FZI
Forschungszentrum Informatik
an der Universität Karlsruhe

RST'07, 26-June-2007

„Challenges for reliable software design in automotive electronic control units“

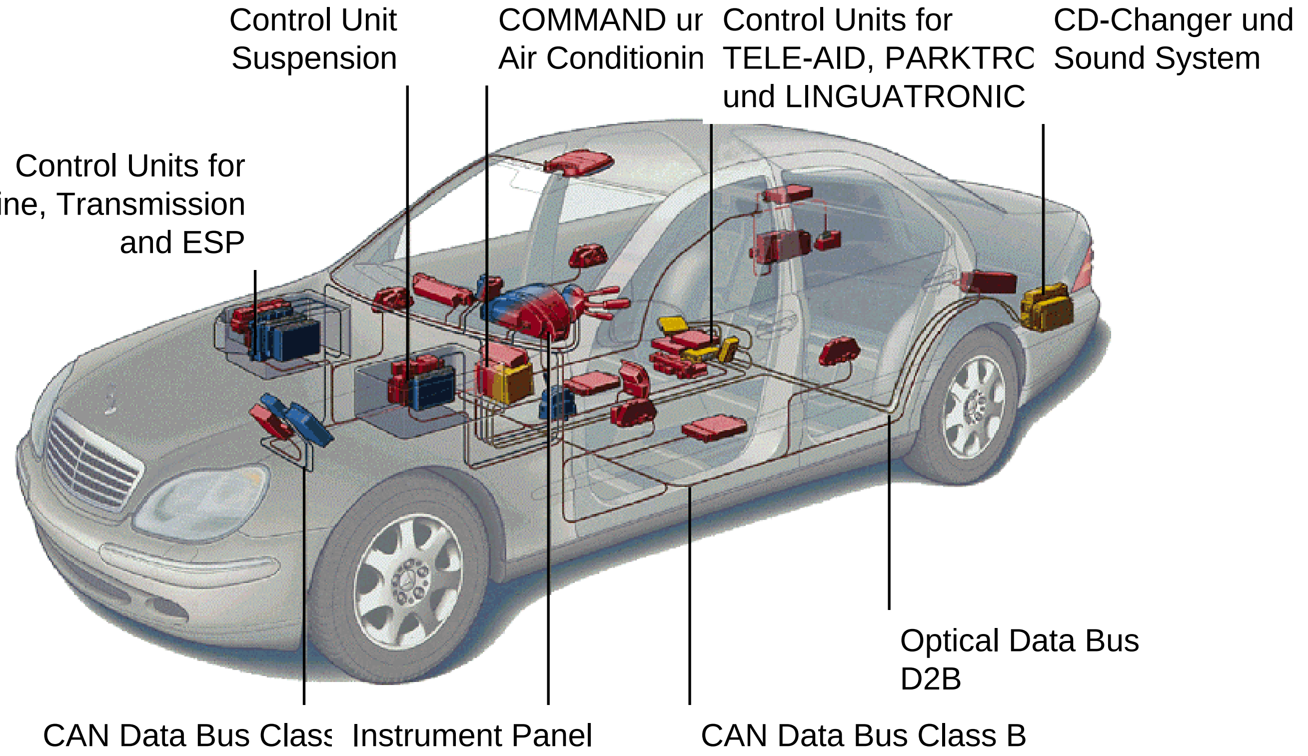
Prof. Dr.-Ing. Klaus D. Müller-Glaser

Contents

System under Design: automotive electronic control units

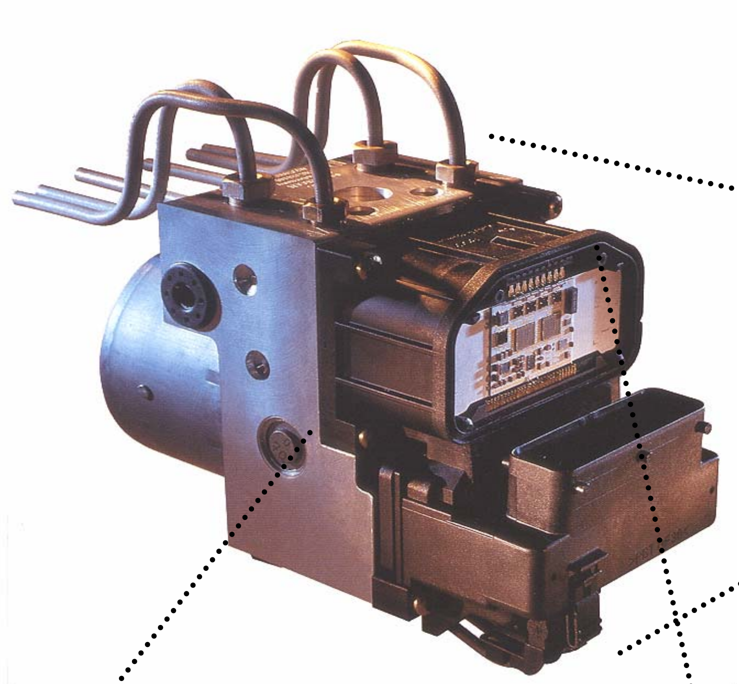
Characteristics: distributed system,
complex distributed functionality

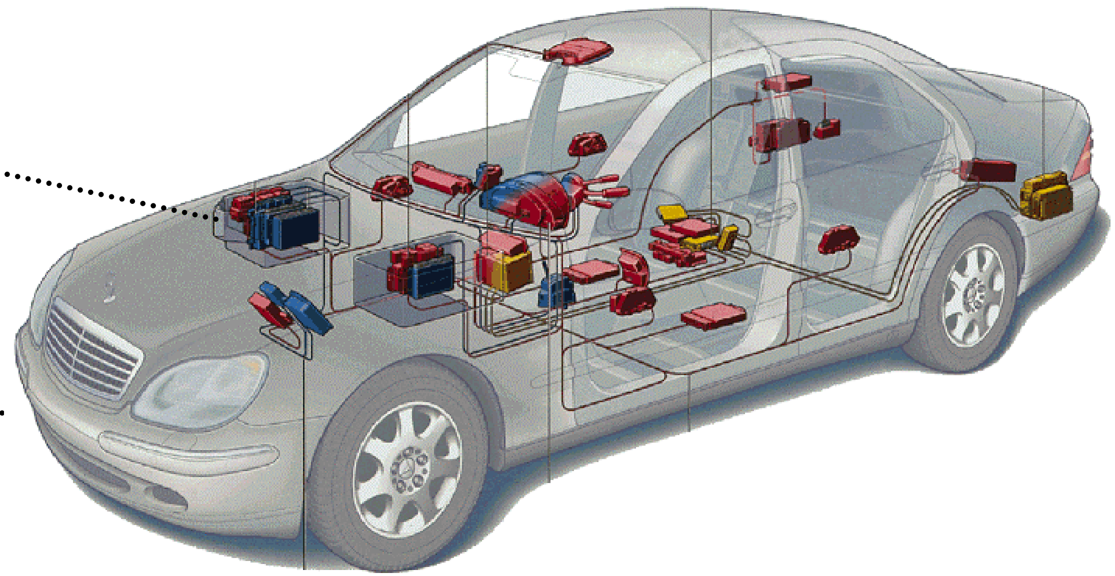
in a premium class car
up to 80 computers (Electronic Control Units ECU)
> 100 Electrical Motors, > 2 km Wiring
millions of lines of code

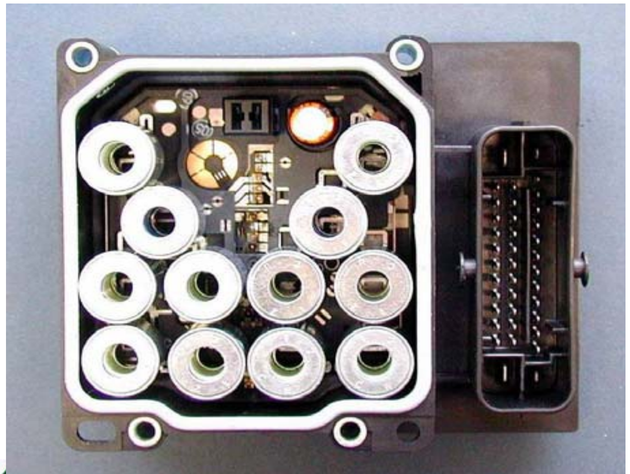


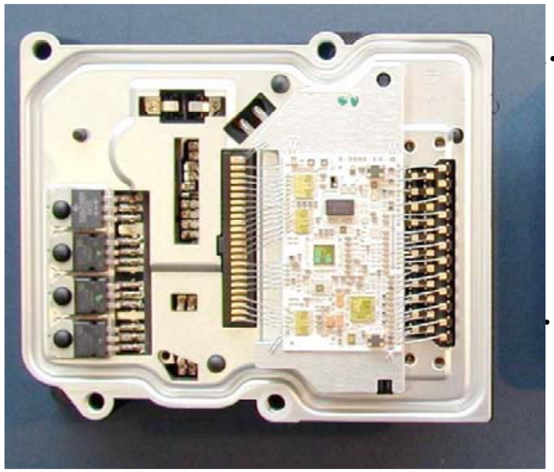
courtesy DaimlerChrysler AG

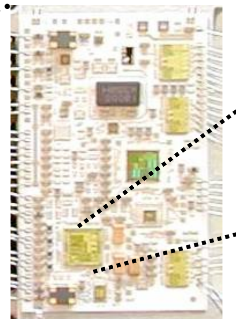
Characteristics: distributed, mechatronic











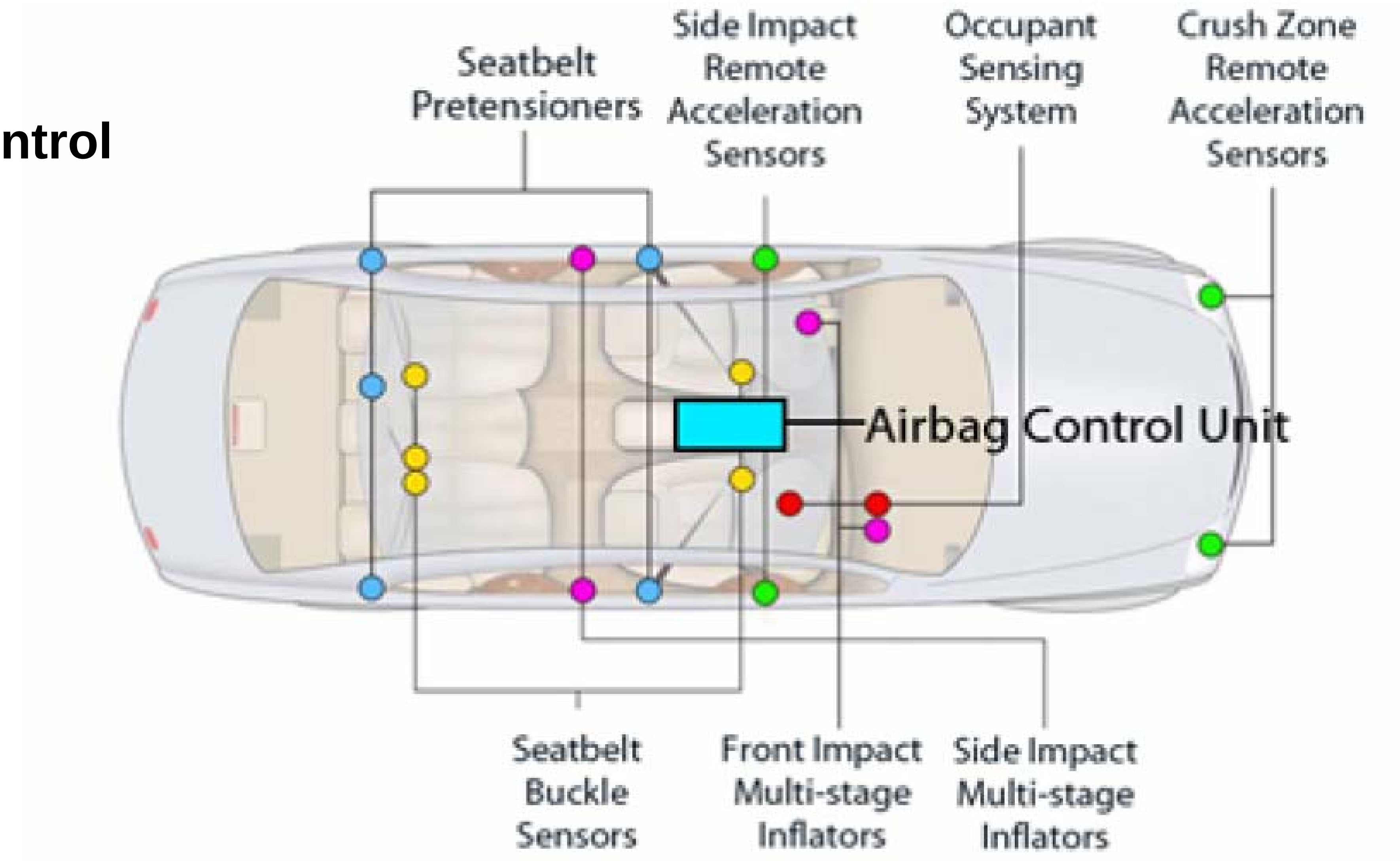
Software is part of ECU

ECU's are part of mechatronic systems for measurement and control

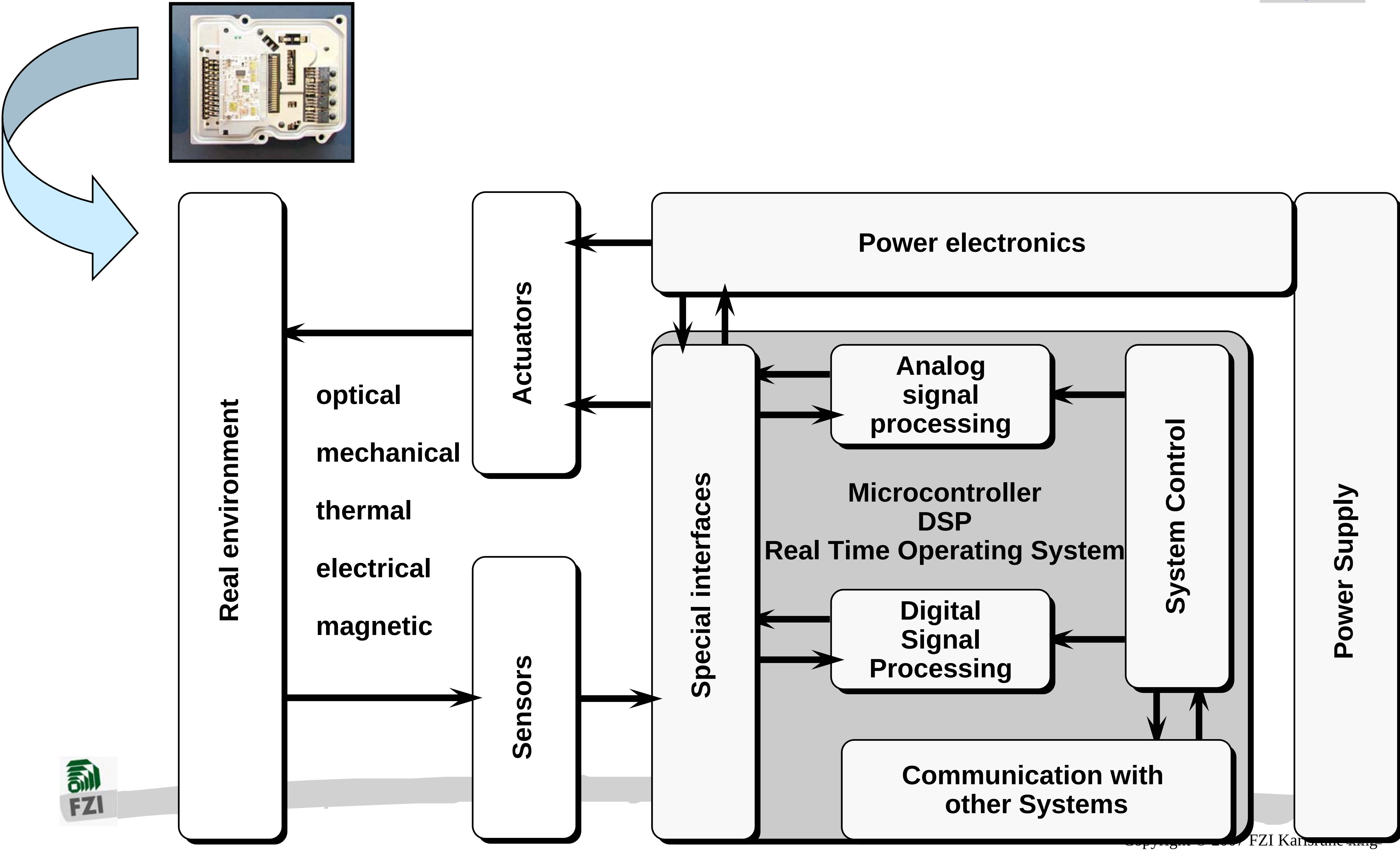
Characteristics: distributed, mechatronic, hard real time

