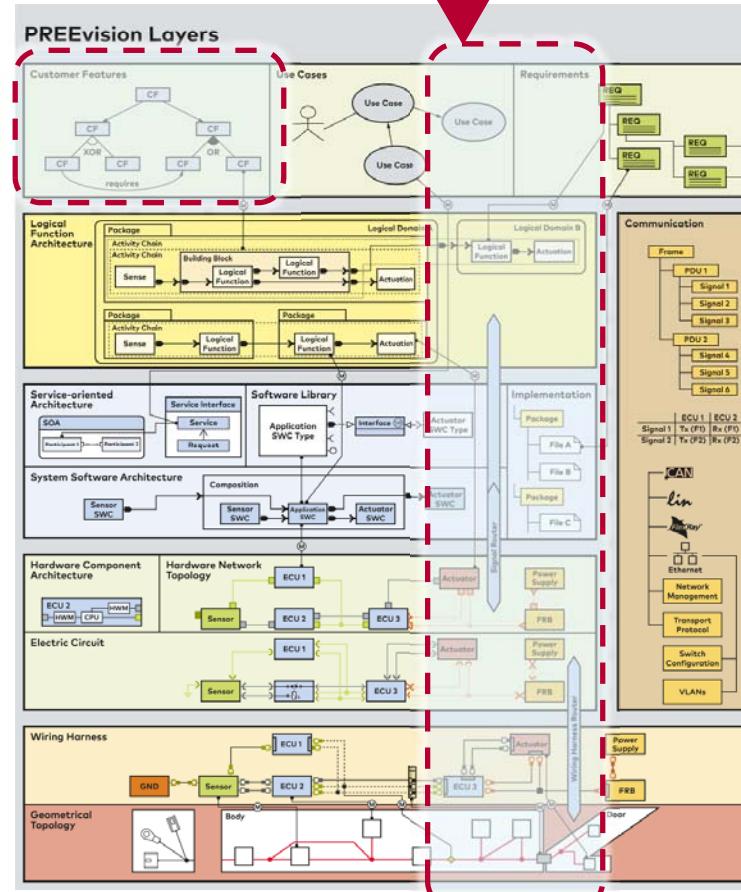


PREEvision Customer Benefits GUI concept



PREEvision Collaboration Platform (VII)

Varianten



Use Case: Variant Management Solution Space in Product Lines

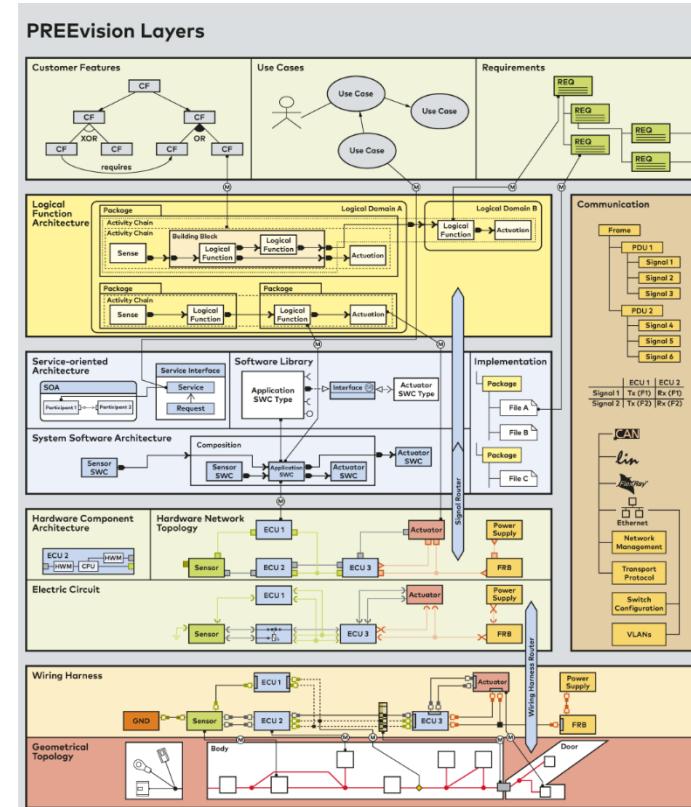
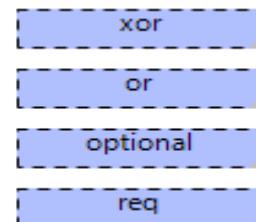
Explicit vs. Implicit Definition

Explicit – Variants as Representatives

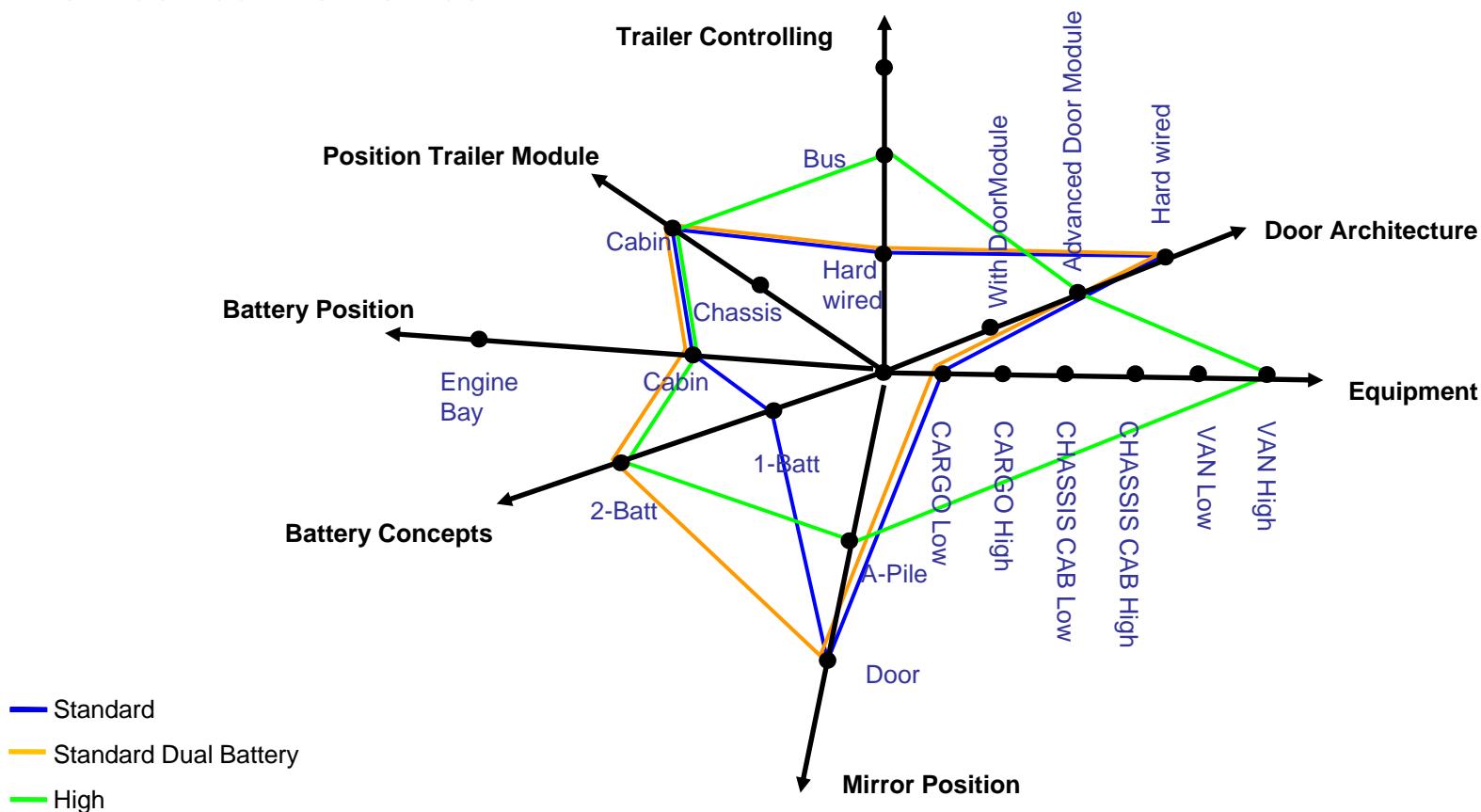
- ▶ Concept Selection by Variant
- ▶ Set and System Content
On all Modeling Layers
And in between