Hard Real Time Constraints

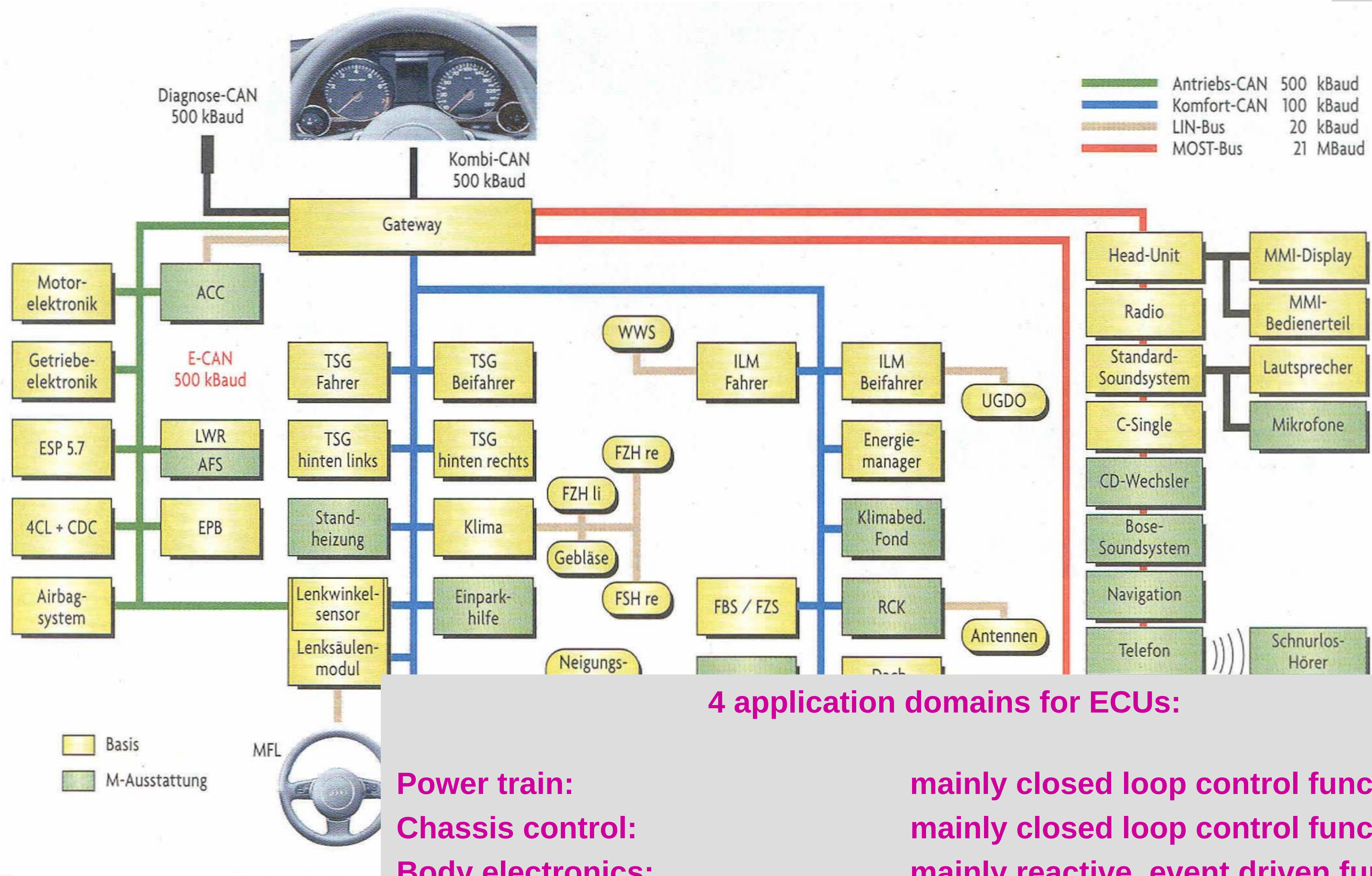
Example:
Airbag Control Unit



General structure of an ECU



Complex Communication (e.g. Audi A8)

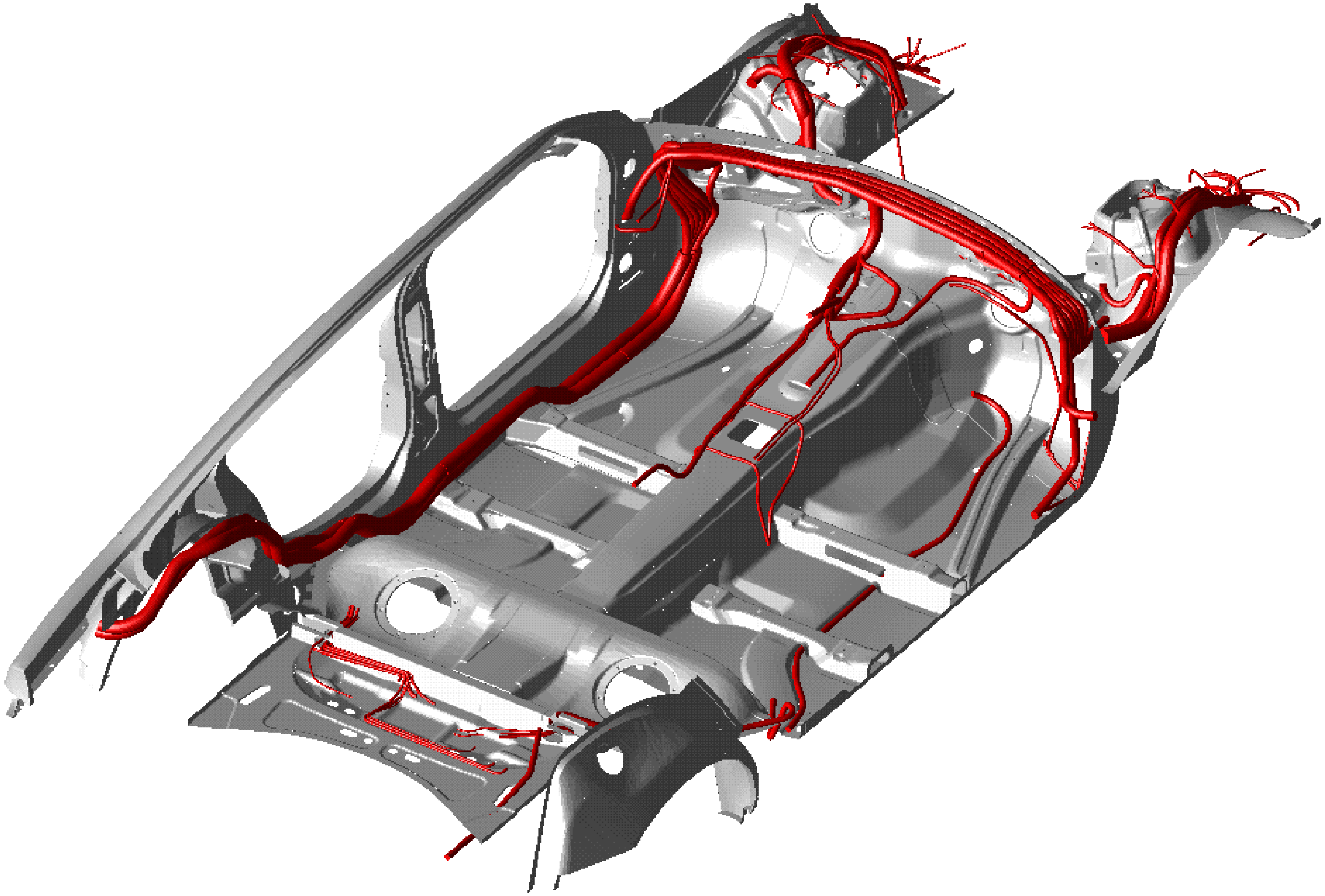


4 application domains for ECUs:

Power train:	**mainly closed loop control functions**
Chassis control:	**mainly closed loop control functions**
Body electronics:	**mainly reactive, event driven functions**
Infotainment:	**mainly reactive, event driven functions** **software intensive >>100k LOC**

Mechanics/Electrics-CoDesign (Digital Mockup - DMU)

The State-of-the-Art DMU technology provides the basis for the mechanical integration and optimization of EE components (ECU's, batteries, wiring harness, ...)



Embedded electronic systems in a car

Relatively high production volumes (5.000 – 1.000.000)

High number of variants (car families, countries, customers),

Reusability

tough operating conditions

- Temperature range: -40°C … +125°C … +175°C
- Supply voltage: 6V … 14V … 28V … (42V)
- Mechanical stress: acceleration, vibration
- Chemical stress: humidity, oil, exhaust gases, road salt …
- Electromagnetic compatibility

High reliability: << 1ppm/h Failure rate

Performance, Reliability, Safety, Security, Costs, Weight, 3D shape and volume

Energy Consumption (5% of fuel for EE-Systems)

Diagnosis and Maintainability (Service, Updates, Lifelong-Guaranty)

Long term availability: > 15 years

More than 30% of production costs for a passenger car is for electric/electronic systems (up to 40% by 2010)

90 % of all innovations are based on electronic systems

Software part is increasing rapidly

Automotive ECU: complex Design Process

Complex, distributed mechatronic system
with hard real time constraints

Design process shared between car manufacturer (OEM)
and several tier 1 suppliers

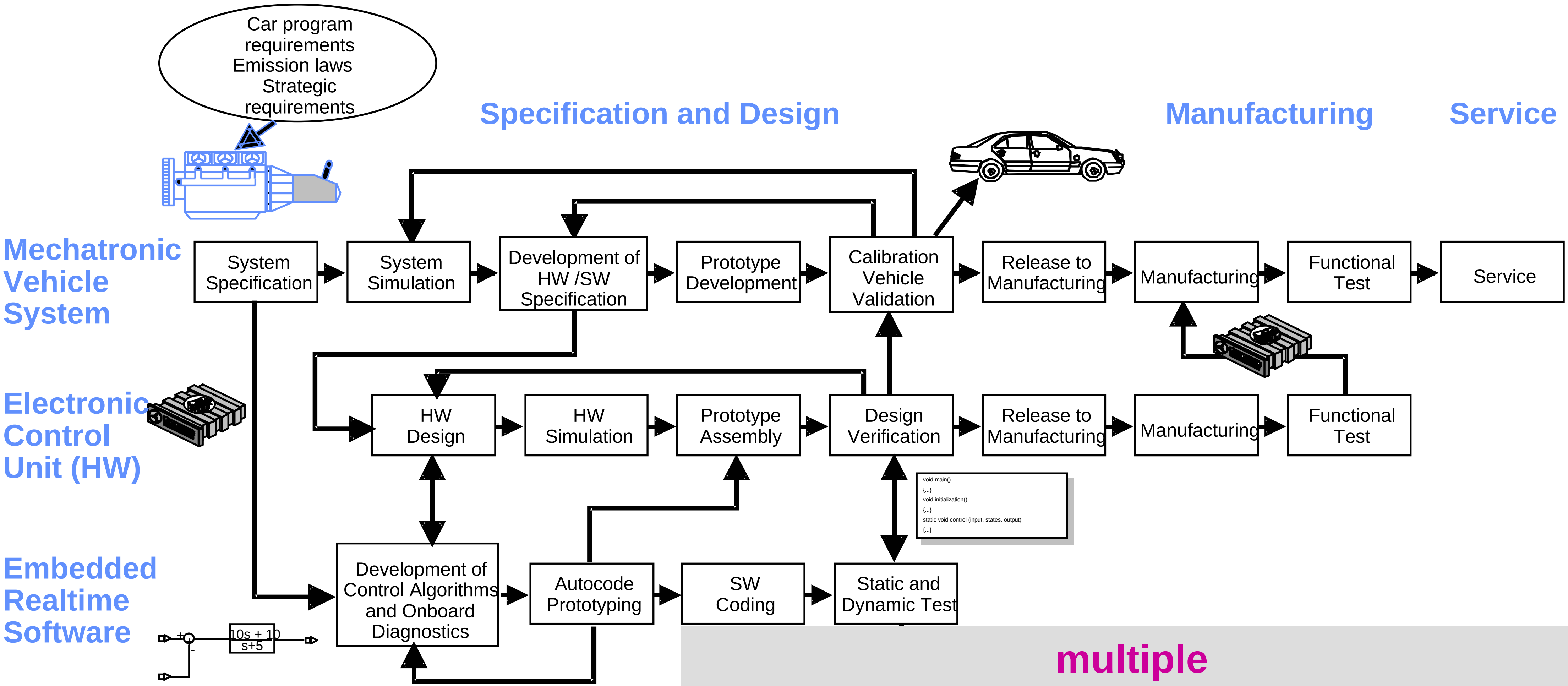
OEM defines features, sets up requirement specification

Supplier refines requirements specification, designs and delivers
optimized and verified subsystem
(complete mechatronic system including sensors, actuators,
ECU hardware and software)

OEM tests subsystem, integrates with other subsystems
and verifies and validates overall system

Complex design process

Hierarchical Organization of Design Processes



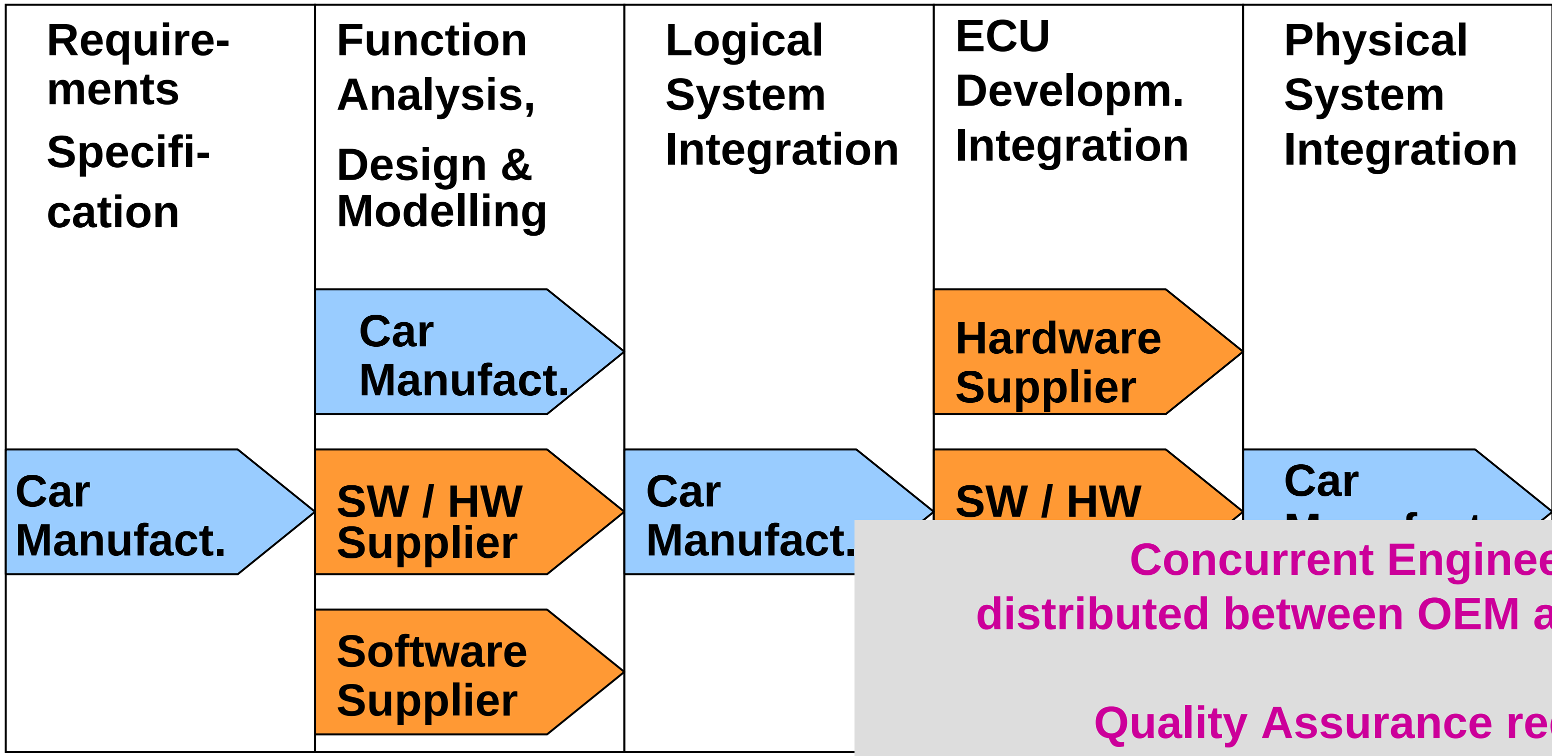
multiple
interleaving design processes

Concurrent Engineering
distributed between OEM and Tier 1 Suppliers

Complex Manufacturer Supplier Relationship

Car manufacturer controls system design and system integration

different „Business-Models“ for software and hardware development by tier 1 suppliers



Concurrent Engineering distributed between OEM and supplier

Quality Assurance requires comprehensive life cycle model (V-Model) strictly controlled design methodology supporting computer aided design tools

FZI

V-Model

Development Standard for IT-Systems of the Federal Republic of Germany

Lifecycle Process Model

http://www.v-modell.iabg.de

V Model: 4 sub models

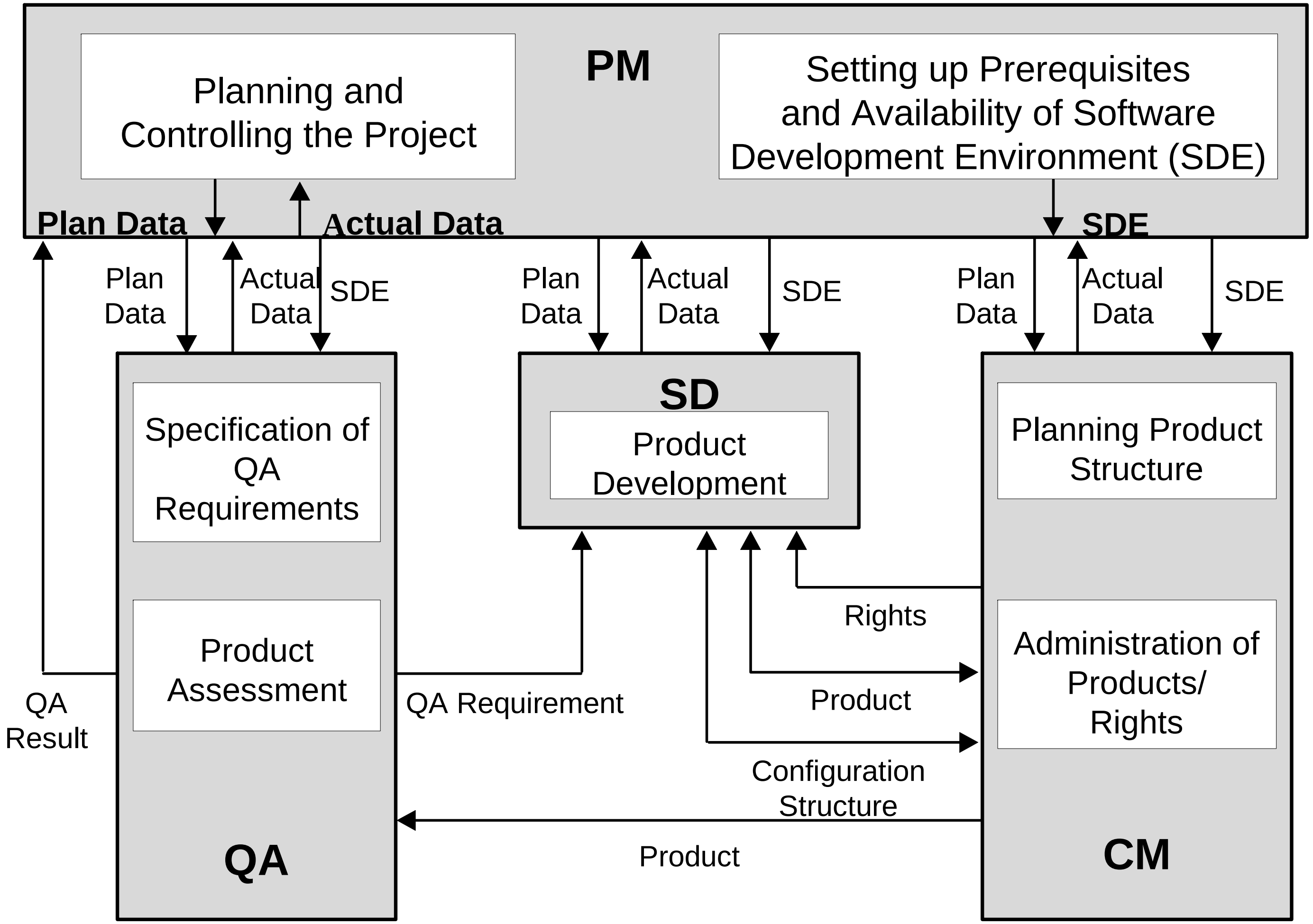
Four sub models are closely linked to one another and influence each other concerning the exchange of products/results.

PM plans, controls and informs the SD, QA and CM sub models.

SD develops the system or the software.

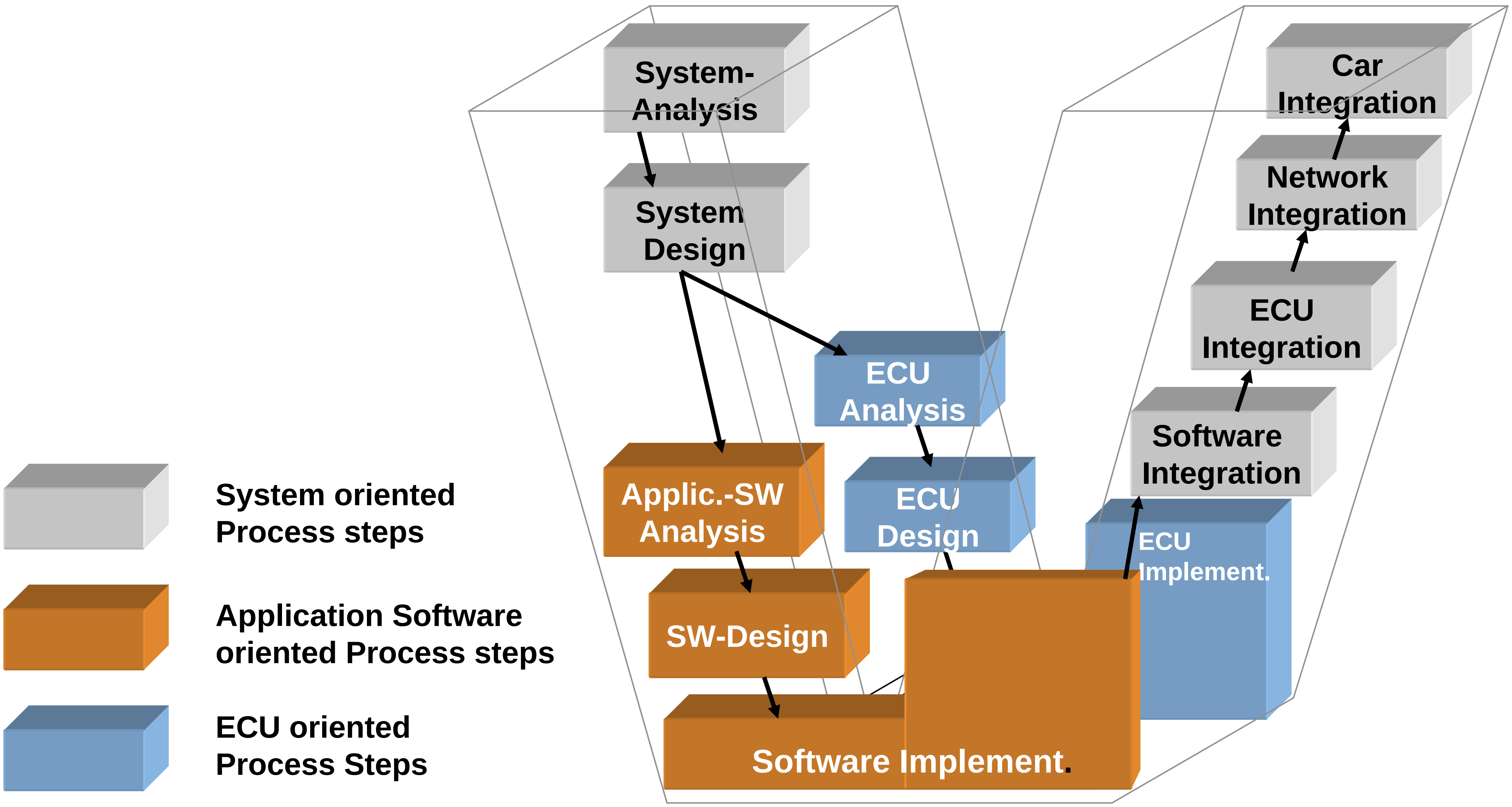
QA specifies quality requirements, test cases and criteria, and examines the products and the compliance with the standards

CM administrates the products generated



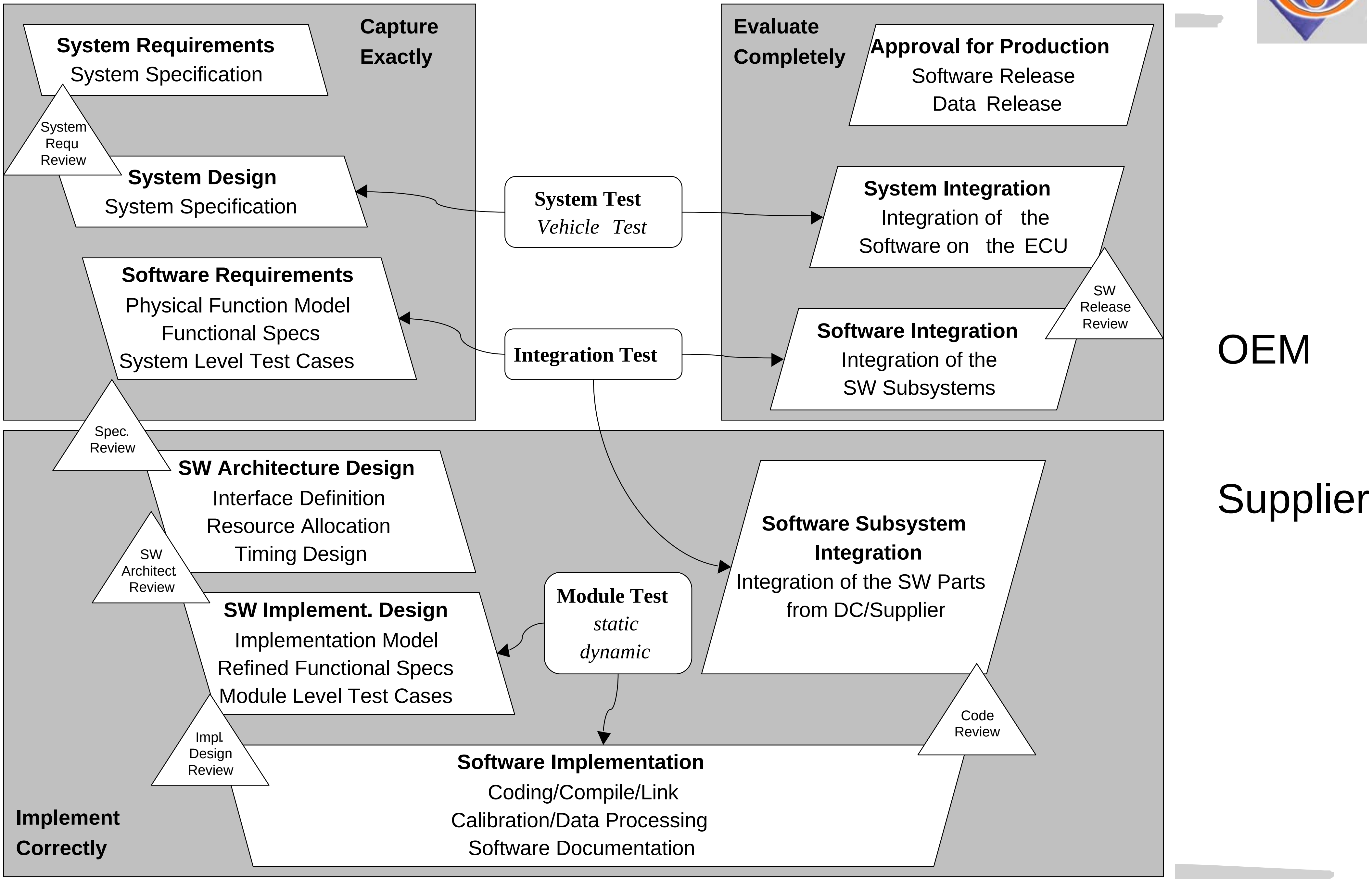
Cooperation of Submodels

V-Model for automotive ECU's



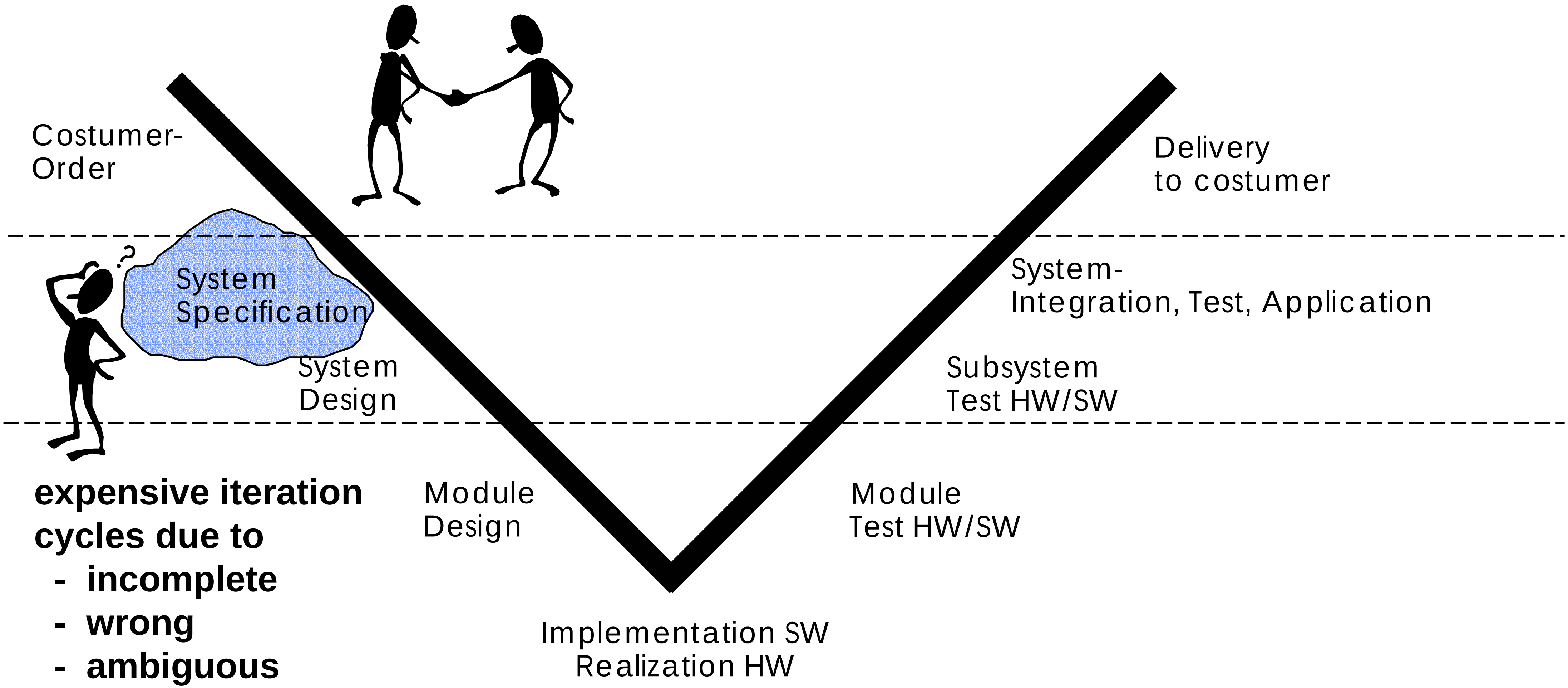
Courtesy ETAS

ECU Software Development



Automotive V-Modell accord. to Bortolazzi (DaimlerChrysler)

System specification as basis for cooperative design process



expensive iteration cycles due to
- **incomplete**
- **wrong**
- **ambiguous**
- **inconsistent**

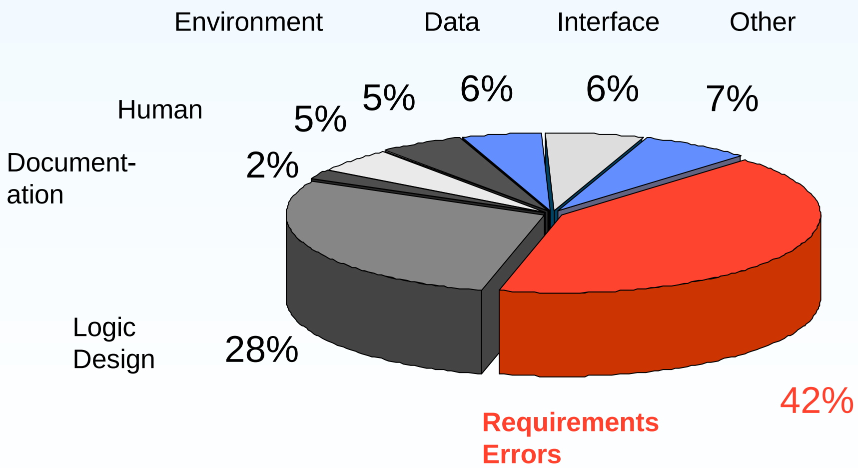
system specification

Formal Specifications, executable,
Model Based Design

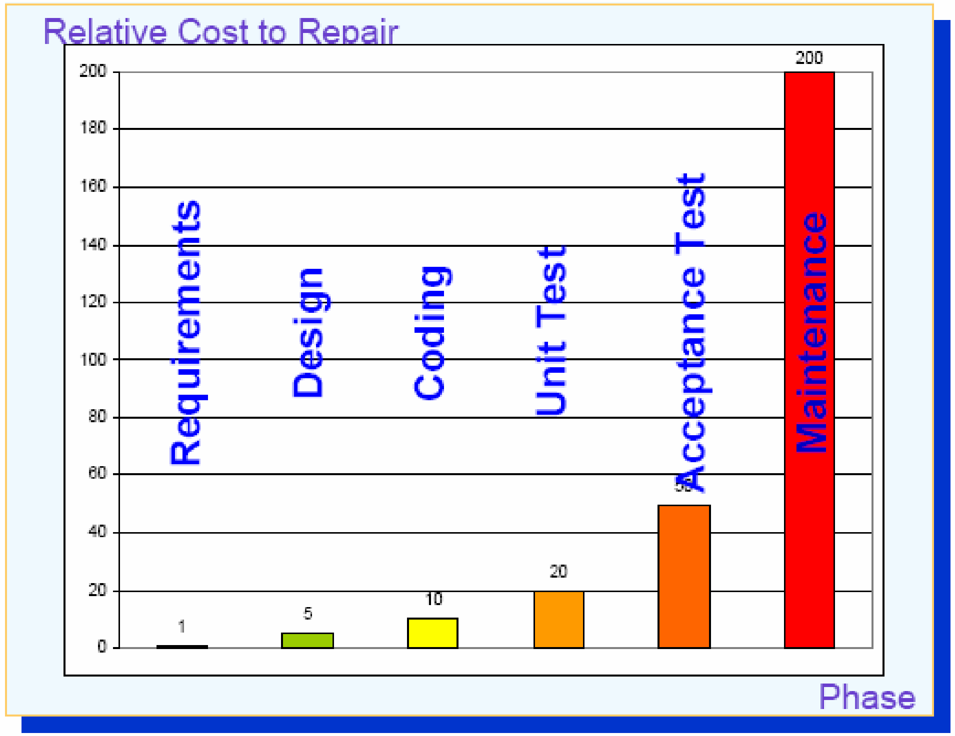
prime error source: requirements specification

More than 40% of system faults originate from errors during requirements analysis and management , costly when late repair...