Implicit

- ▶ Variant Structure
(Hierarchy)
- ▶ Propagation Rules
- ▶ Variant Conditions



Beispiel Architekturvarianten

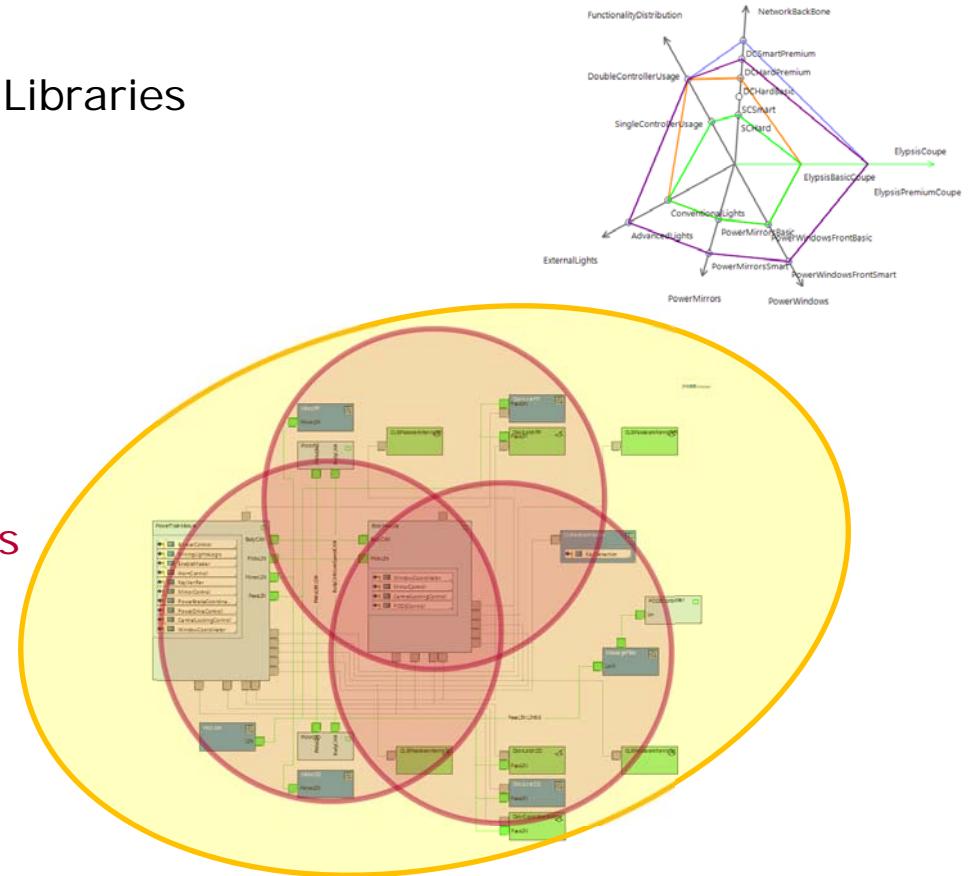


Use Case: Variant Management Super-Set Modeling

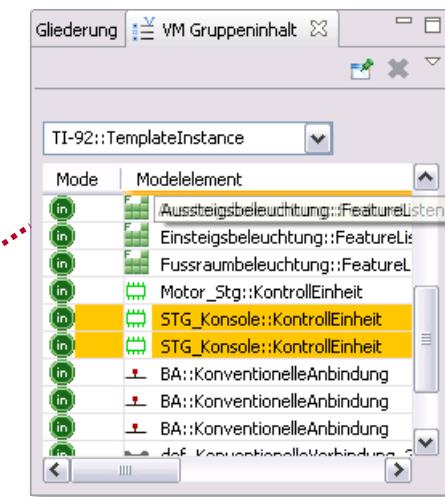
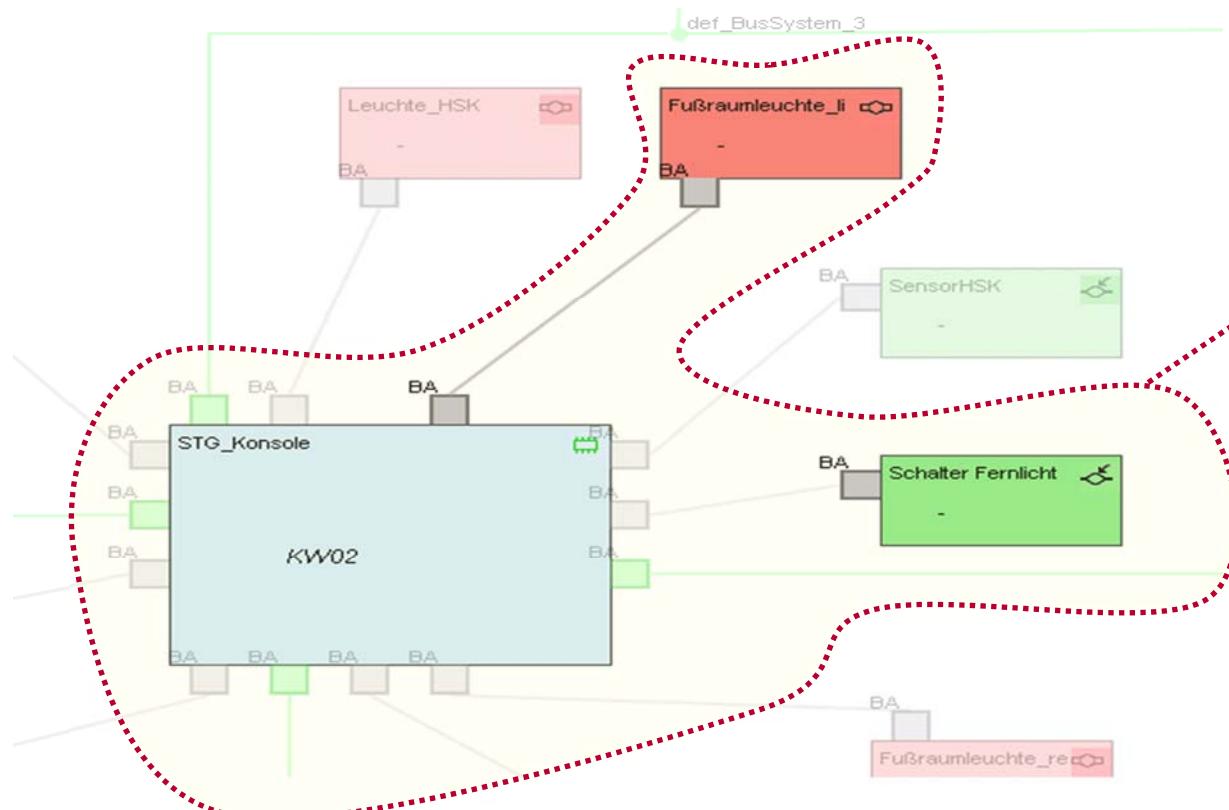
'150% Modeling' of Product Lines and Libraries

- ▶ **Technical Concepts**
Definition and Selection
- ▶ **Equipment**
Defining Configurations of Features
- ▶ **Building**
Representative Architecture Variants

- ▶ Set-based Structuring
- ▶ System Integration

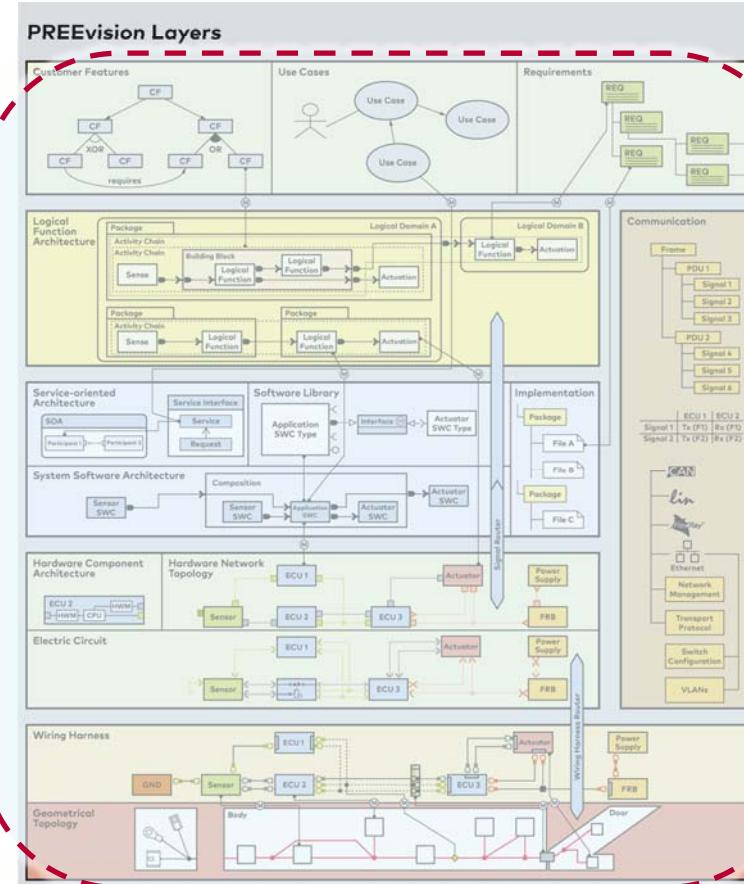


Bilden von Varianten

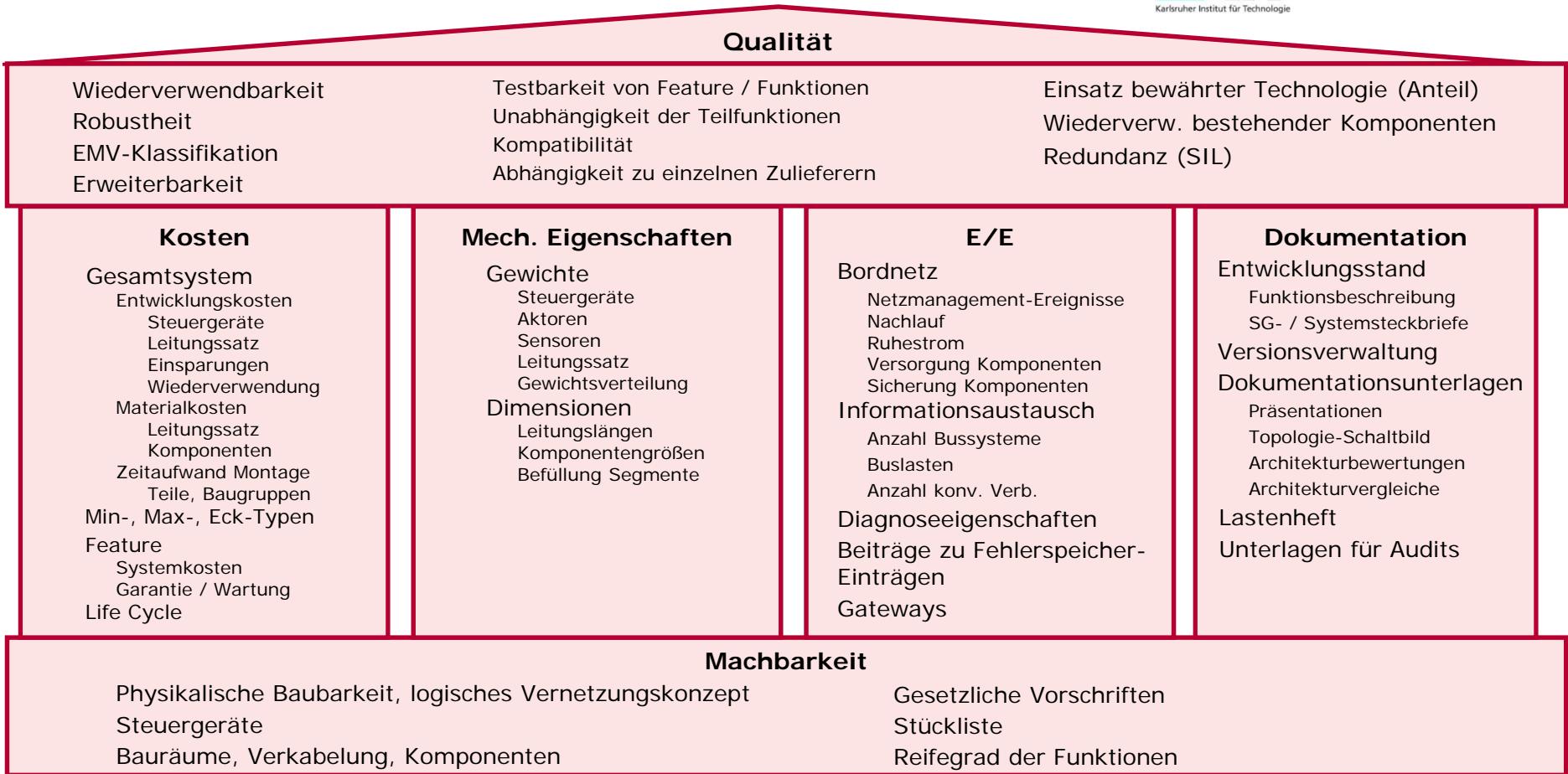


PREEvision Collaboration Platform (VIII)