Percentage of errors classified by problem type on a large IT project



Sheldon, et al
IEEE Software, July 1992

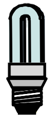


Relative cost to repair a defect at different lifecycle phases

OEM Supplier OEM

Typical Design Flow

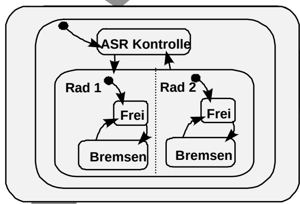
Idea



System-Analysis
executable Specs

System Design
model based

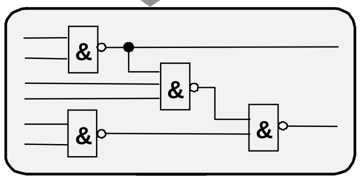
Customer Requirements
Technical Requirements
Real Time Requirements
System Architecture
Simulation, Verification



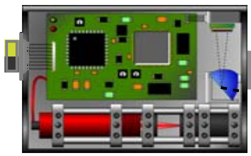
HW/SW-Requirements Analysis
Preliminary HW/SW-Design
HW-Architecture, SW-Architecture
Interface Description

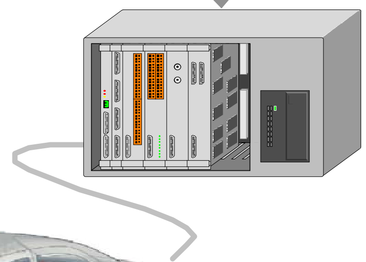
```
PROCESS (schlupf, state)
BEGIN
   CASE state IS
      WHEN freilauf =>
         IF schlupf > 0 THEN
            next_state <=
bremsen;
         ELSE
```

Rapid Prototyping
Hardware Platform
Code Generation
Real Time Operating System
configurable Interfaces



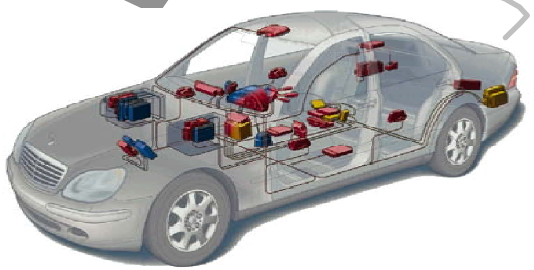
Detailed HW/SW-Design
SW-Design, Data Dictionary
HW-Drawings
HW-Analysis Report



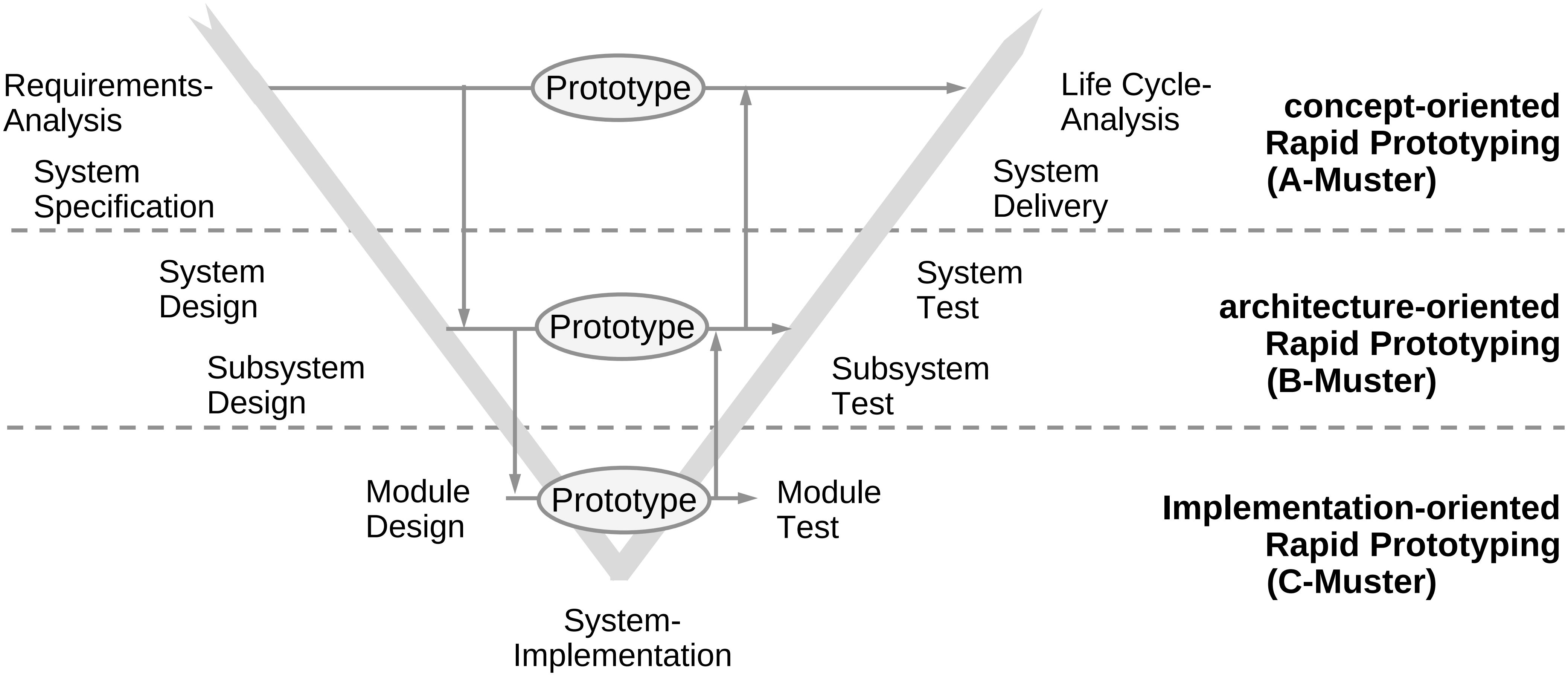


HW/SW-Implementation Integration
W-Modules, Data Dictionary, SW-Comp
HW-Component, HW-Module
HW-Realization Documents

System Integration
Calibration, Application
Transition to Utilization



ECU development for passenger cars: 3 Prototypes



Verification and Validation

Software Quality Control

Verification
Am I Building the Product Right?

Static Techniques

Review
Walkthrough, Fagan Inspection, Peer Review, Argument etc.

White Box Test
Static Analysis, Formal Proof, Control and Data Flow, etc.

Dynamic - Module/Integration Test

Black Box Test
Functional Performance, Stress Testing etc.

White Box Test
Structural, Path, Branch, Condition Decision Coverage etc.

Validation
Am I Building the Right Product?

Animation
Formal Specification, CASE Modeling, Rapid Prototyping, Virtual Reality etc.

System/Acceptance Test
Functional Performance, Stress Testing etc.

Design Challenges

"Smart Systems" - Engineering

- complex, distributed, heterogeneous, HW and SW
- technology road maps spectacular
  however, design gap gets larger

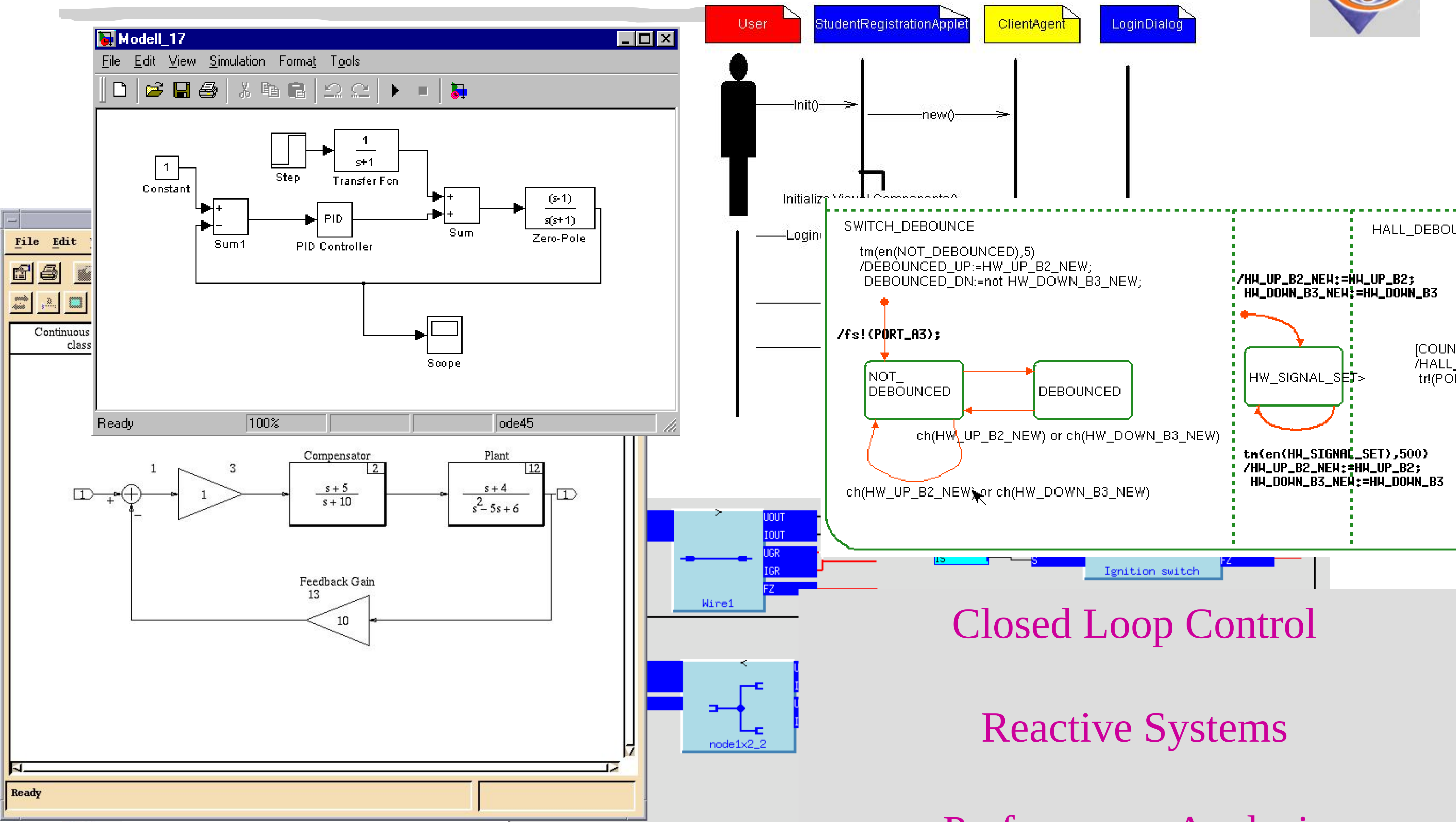
Smart - "Systems Engineering"

- design methodology
- early system design phases most important
- model based design, executable specification
- system level modeling and simulation
- rapid prototyping, hardware in the loop
- productivity (reuse, automatic code generation)
  enhanced design quality

promising approach

Model Based Design

Model Based Design - graphical descriptions preferred



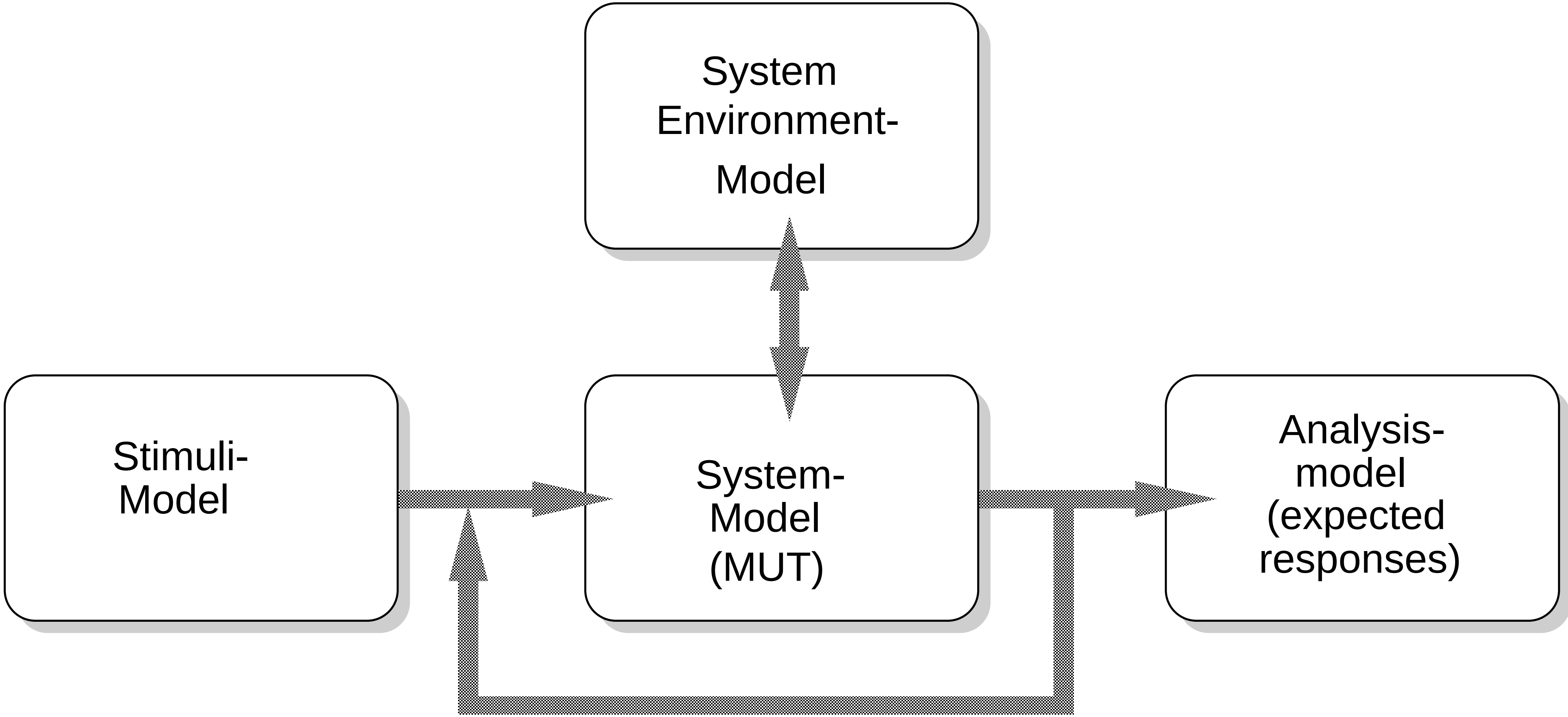
Closed Loop Control

Reactive Systems

Performance Analysis

Model Based Design:

Models for Executable Specification and Analysis (Simulation)



Modeling

Modeling for complete system including system environment
(ECU, car, driver, road, weather conditions)

Domain specific models for Subsystems and Components
(closed loop control, reactive systems, software intensive systems)

Different abstraction levels, Parameter variation and boundaries
(functional and non-functional data for early design space exploration)

Use of characterized libraries (reuse, variant design)

Model verification through extensive testing

Model characterization

Model documentation

Macro modeling

Meta modeling

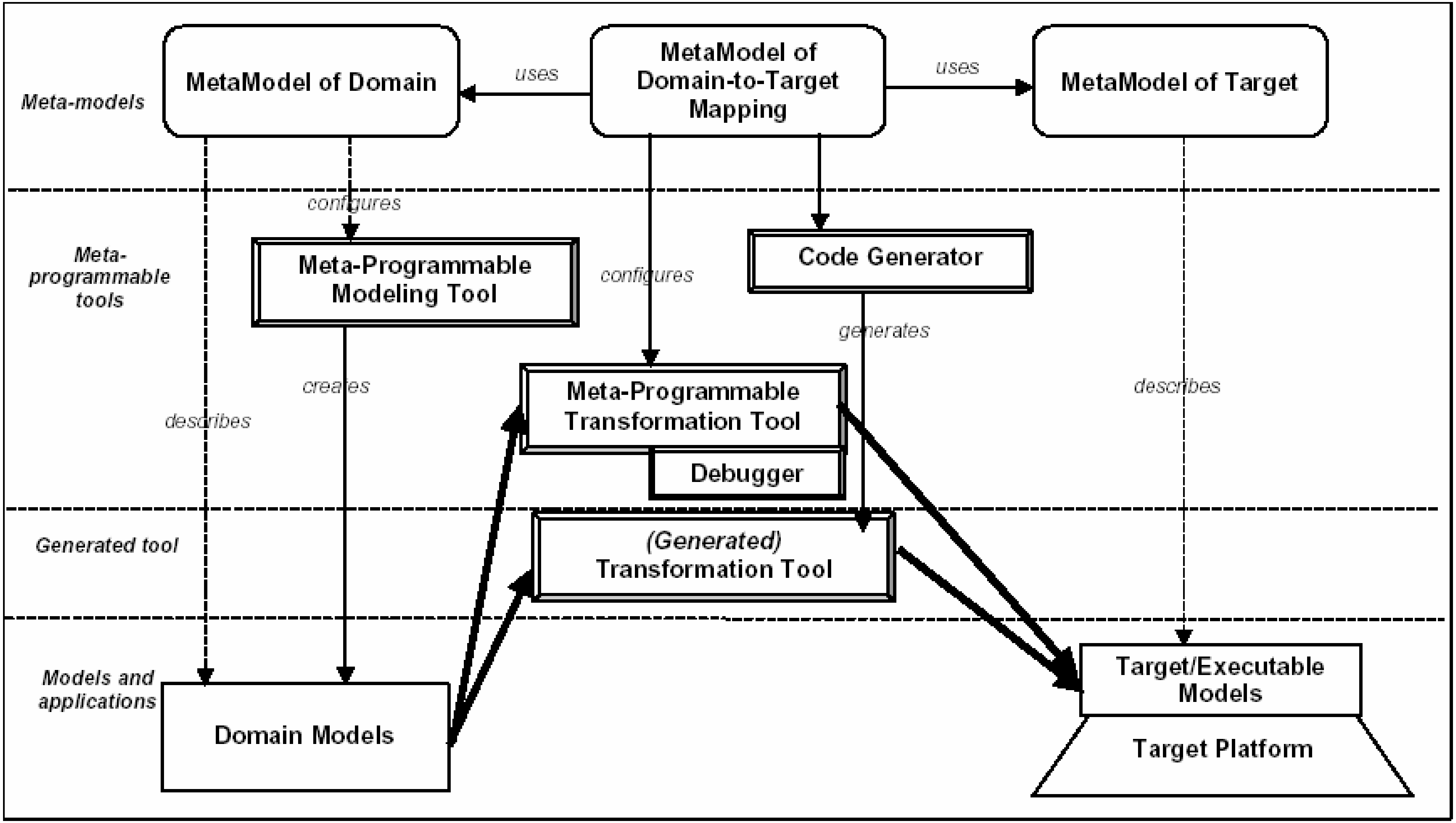
Challenges for Modeling

Domain Specific Modeling Languages
Model Synthesis
Model Validation
Model Transformation
Executable Models
Automatic Generation of Product Artefacts

Meta-Modeling
Tools on Meta-Model-Level
Integration of Domain Specific Tools
on Meta-Level

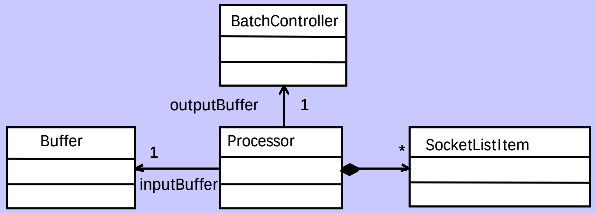
Generic Modeling-Platforms

Meta-model based design (Sztipanovits, Karsai: Vanderbilt University)

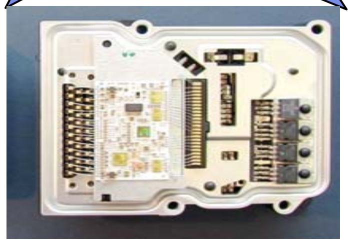


Modeling for heterogeneous electronic embedded systems

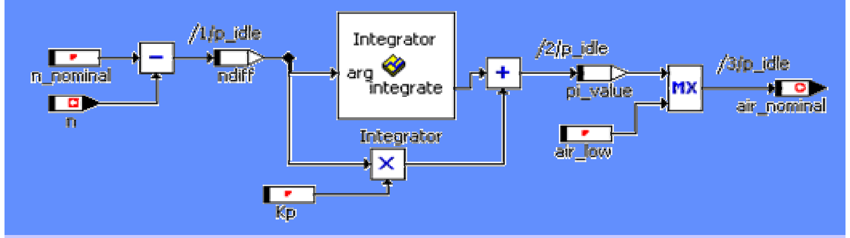
Architecture
Modelling with UML



Real-time Studio (ARTiSAN)
Rhapsody in C++ (i-Logix)
Rose (Rational Software, IBM)
Together (Borland)
Poseidon (Gentleware)
MagicDraw (NoMagic)
Ameos (Aonix)
TAU2 (Telelogic)

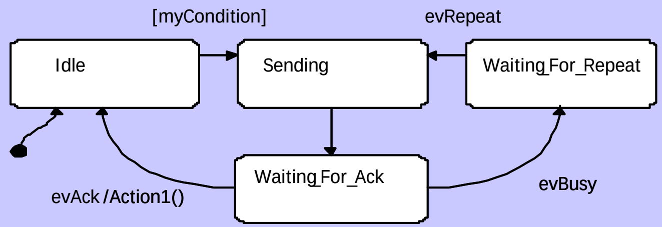


Signal flow oriented
Modelling with block diagrams



ASCET (ETAS)
MATLAB/Simulink (The MathWorks)
MATRIXx (National Instruments)

Event driven
Modelling with state charts

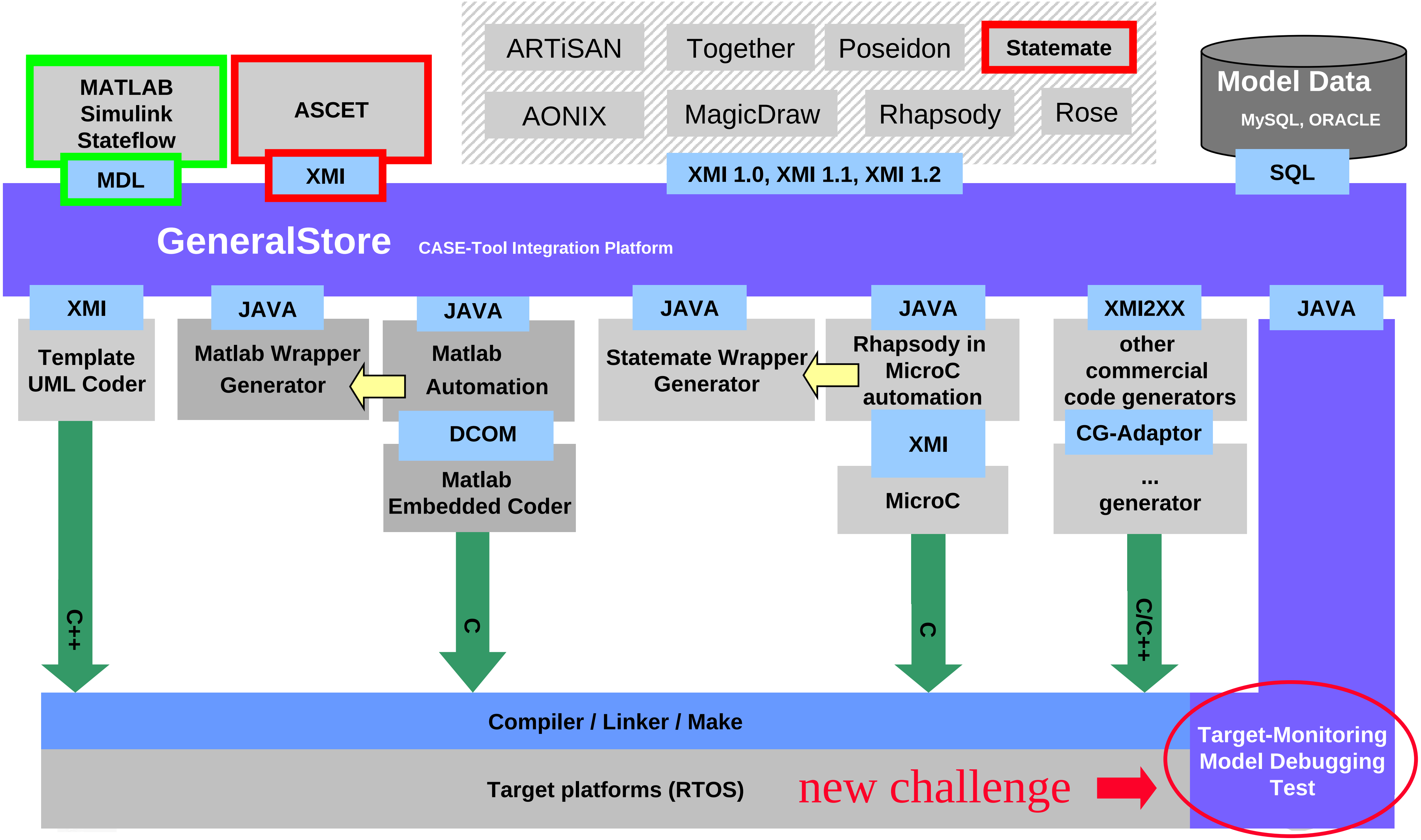


Rhapsody in C++ (i-Logix)
Statemate (i-Logix)
Stateflow (The MathWorks)
ASCET (ETAS)

Heterogeneous modeling requires integration platform

e.g. ETAS Integrio, Vector DaVinci

ITIV/FZI Tool integration platform (model transformation)



Meta-Modeling 4 Abstraction Layers (OMG standard)

abstract

real

M3 layer

MOF

M2 layer

UML 1.5

UML 2.0

MATLAB Simulink

Statechart (D. Harel)