Metriken

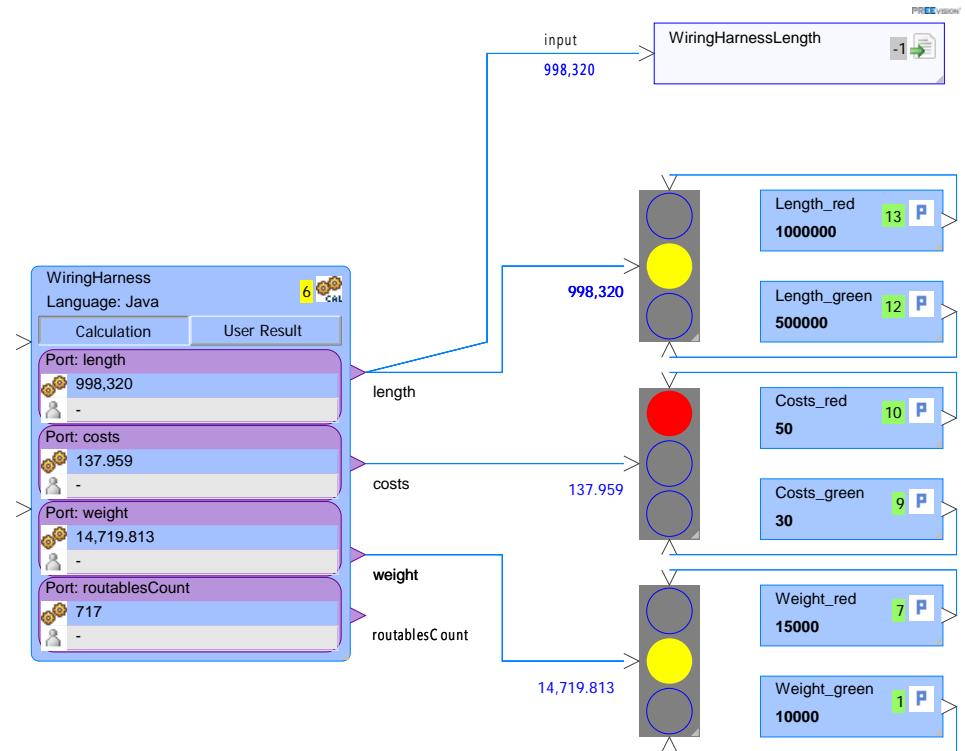


Bewertungskriterien einer E/E-Architektur

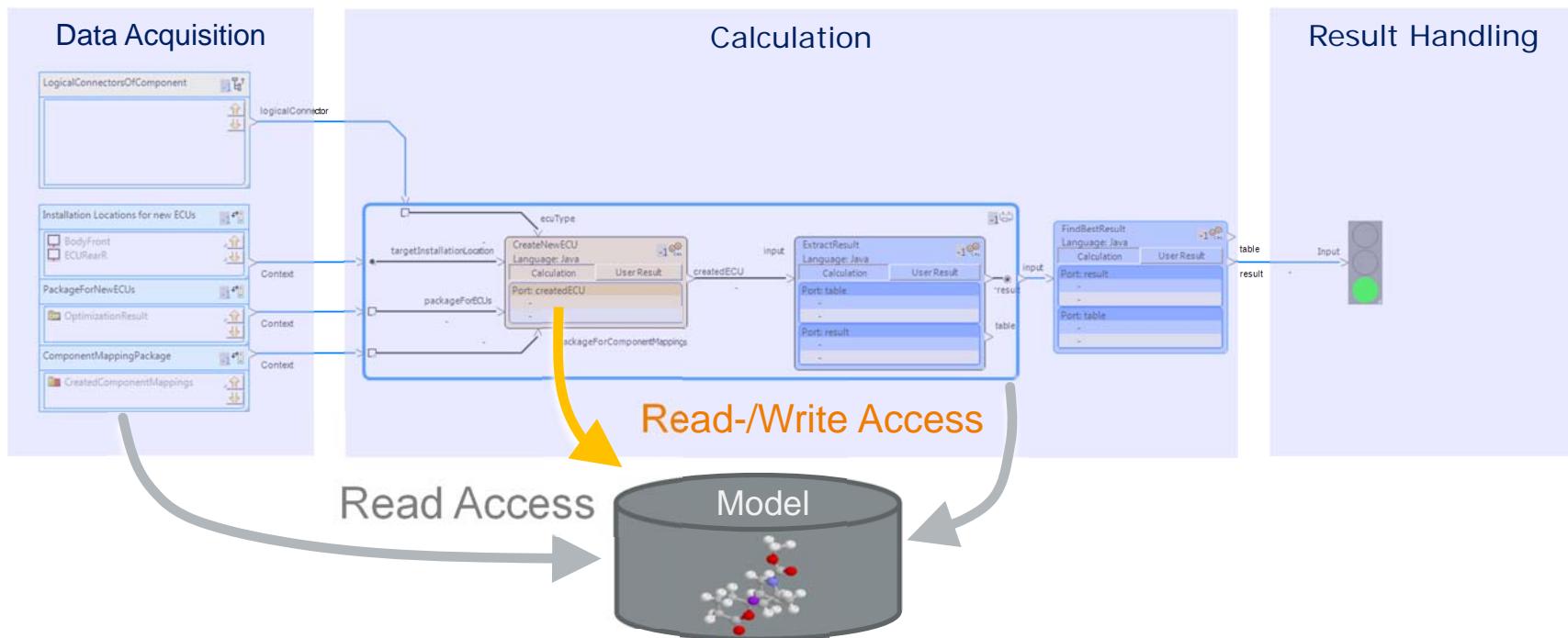


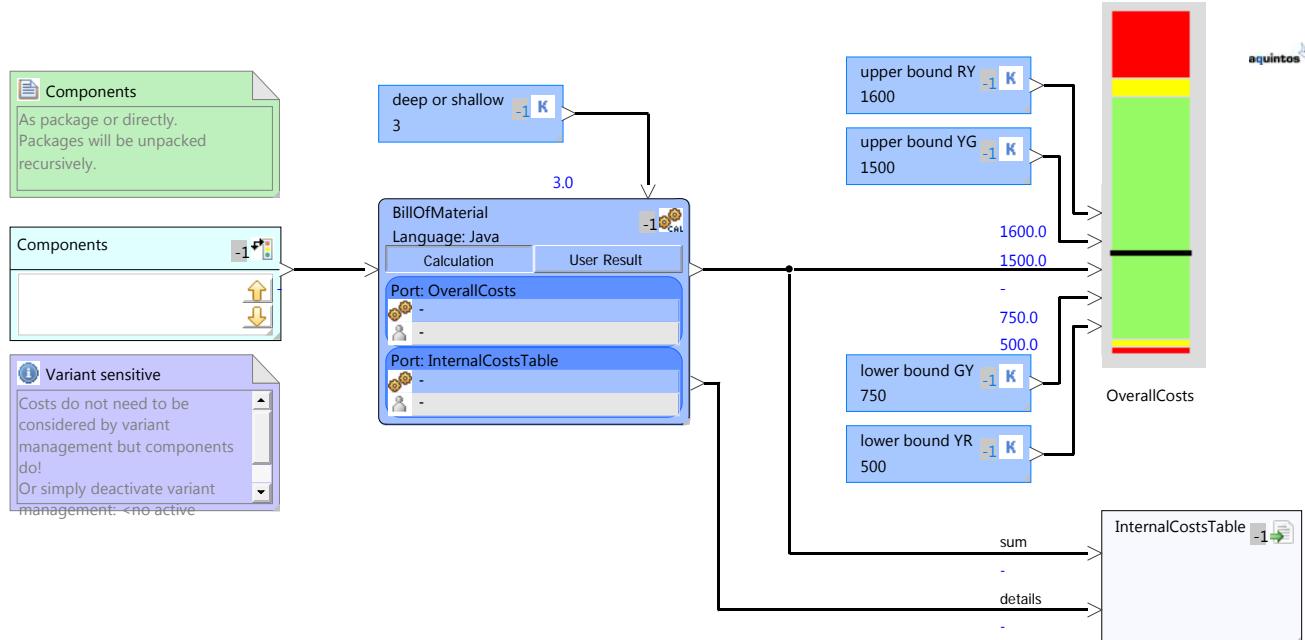
Metrics

- ▶ Are used to perform **calculations** on the data model for analysis and optimization.
- ▶ Can be widely used to enrich tool with customer **individual IP**
- ▶ Always run on the full model – the result is up to date.
- ▶ Output of metrics is streamed into **Reports** or into **GUI** as lights, scales or values.



- Are based on a graphical notation and can be expanded by Java-code

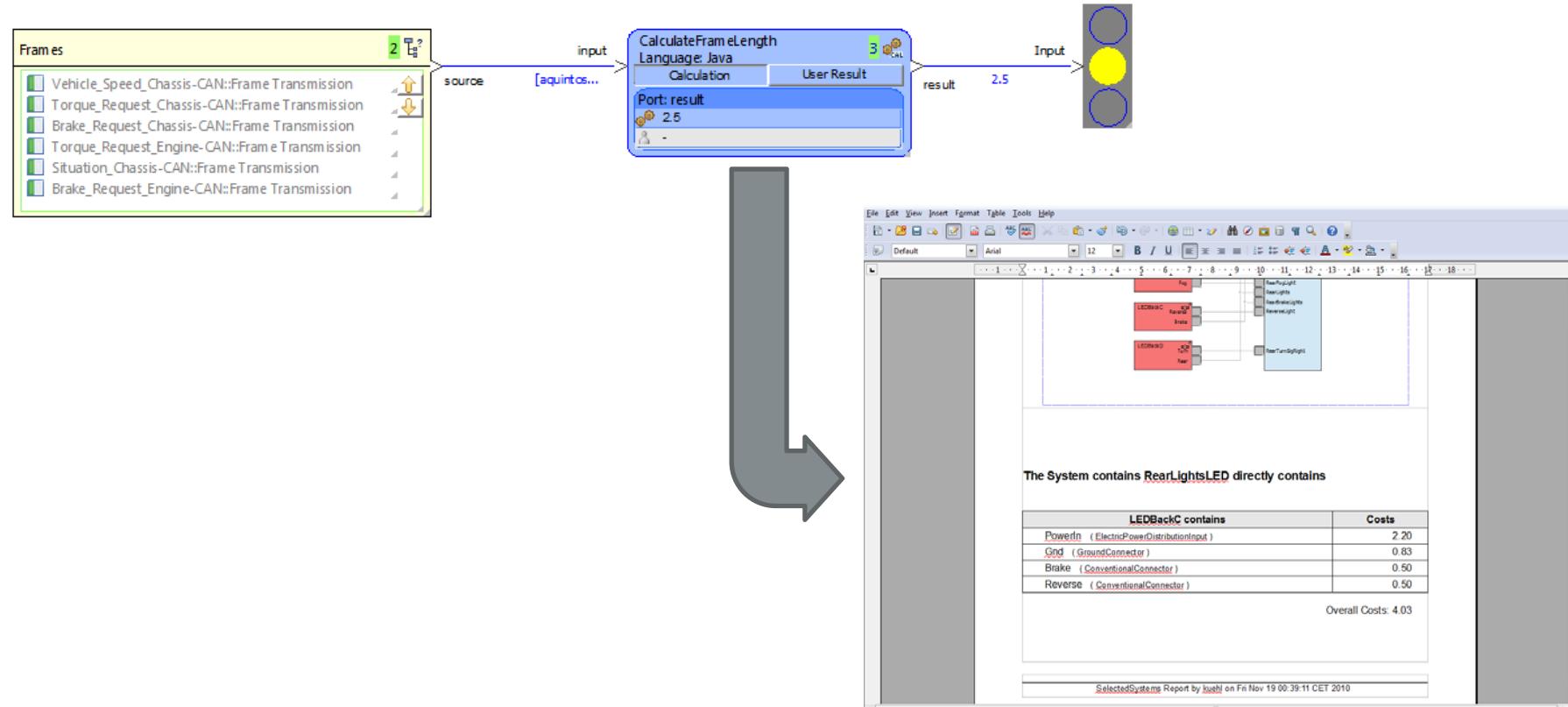




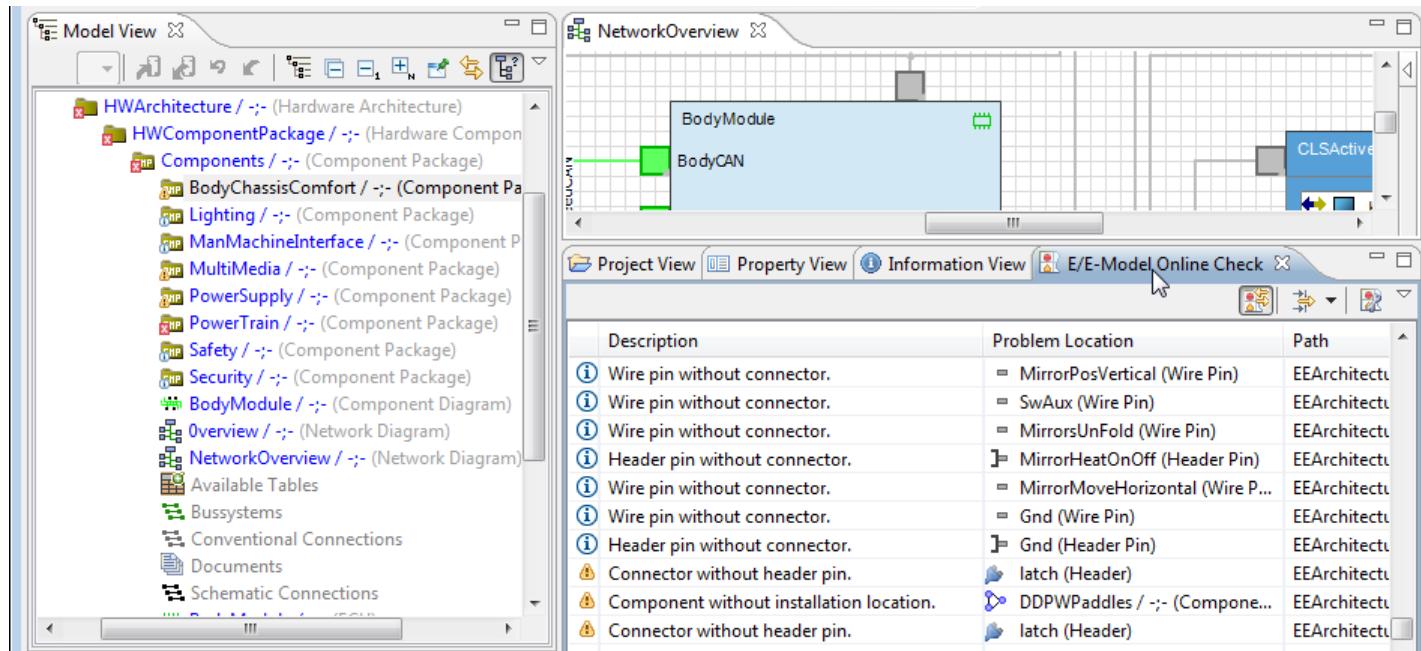
- ▶ Metrics are used to **compute** different evaluation **criteria**
- ▶ **Output** of metrics is streamed into **Reports** or into **GUI**

Metrics

Use Case: Report Generator



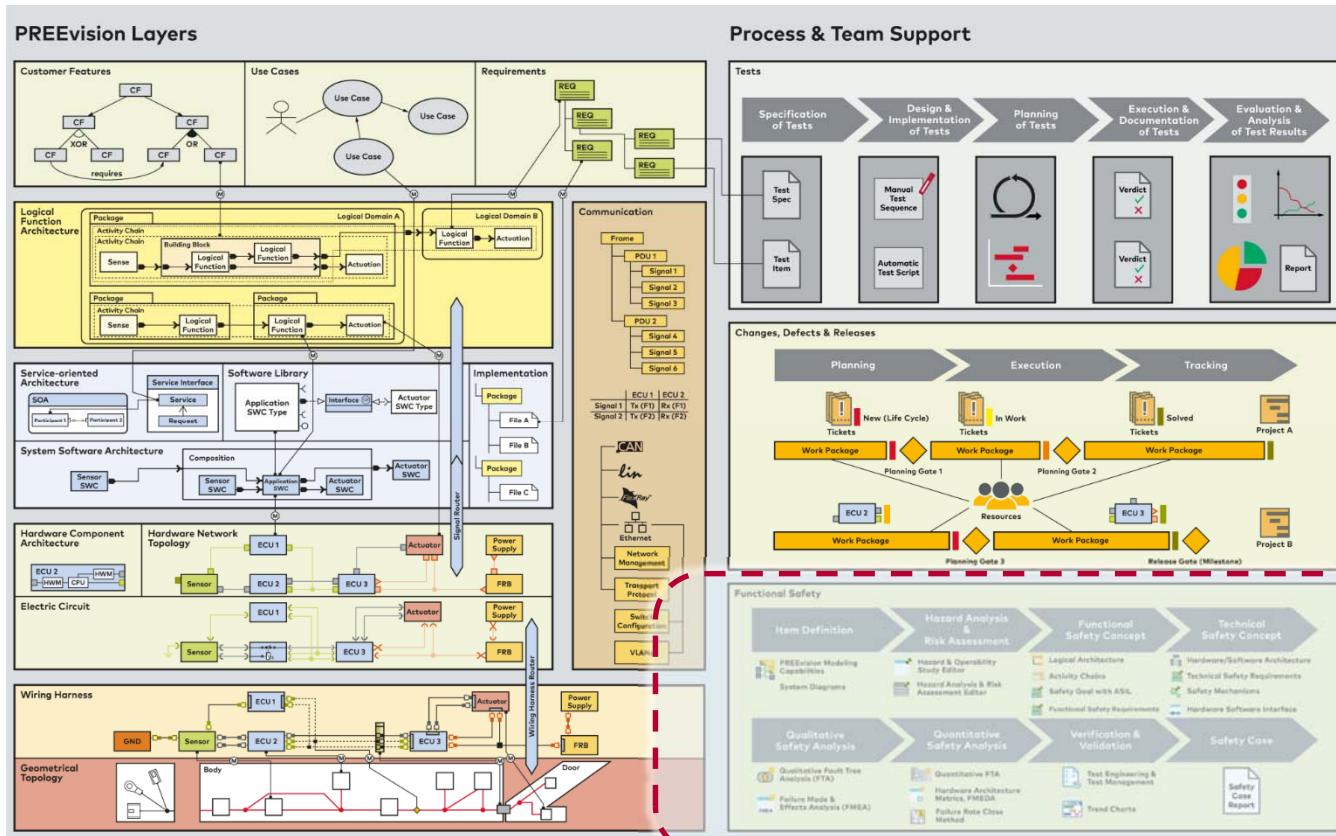
Consistency Checker



- ▶ Errors, Warnings, Information on **Architecture or selected Parts**
- ▶ Relevant **Consistency Checks** to be selected/extended
- ▶ Interactive **To Do List**

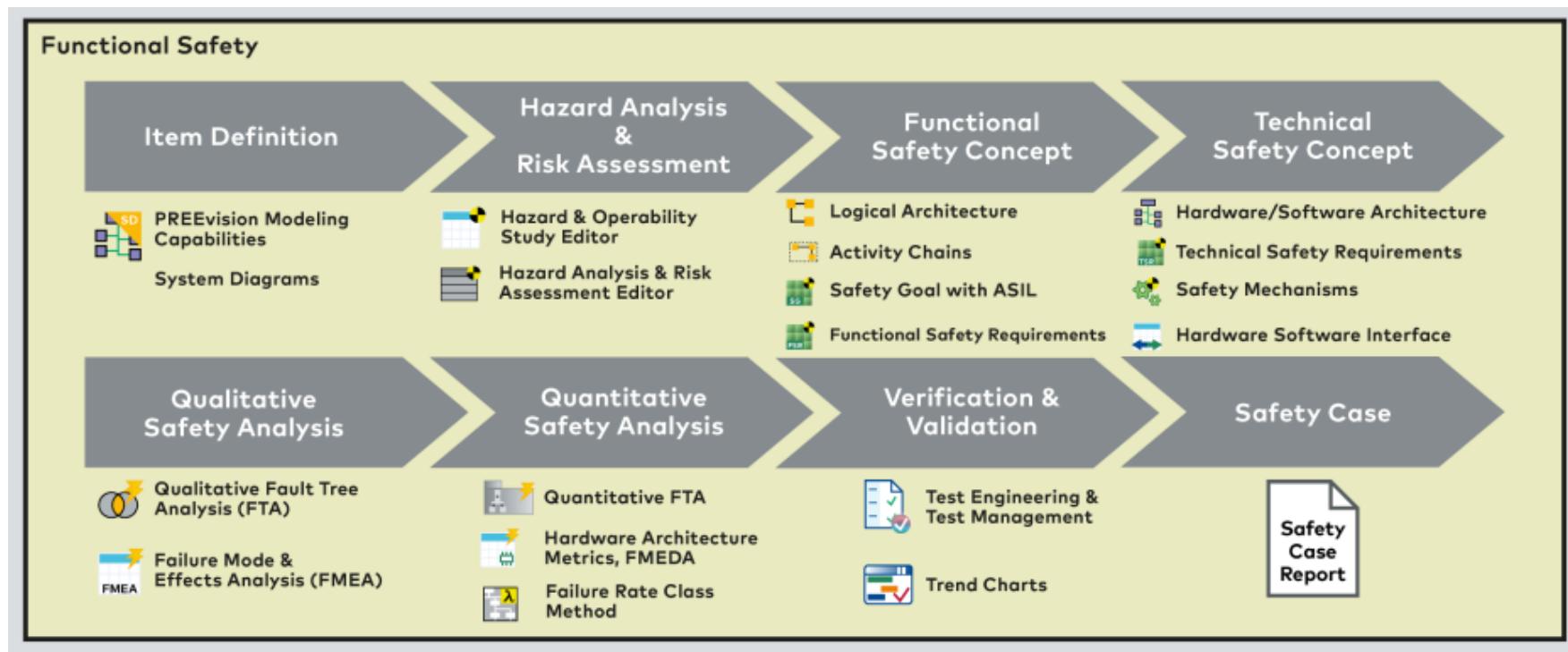
PREEvision Collaboration Platform (IX)