M1 layer

UML Model

Simulink Model

Statechart Model

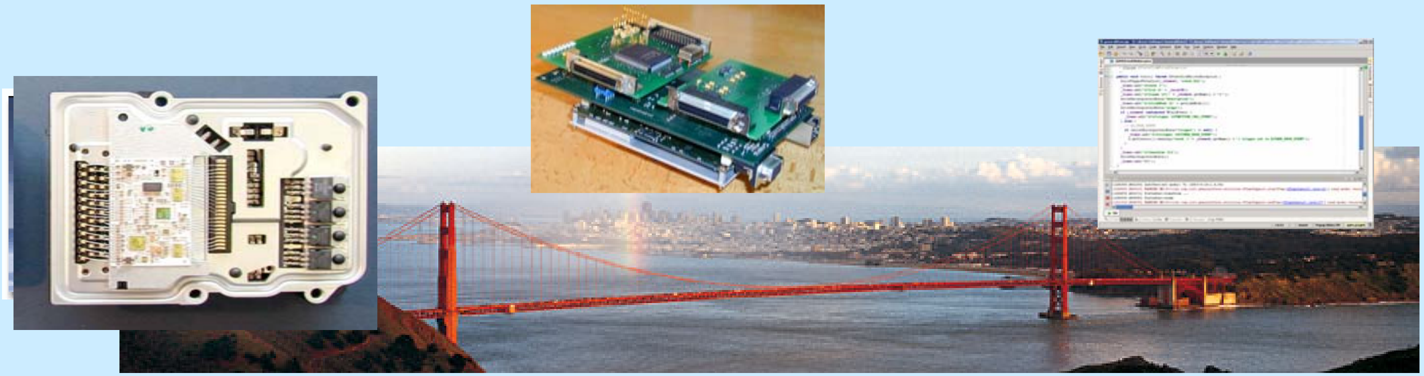
M0 layer

Objects

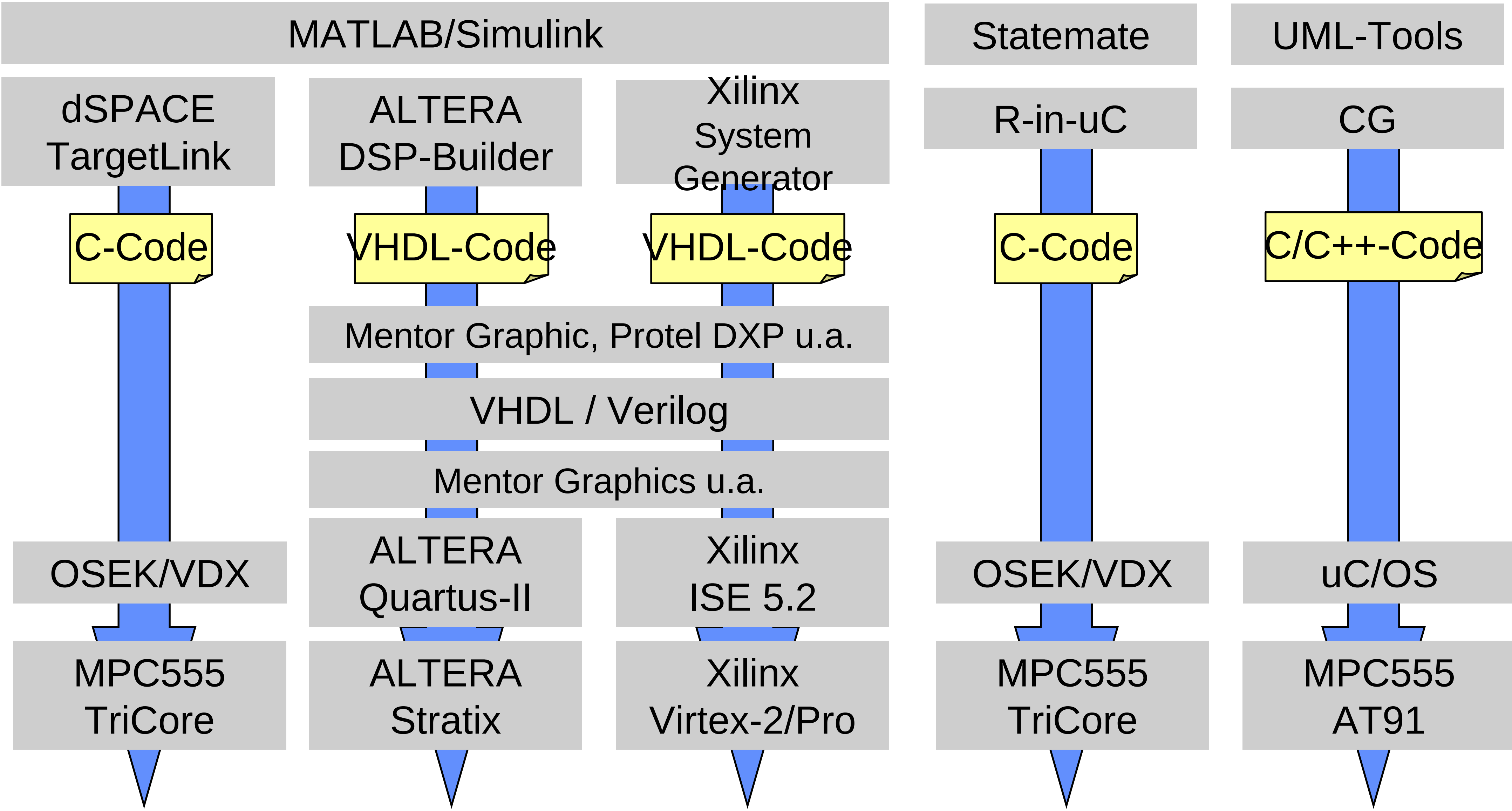
Data

Source code

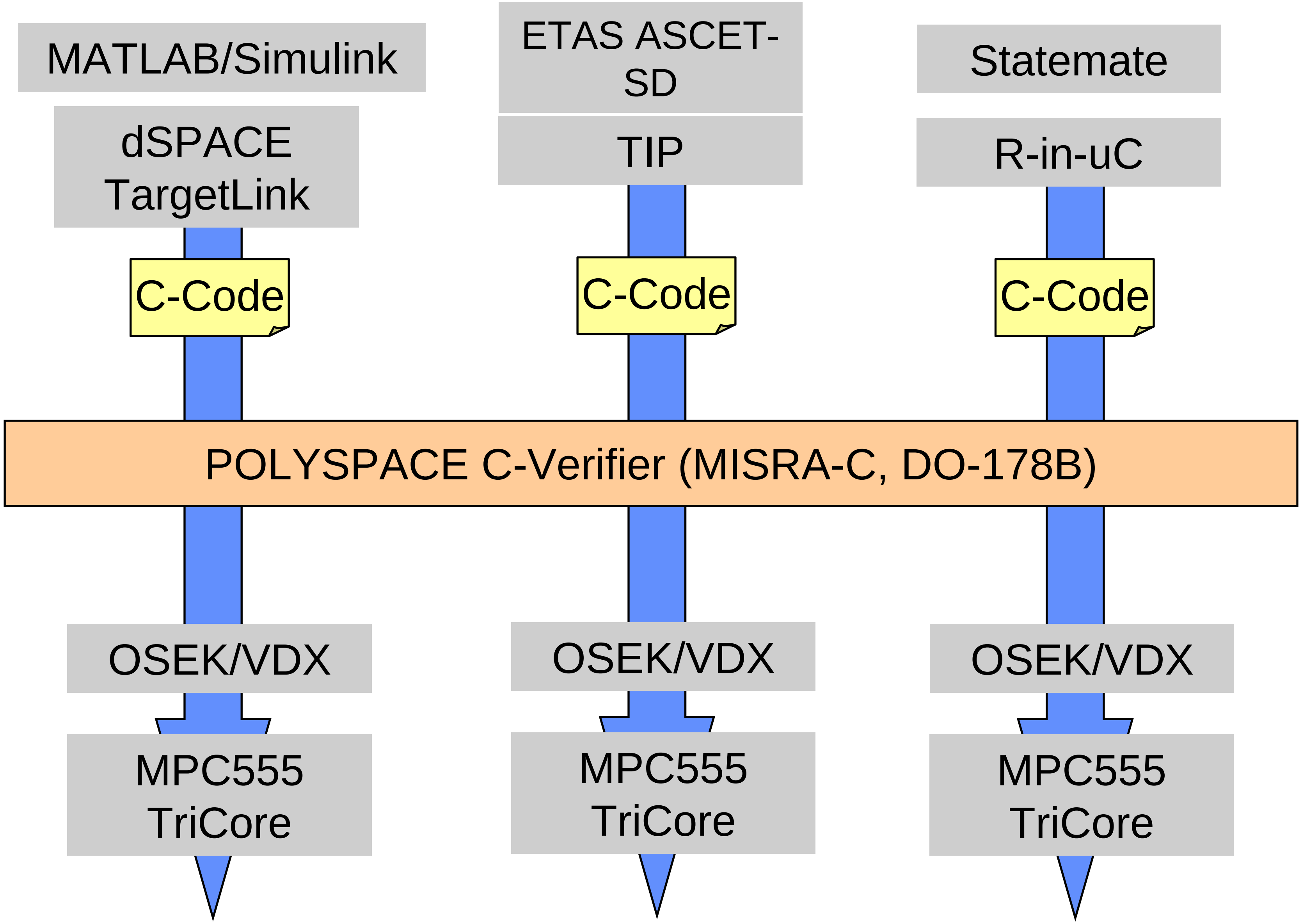
Real artefacts



Tools Chains used at ITIV/FZI



ITIV/FZI Tool Chains (Automotive) Verification Support



Tools used for ECU design

specification support	(Doors, QFD/Capture)
reactive systems	(SDL, Stateflow, Statemate)
closed loop control systems	(ASCET-SD, Matlab/Simulink, MatrixX)
software systems	(Real-time Studio, Rhapsody in C++, Rose, Together, Poseidon, MagicDraw, Ameos TAU2)
performance analysis	(SES/Workbench, Foresight)
tolerance analysis	(Rodon)
rapid prototyping, HiL	(dSPACE, ETAS, IPG, Quickturn)
application, test, diagnosis	(ETAS, Hitex, Vector, RA)
C-Verifier	(PolySpace)
ASIC Design	(Cadence, Mentor, Synopsys)

Design Challenges: Automotive ECU

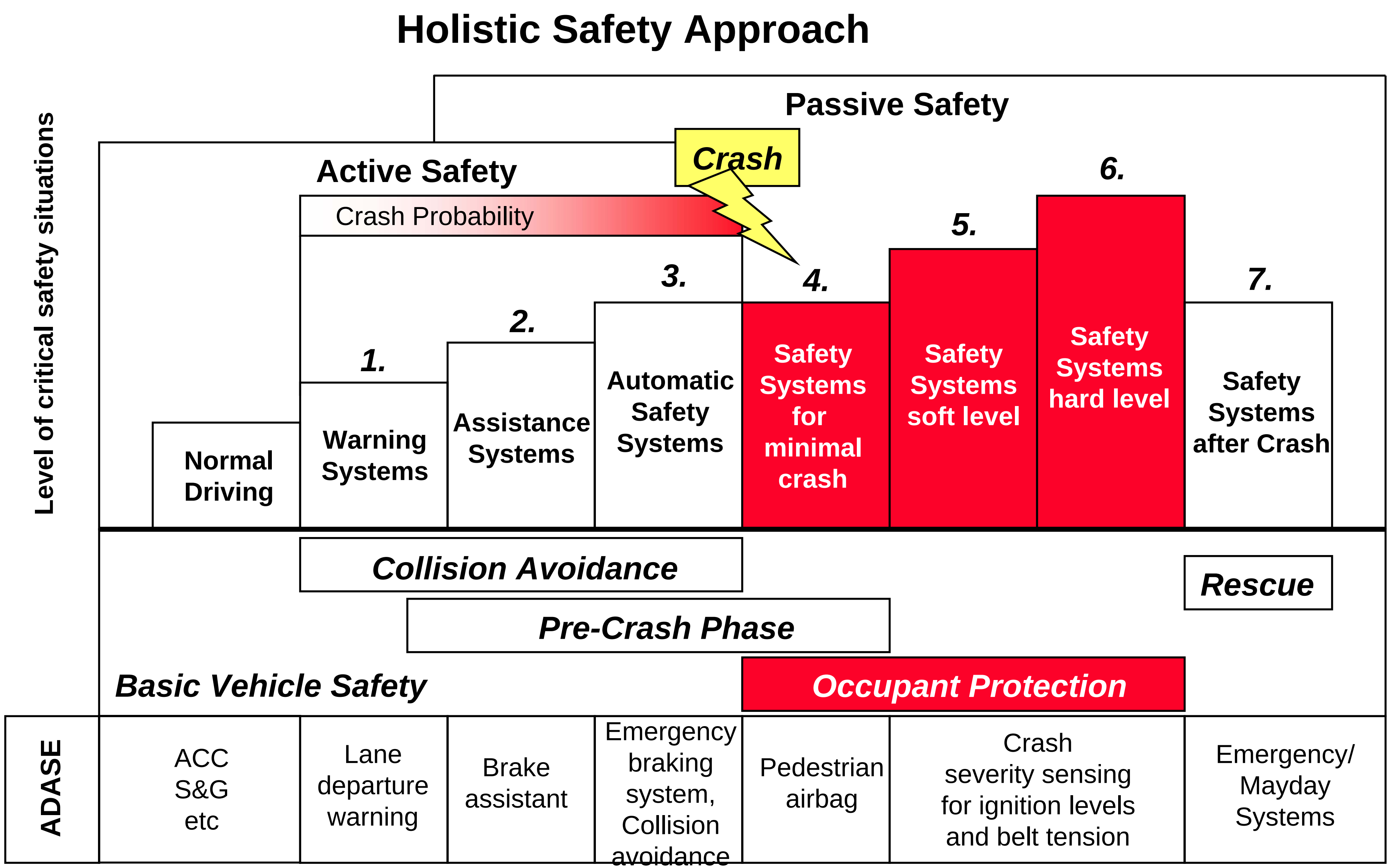
- Complex, distributed mechatronic system with hard real time constraints
- Design process shared between car manufacturer (OEM) and several tier 1 suppliers
- OEM defines features, creates specification model
- Supplier develops specification model into implementation model, does analysis and design, verification and validation, builds and tests, finally delivers optimized subsystem to OEM (Sensor, Actuator, ECU hardware and software)

However,

- Still increasing complexity (more comfort and safety functions coming)

Challenge: new safety functions

EUCAR: Active Safety – System Integration

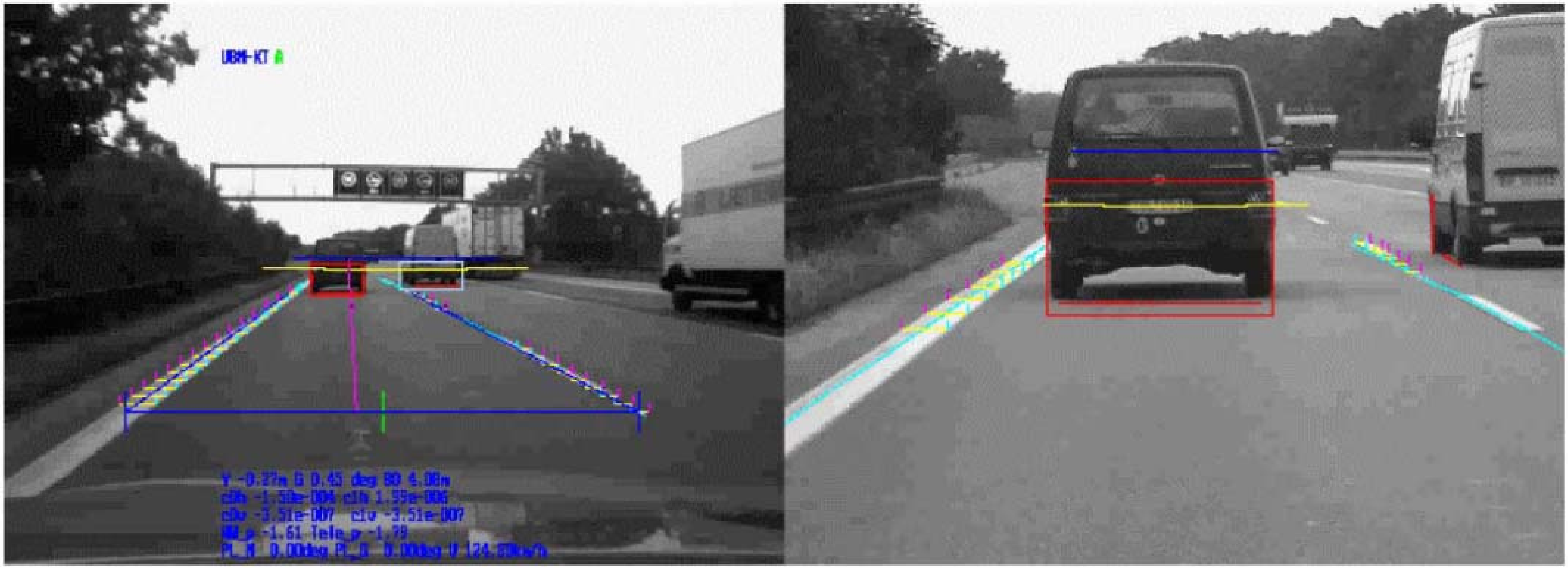


Future systems

Example active / passive safety:
Recognition of traffic signs
and traffic members (obstacle)

Active Safety: Next Generation Technology

Variety of mechanical, radar, video sensors
to provide optimum of crash avoidance, crash detection



Plus future car2car, road2car, TMC2car communication forming highly dynamic, reconfigurable sensor/actor networks

Distributed ECU's in cars - design challenges

Still increasing complexity (more comfort and safety functions coming)

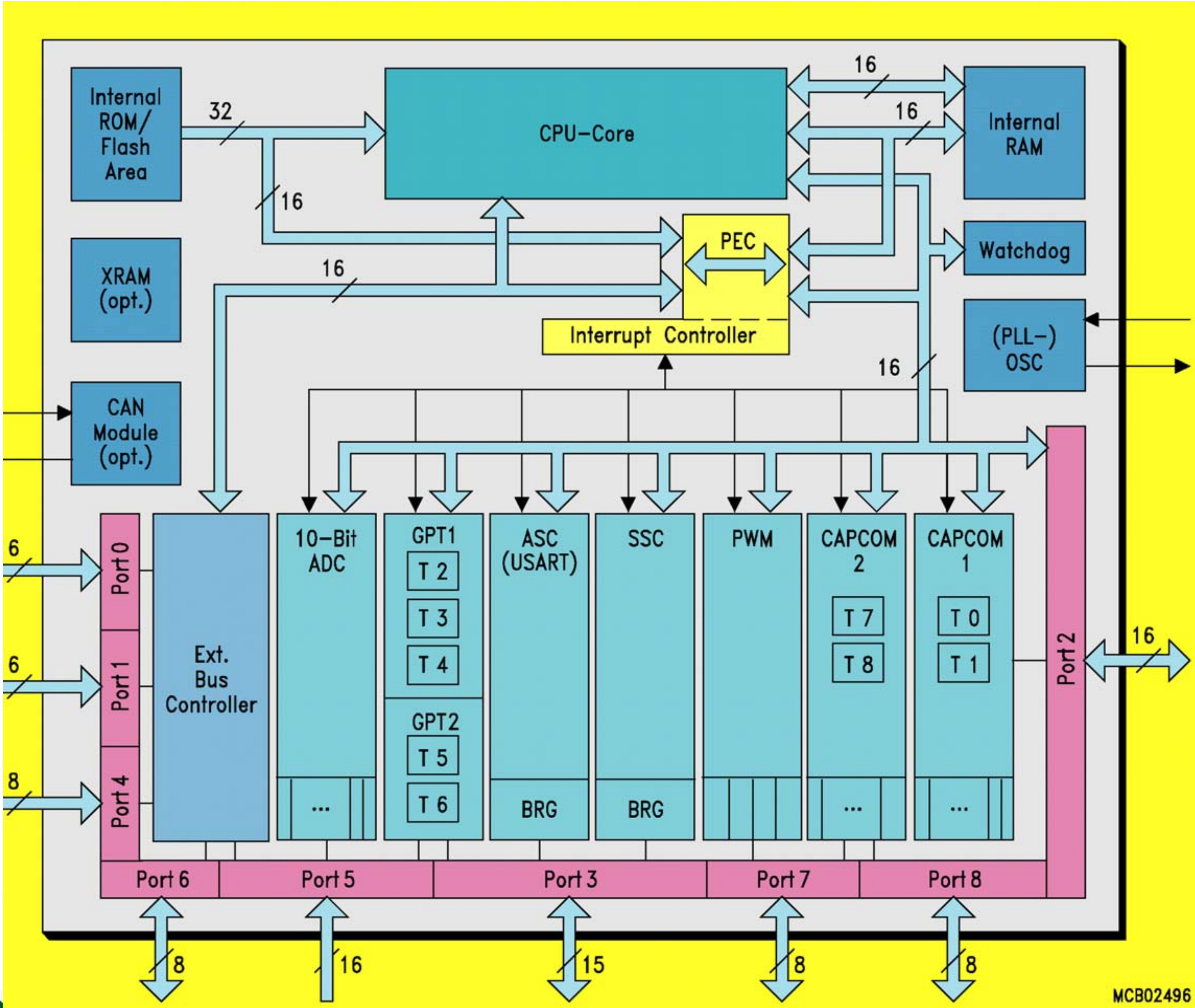
number of ECU's must not increase, should decrease!

less, but more powerful HW platforms (8, 16, 32-bit µC)

eventually new, more flexible architectures
(e.g. dynamically reconfigurable?!)

FZI

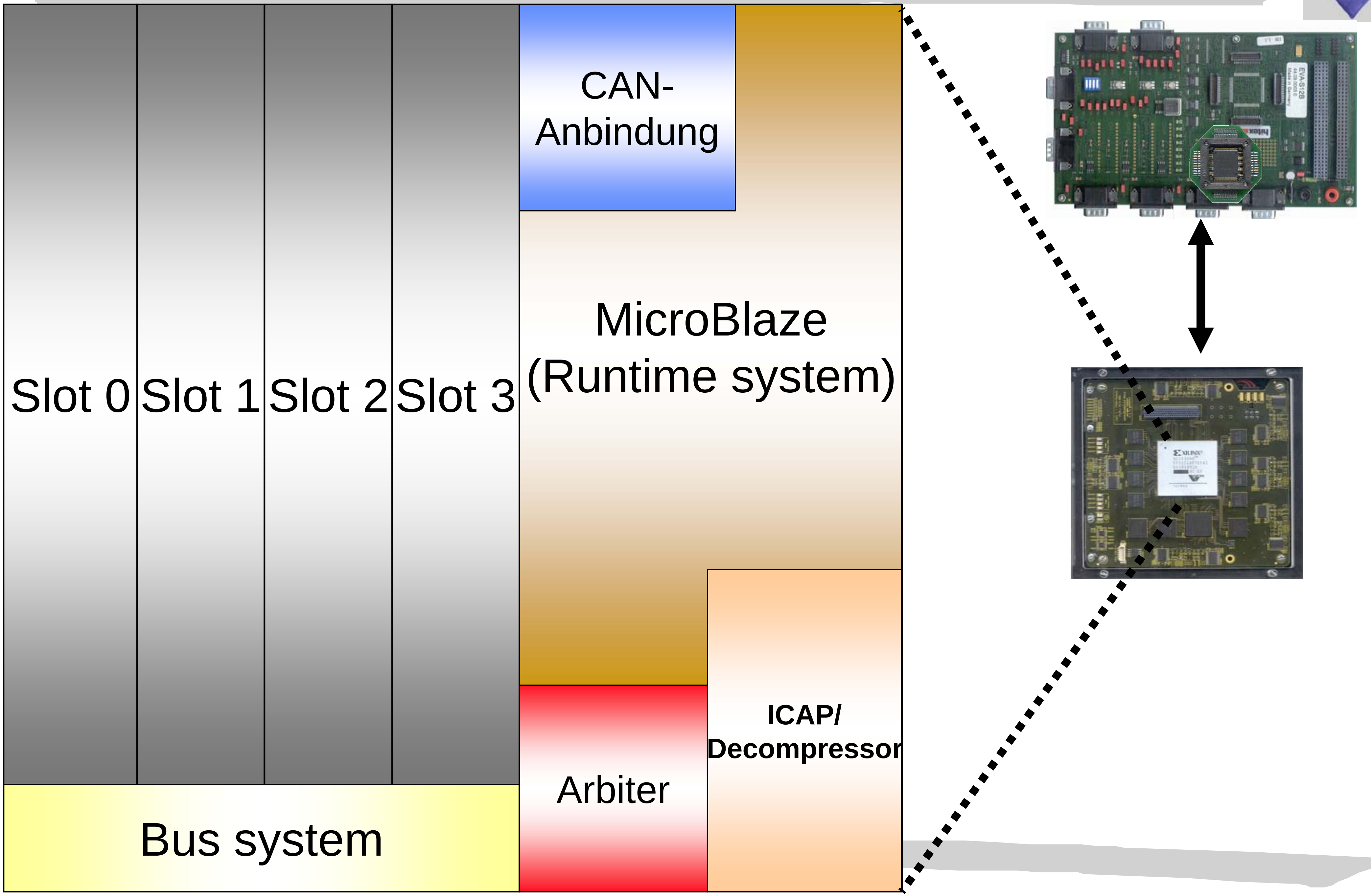
Typical automotive micro controller architecture