Safety



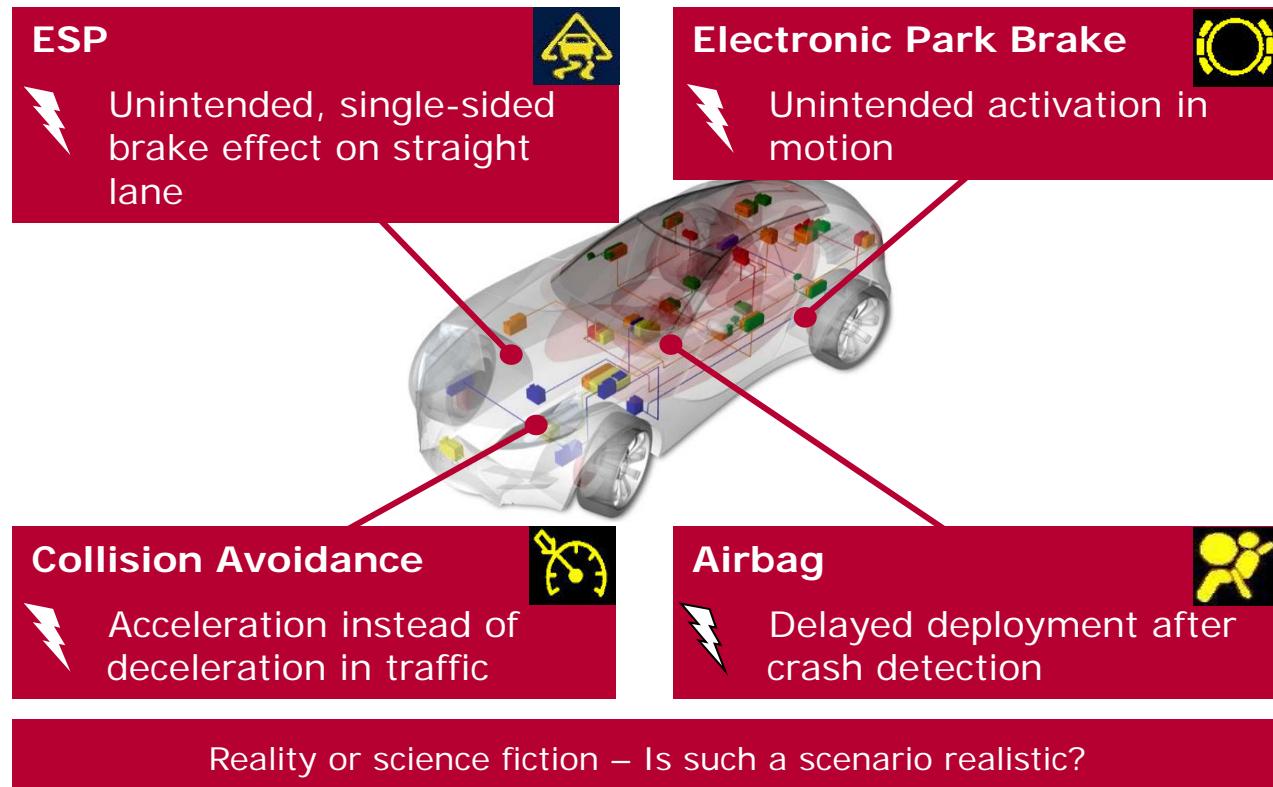
PREEvision Collaboration Platform (IX)

Safety



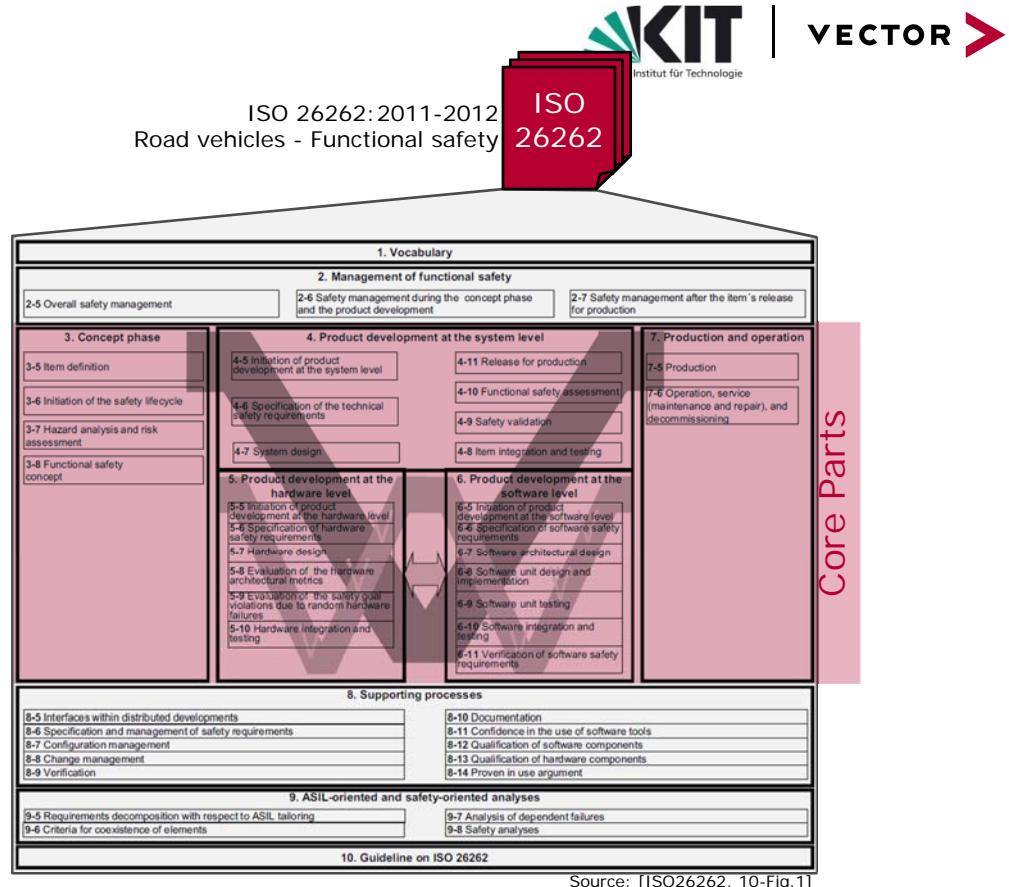
Introduction Functional Safety

Motivation: Many Functions are Safety-Critical



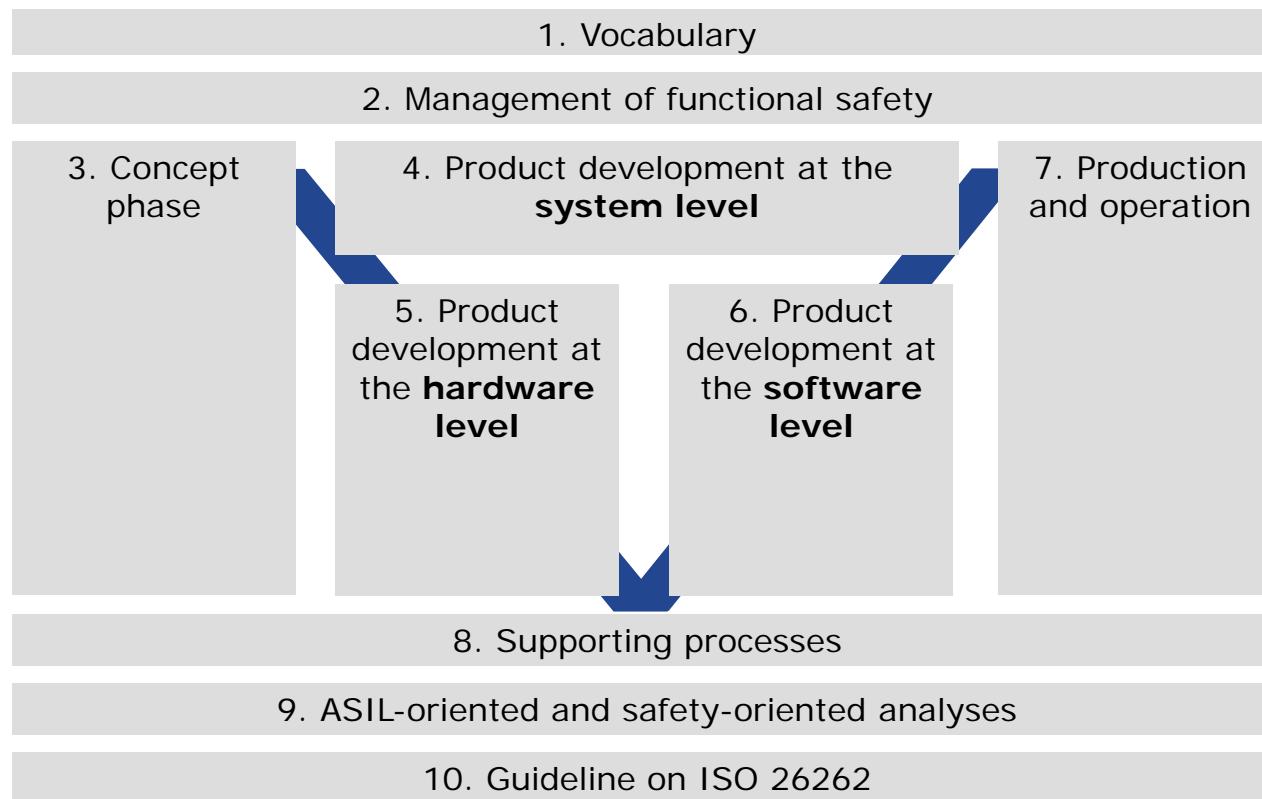
Introduction Functional Safety Challenges

- ▶ 10 Parts
- ▶ 43 Chapters
- ▶ 100 Work products
- ▶ 180 Engineering methods
- ▶ 500 Pages
- ▶ 600 Requirements



▶ Complex standard → Risk of overheads and costs if applied ad hoc

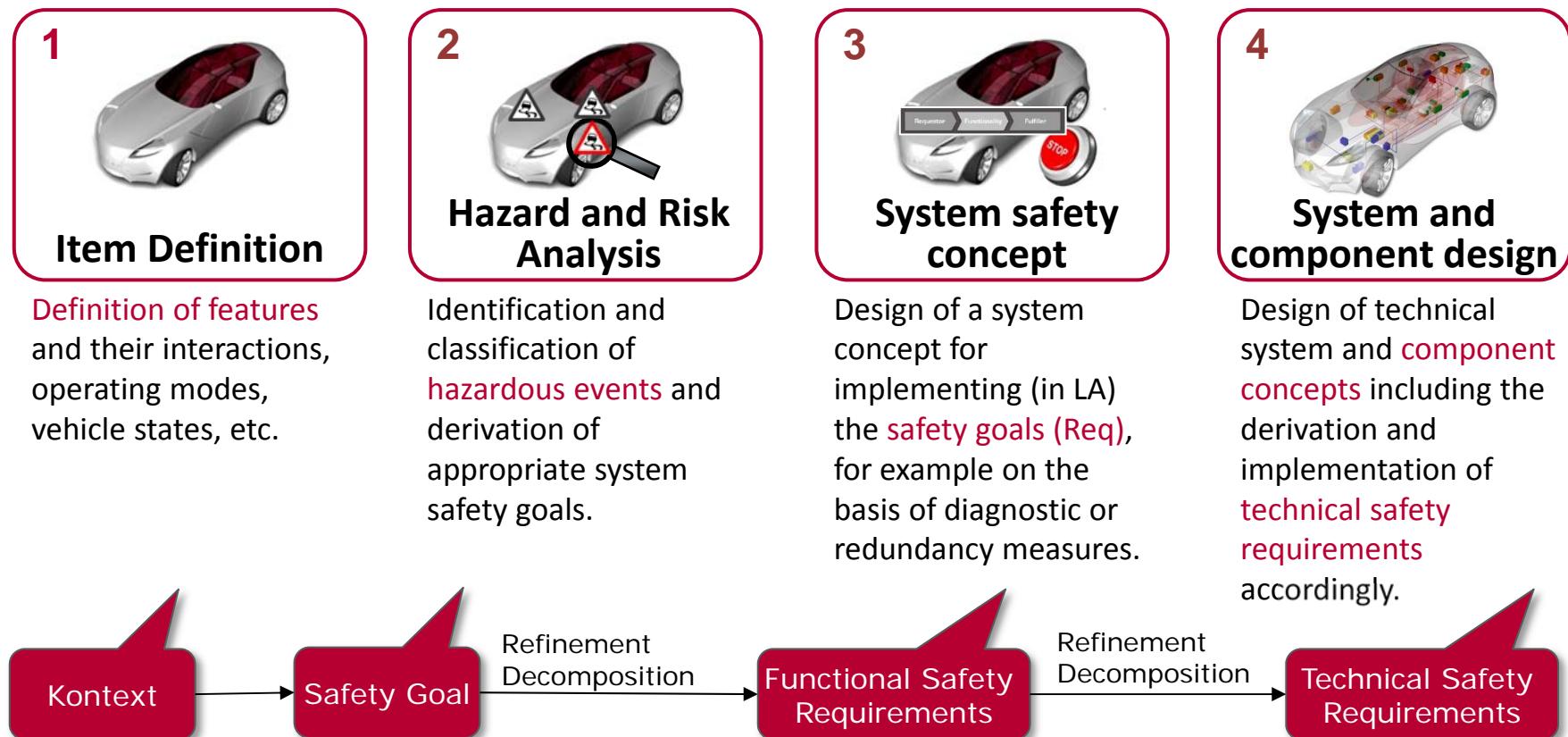
Introduction Functional Safety Parts of ISO 26262



Overview Functional Safety in PREEvision Integrated Model Based System Engineering Platform

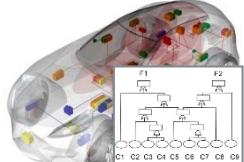


The safety lifecycle according to ISO 26262



The safety lifecycle according to ISO 26262

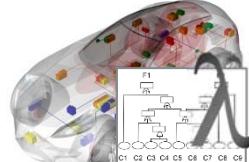
5



Qualitative Safety Analyses

Application of deductive and inductive safety analysis techniques (e.g. FTA, FMEA) to validate the ability of the design to meet the system safety goals.

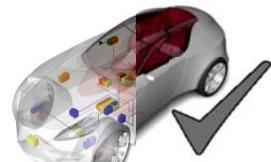
6



Quantitative Safety Analyses

Calculation of the probability of the system failing to meet the safety goals and confirmation that the failure rate and diagnostic coverage targets are met.

7



Verification and Validation

Confirmation through review, analysis and test that all safety requirements are correctly implemented in the delivered system and that all assumptions made in the safety concept are valid.

8

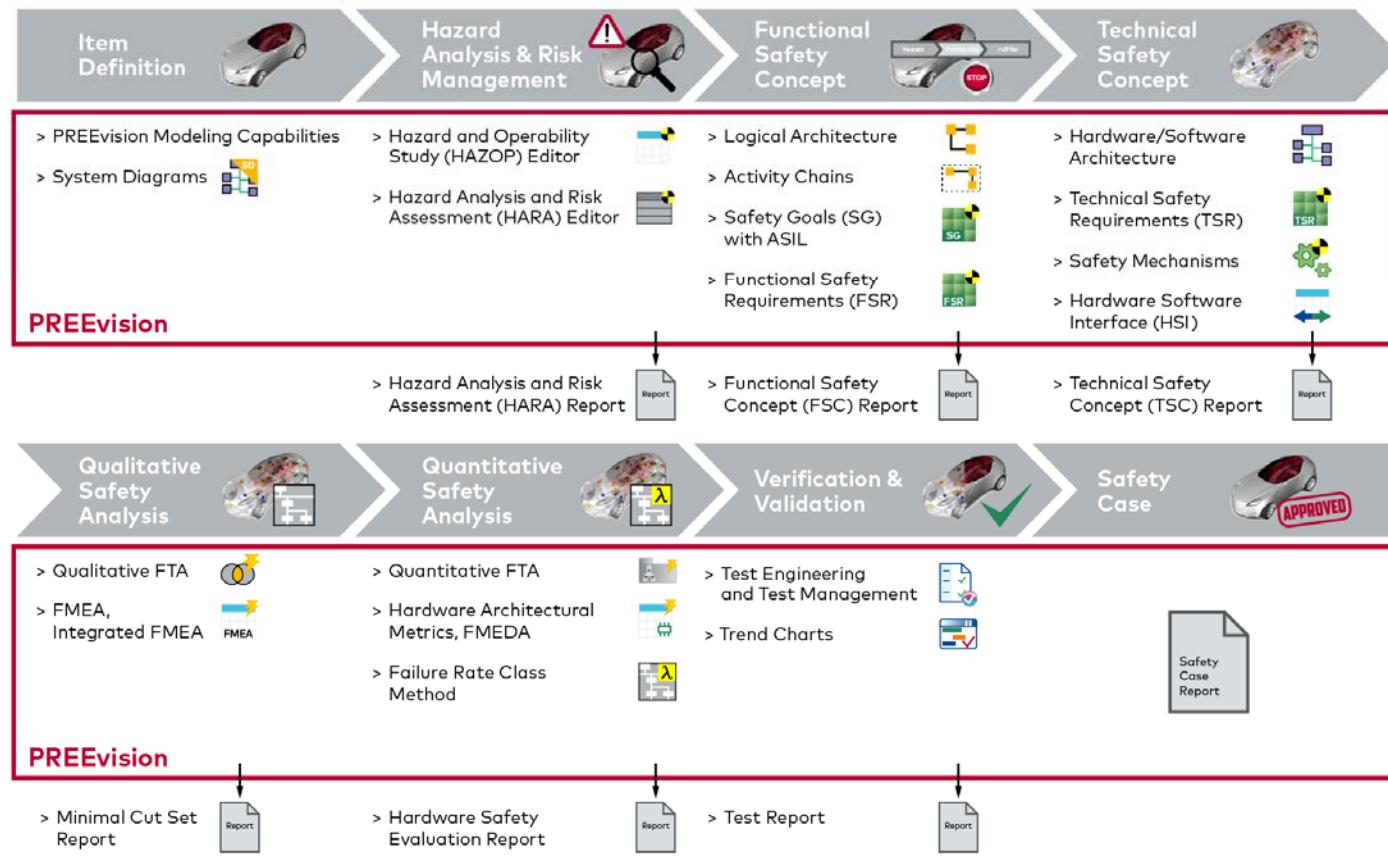


Safety Case

Construction of a structured, coherent, complete and convincing argument that the system meets all its safety goals and appropriate regulations.

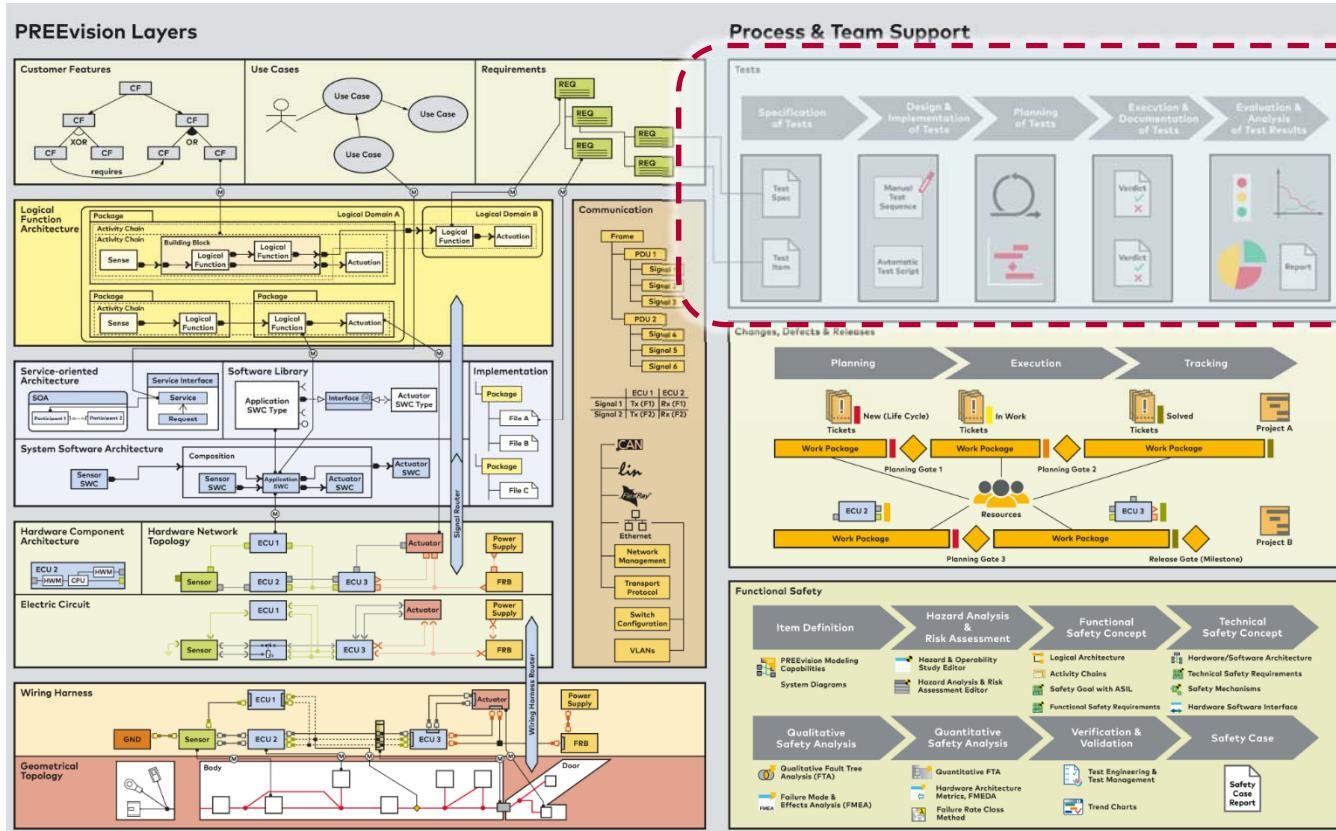
Overview Functional Safety in PREEvision