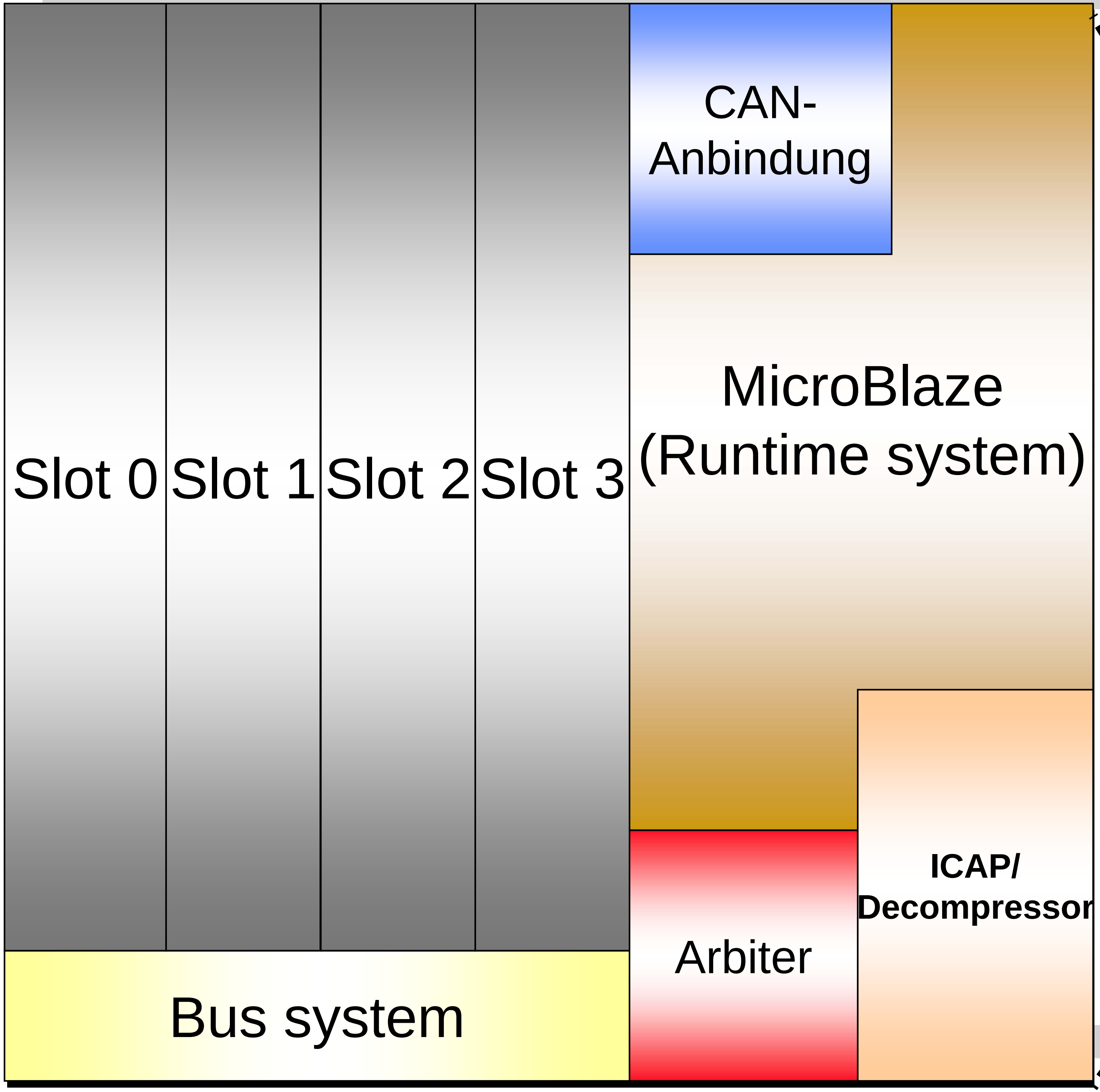
Running the standard operating system OSEK/VDX

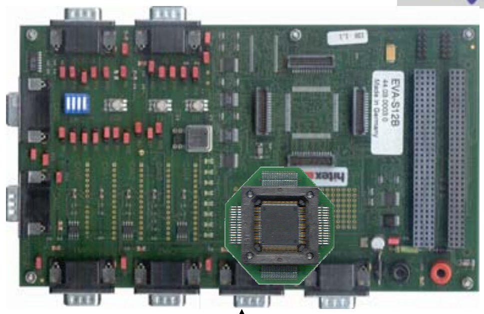
Motorola M683xx
Motorola HC08,12
Motorola MPC5xx
Infineon C16x
Infineon TriCore
Hitachi SH2
Hitachi H85/26xx
TI TMS47OR1
Mitsubishi El. M32R

Dynamic reconfiguration of a FPGA module slot



Dynamic reconfiguration of a FPGA module slot





Distributed ECU's in cars - design challenges

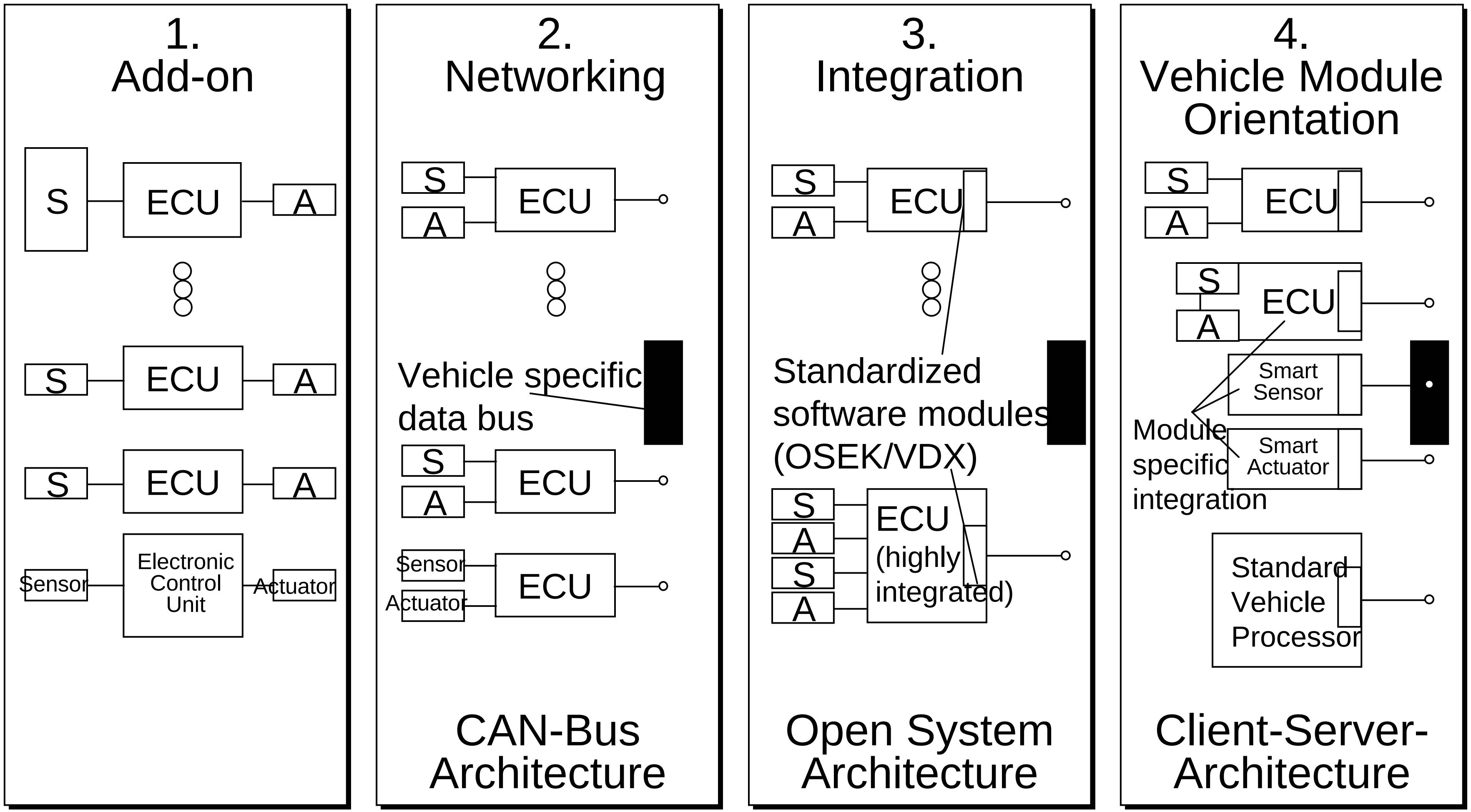
- Still increasing complexity (more comfort and safety functions coming)

- number of ECU's must not increase, should decrease!
  - less, but more powerful HW platforms (8, 16, 32-bit µC)
  - eventually new, more flexible architectures (e.g. dynamically reconfigurable?!)

- Given new hardware platforms requires redistribution (mapping) of software onto fewer hardware platforms
- Easy redistribution only possible with open system architecture (standardized communication, standardized RTOS)

FZI

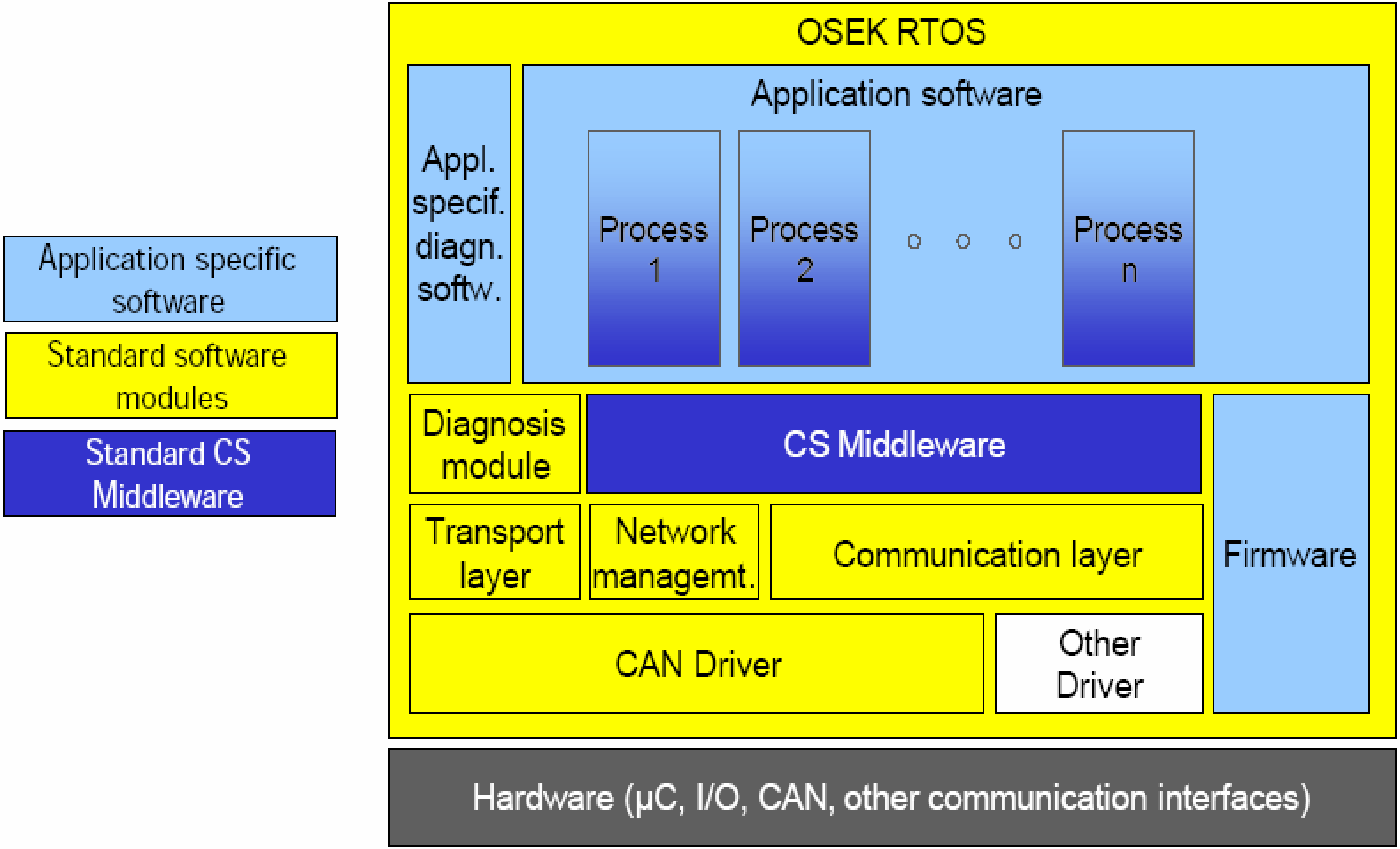
Evolution of hardware/software architectures in a car



Evolution led to open system architectures with modular software architecture:

Milestones: CAN, OSEK/VDX, (AUTOSAR)

Architecture Real Time Operating System OSEK



AUTOSAR

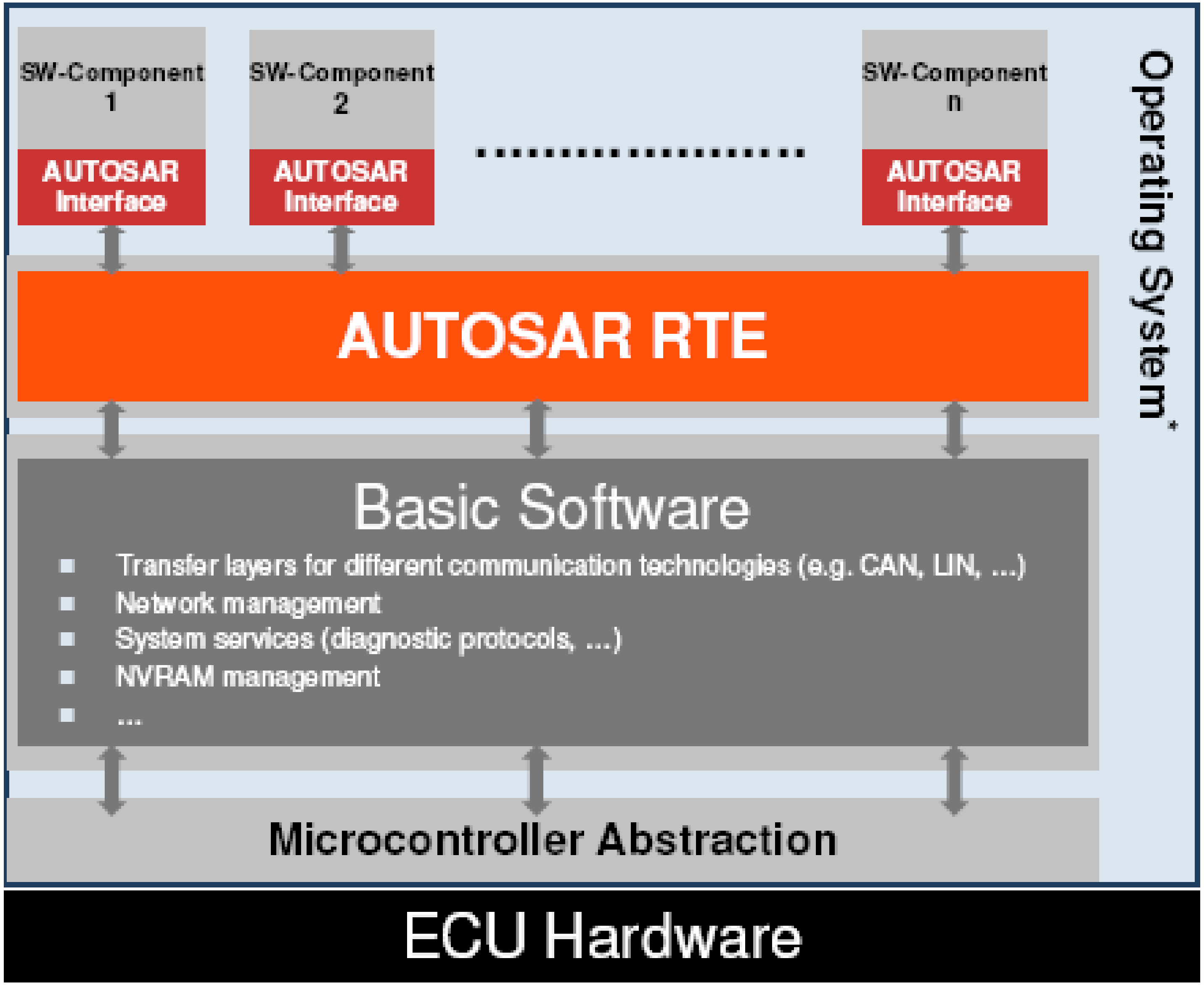
Automotive Open System Architecture (AUTOSAR):