Supported steps of the safety lifecycle



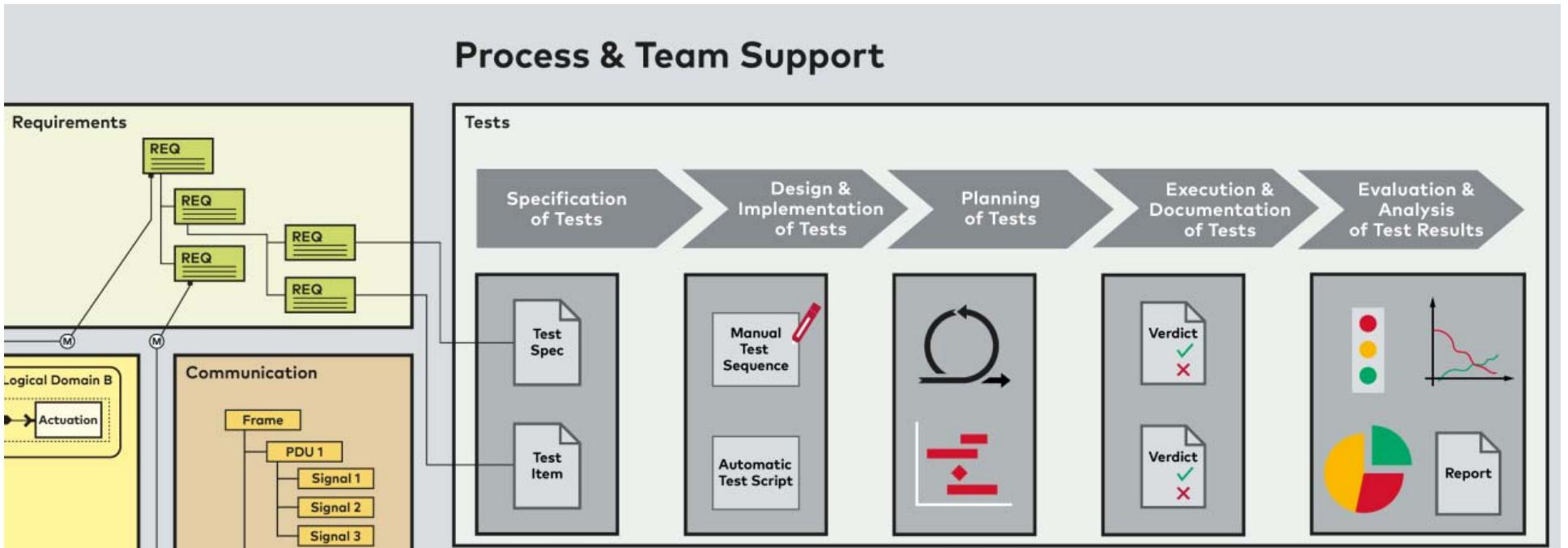
PREEvision Collaboration Platform (IX)

Test Daten Management

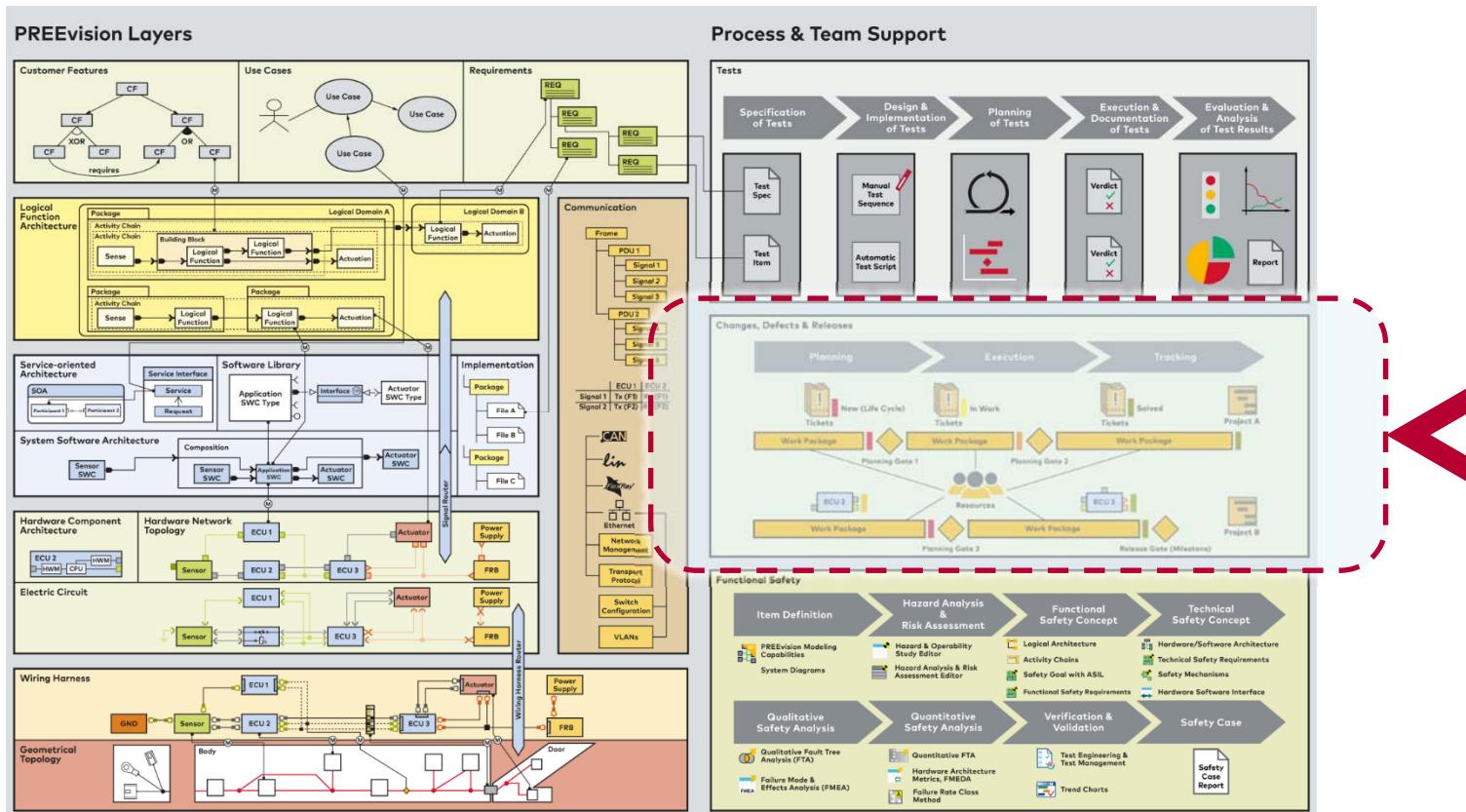


PREEvision Collaboration Platform (IX)

Test Daten Management

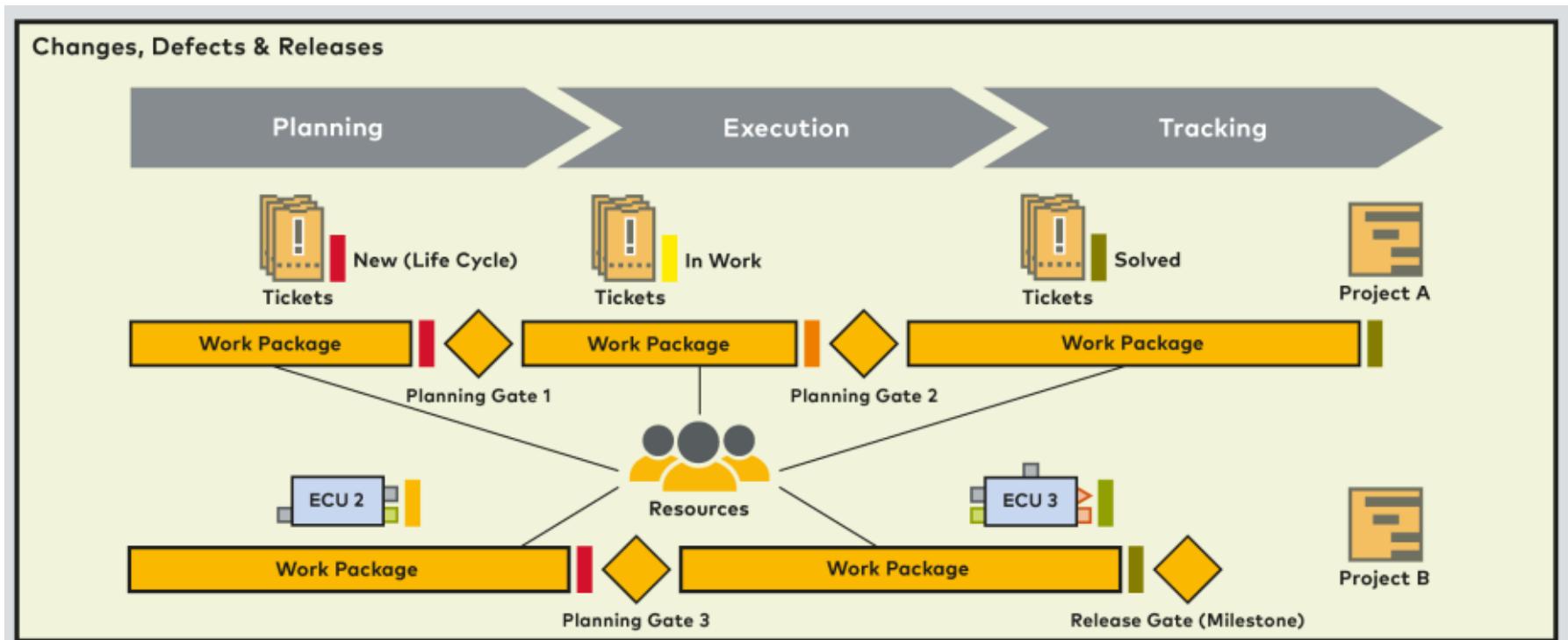


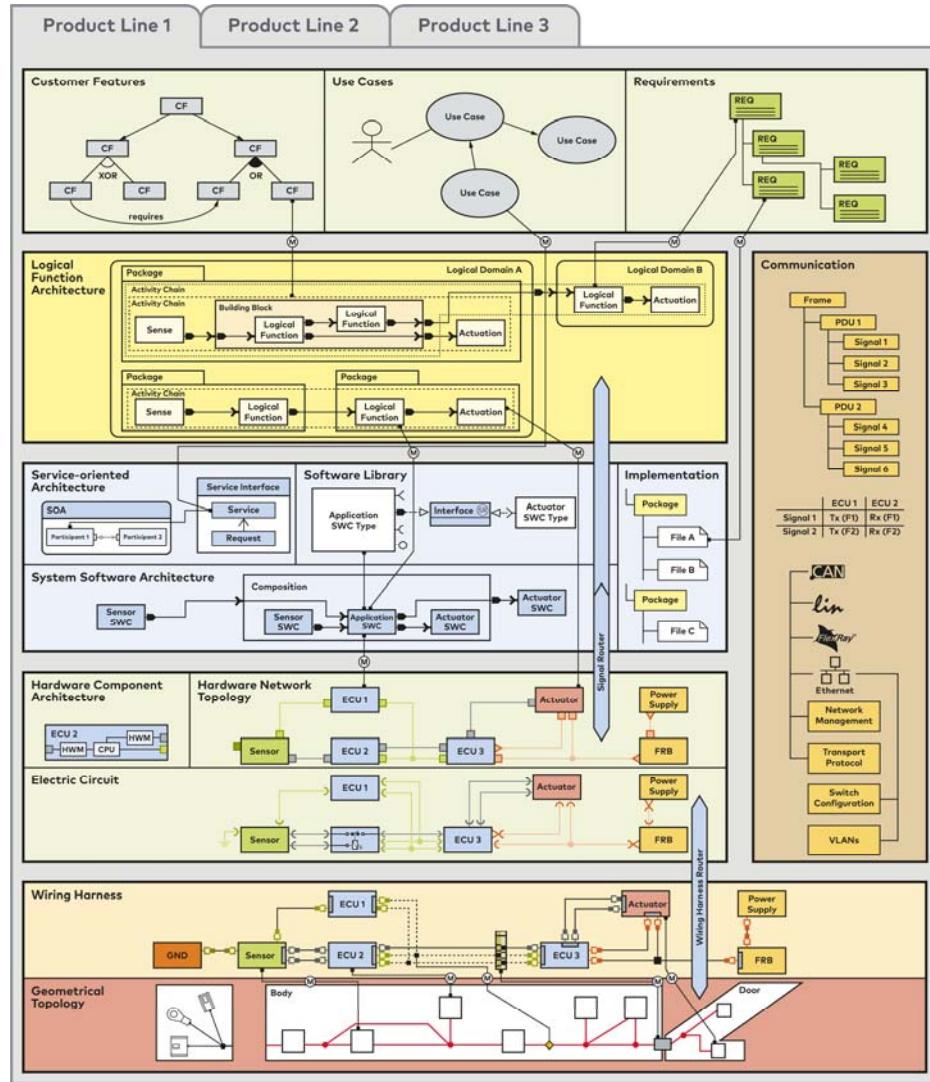
PREEvision Collaboration Platform (IX) Change Management



PREEvision Collaboration Platform (IX)

Change Management





PREEvision Layers

Requirements

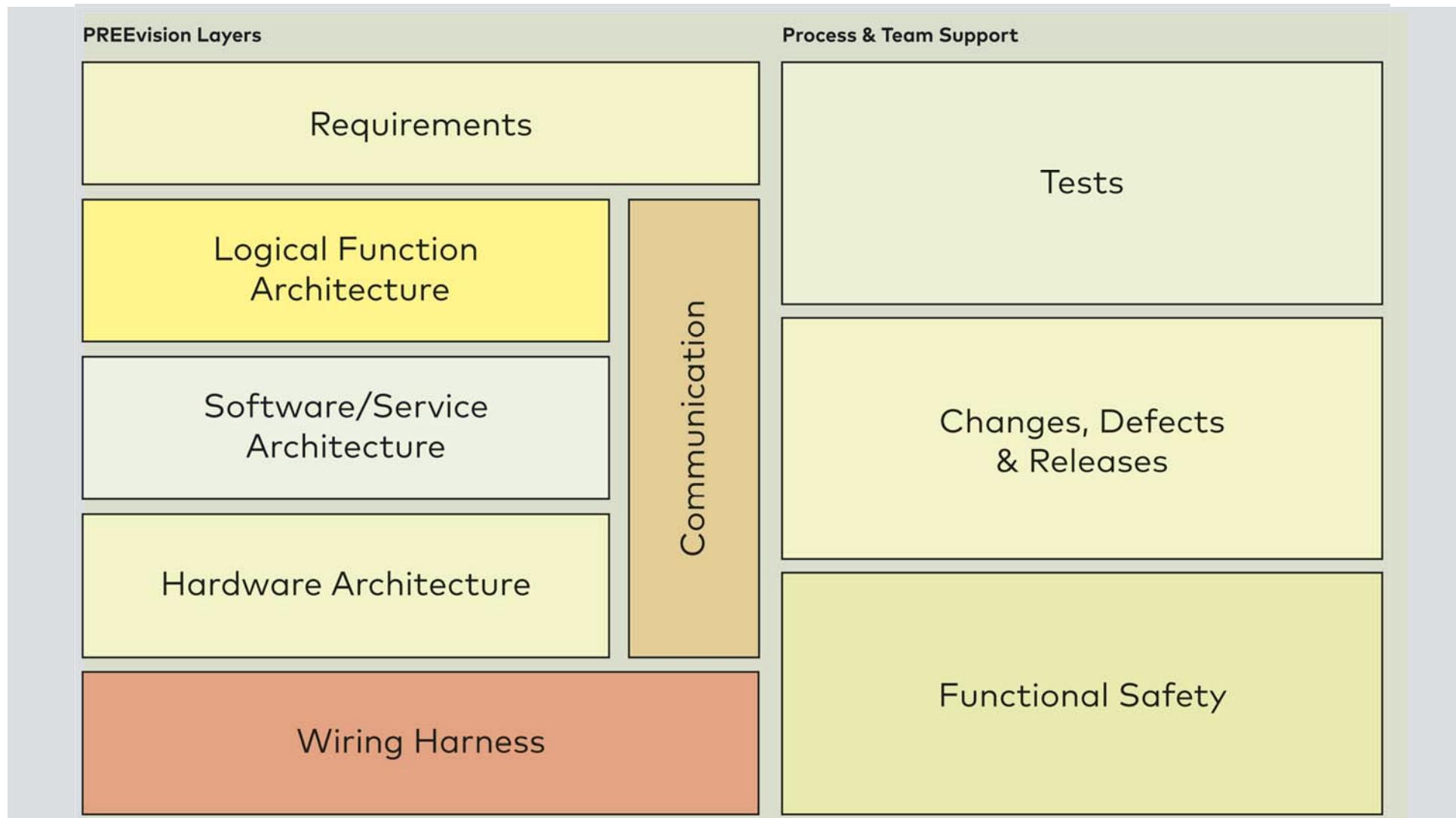
Logical Function Architecture

Software/Service Architecture

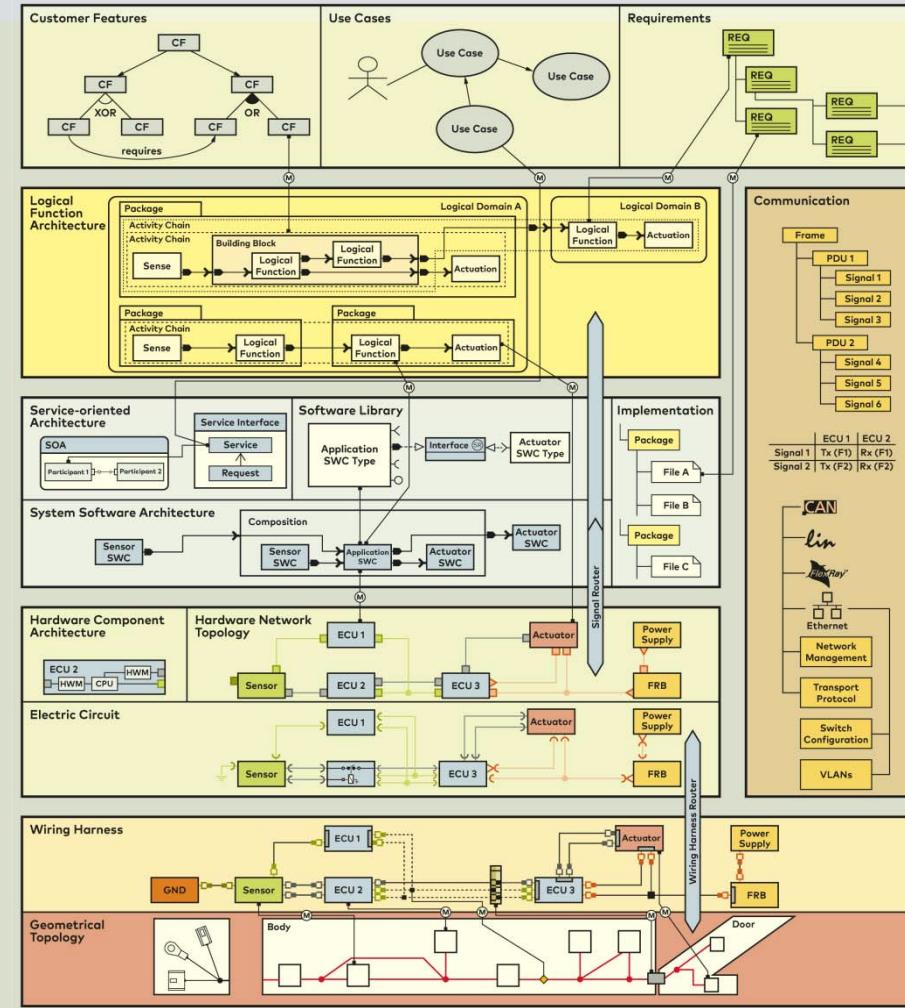
Hardware Architecture

Wiring Harness

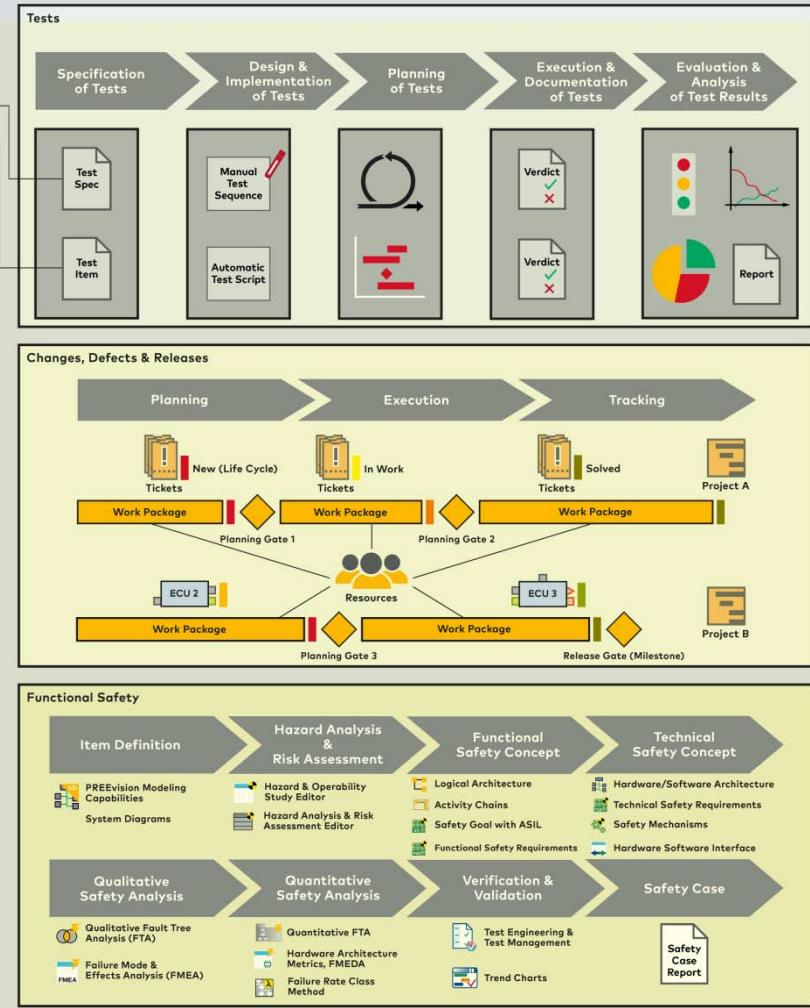
Communication



PREEvision Layers



Process & Team Support



Thank you for your attention.

For detailed information about Vector
and our products please have a look at:

www.vector.com

Authors

Dr. Clemens Reichmann
Productline Process Tools
Vector Informatik GmbH, Stuttgart
aquintos GmbH, Karlsruhe