- standardized and open interfaces
- HW– independent SW-comp.
- enables standard SW-function libraries

AUTOSAR RTE:

Specification of interfaces and communication mechanisms

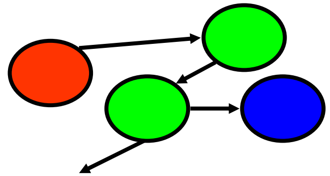
separate application programs from underlying ECU HW and Basic SW



* z. B. : OSEK, QNX, VxWorks, Windows CE, …

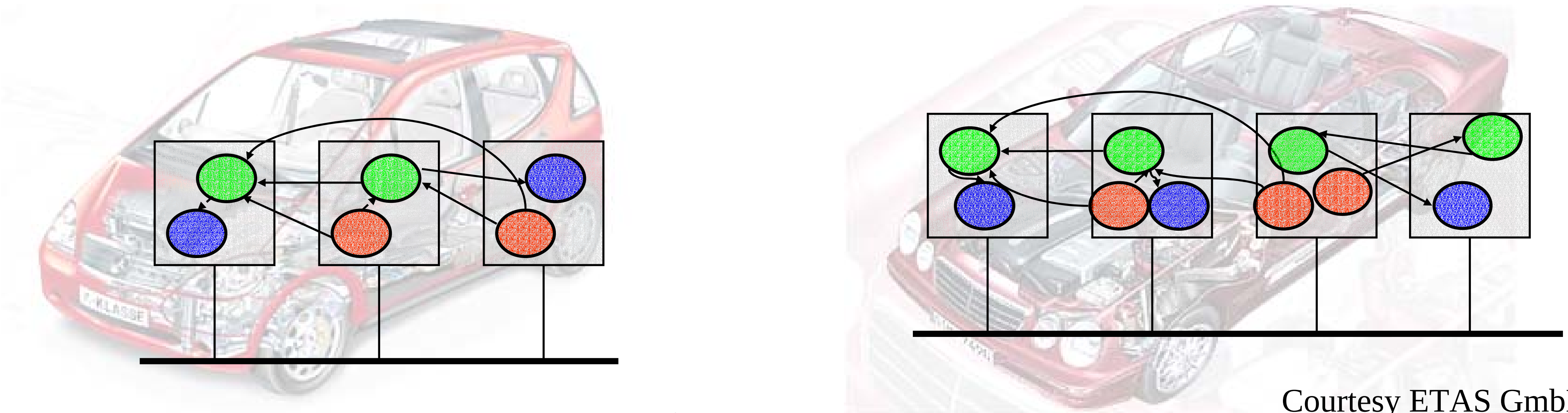
Desired

Reuse of Designs
Reuse and maximum usage of Hardware
Reuse of Software
Reuse of Validation and Verification



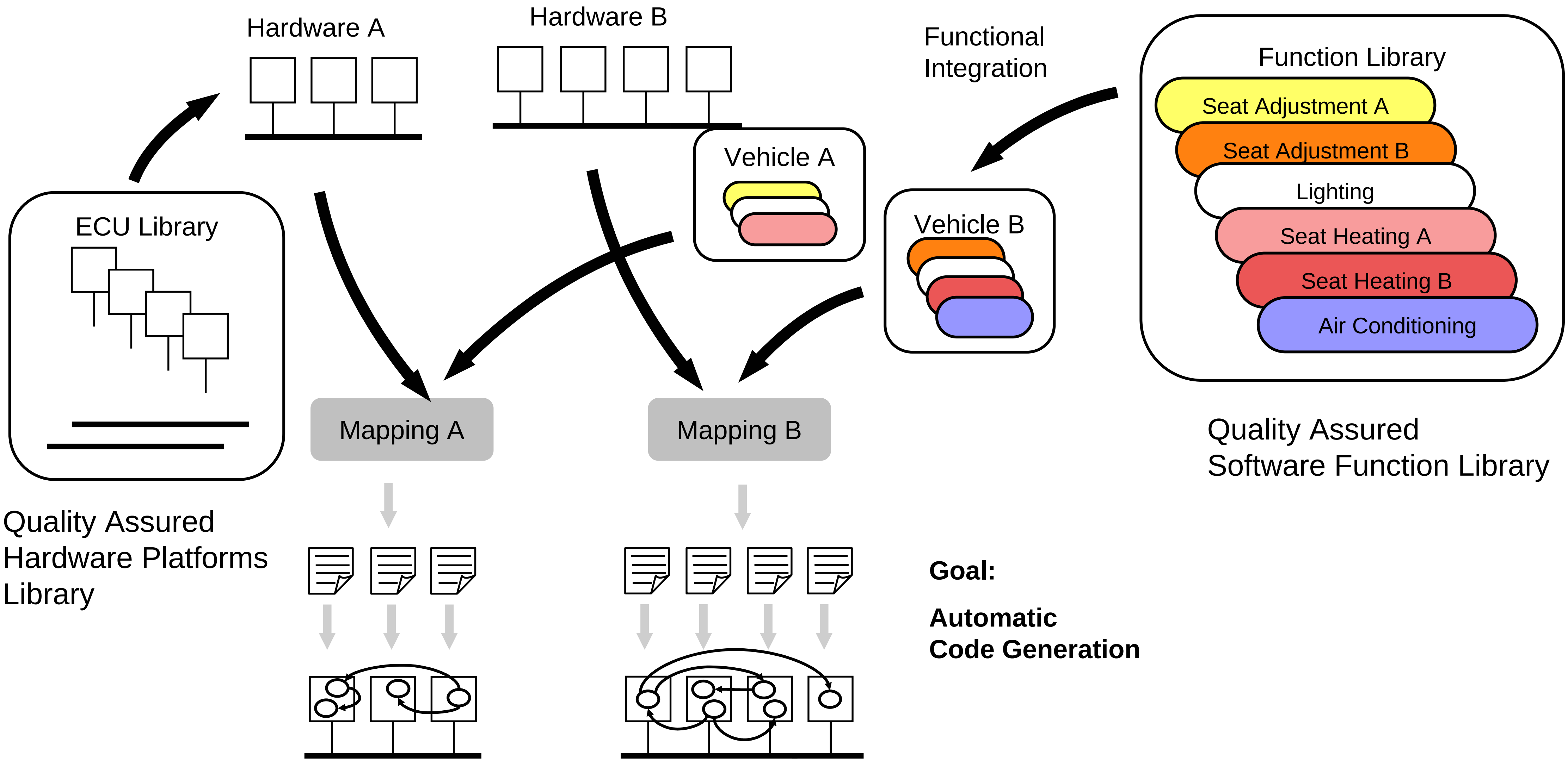
Additional Functions

Basic Functions



Courtesy ETAS GmbH

Goal (AUTOSAR)



Goal:

Automatic
Code Generation

Challenge

unlimited

Algorithm Integration

C, C++
Matlab
SDL, SPW
Cossap

Functional Network

Does the functionally integrated design work

Executable Functional Specification

mapping

Performance

Architecture Performance

CPU, DSP
Bus, *I/O*
Memory, HW
SW, RTOS

Unambiguous Structure

Are Partitioning & Performance Sufficient?

Executable Performance Specification

detailed design

Alberto Sangiovanni Vincentelli

Another Challenge: Upcoming X-by-Wire Systems

Delayed for 4 to 5 years

First driver assistance systems overruling driver currently being introduced (truck emergency brake system)

EN 61508 norm for safety critical electronic control systems not yet finally adapted for car industry.

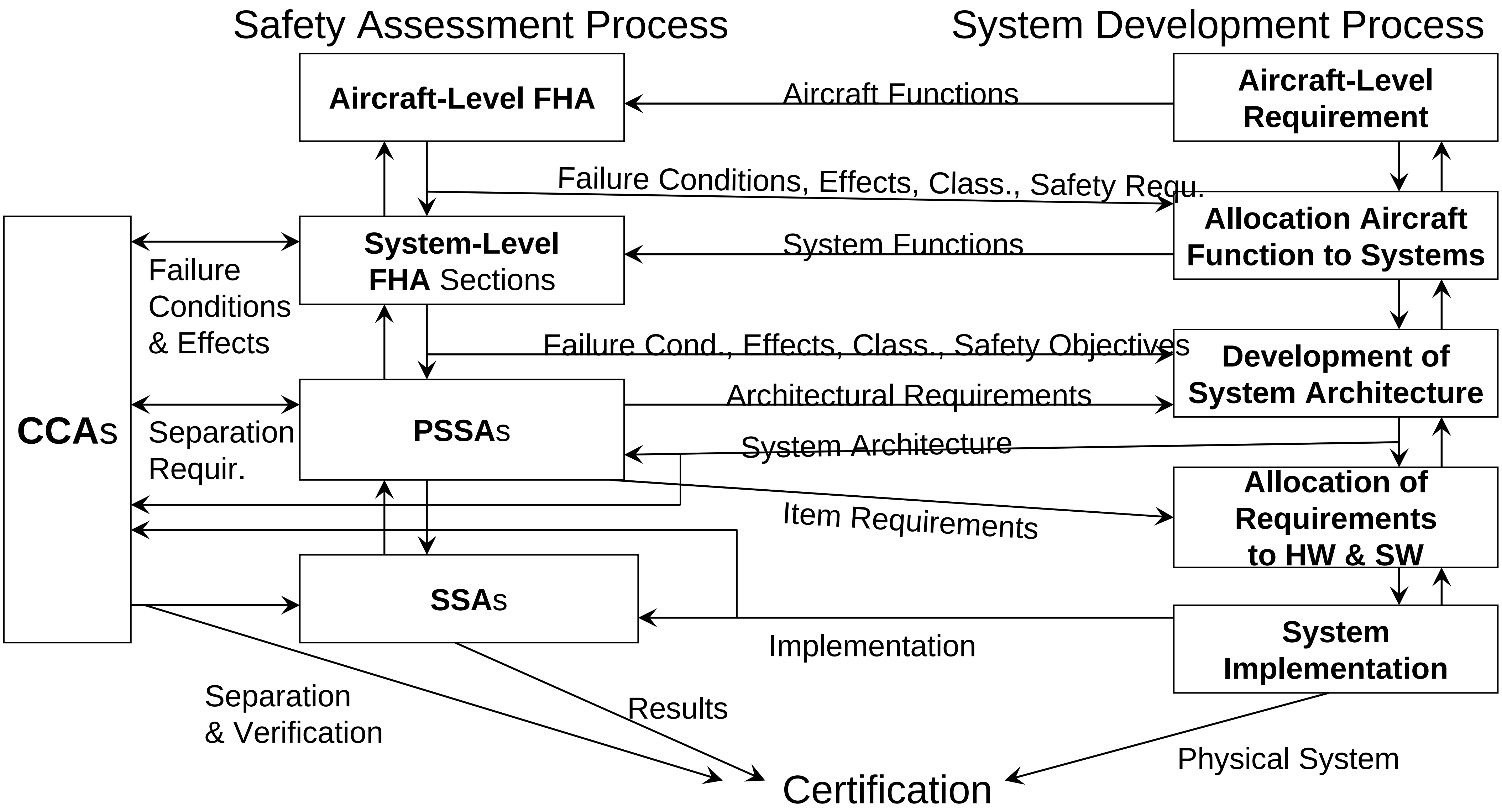
System Redundancy required:

- **HW redundancy: sensors, actuators, ECU's, busses (Flexray) doubled**
- **Information redundancy: error detection/correction codes used**
- **Time redundancy: all messages send twice on each bus**
- **Software Redundancy: two version programming?!**

Certification required as in aerospace industry?

Safety critical aerospace ECU development

Development Standard SAE ARP 4754



Distributed ECU's in cars - design challenges

Still increasing complexity (more comfort and safety functions coming)

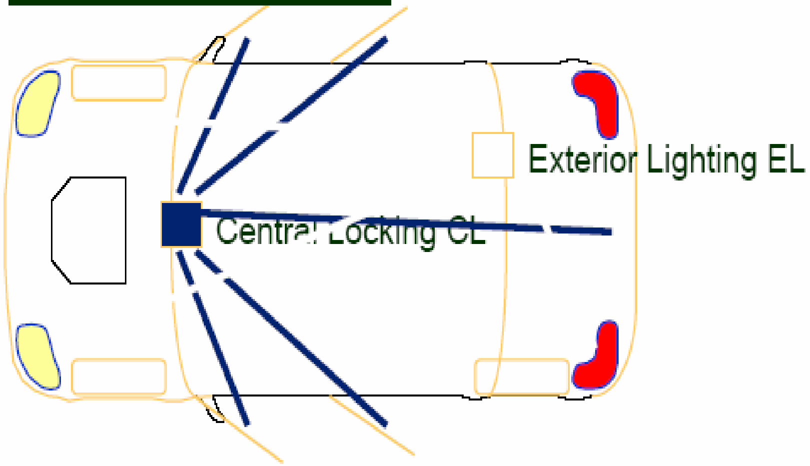
Today's E/E architecture in a car is characterized by an assembly of (too) many locally optimized subsystems

Only OEM can go for global optimum

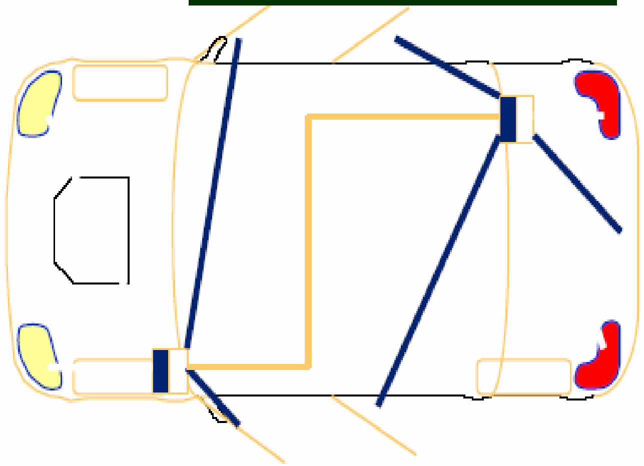
new system level design exploration tools are required

EE-Architecture alternative solutions

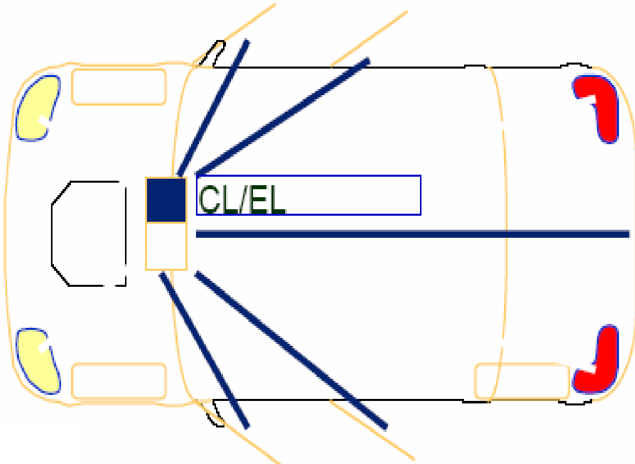
System Oriented



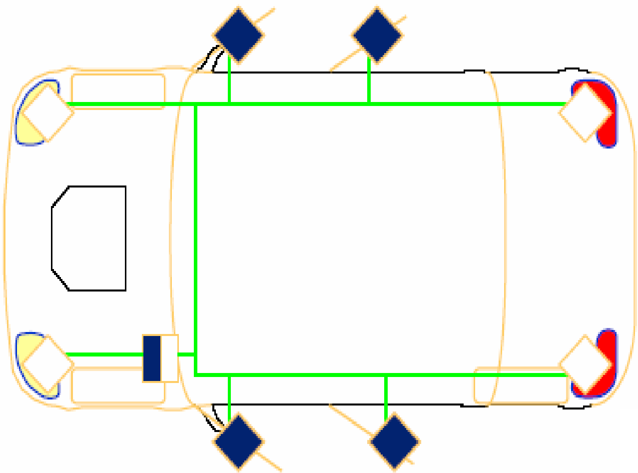
Topology Oriented



Central Control Unit



Central Control Unit with Specific Sub Busses



Tools to support architecture based development process

Understand Principal Requirements	Develop or Select Architecture	Represent and Communicate Architecture	Analyze and Evaluate Architecture	Implement Systems Based on the Architecture	Ensure that implementation confirms to the architecture
Analyze lessons learned Functional analysis (behavioral, perfomance, quality, cost) Analysis of existing vehicle models (final version) Technology forecast Results from research projects	Develop system requirements and quality attributes Architectural Tradeoff Analysis Results from research projects Identify major building blocks (e.g. transceivers, standard software components)	Communication to - management - development engineers - suppliers General definition of the - scheduling implications - work assignments - test activities - maturity implications - design rules Model building simulation, verification rapid prototyping	Analysis of runtime behavior in realtime environment - performance - behavior - communication patterns Identify optimization efforts	Adapt architectural design processs to vehicle specific schedule (master project plan) Vehicle specific definition of the - scheduling implications - work assignments - test activities - maturity implications - design rules	Design Rules Quality Attributes Reusable building blocks (OS, NM, COM, add. Services) Specific requirements defined in QGs Reviews and Inspections **Lessons Learned**

Courtesy J.Bortolazzi, DC

Requirements for new system level tools

Model based design as a basis.

- Is accepted in research and predevelopment, not yet standard in ECU development

Design space exploration means

- distribution of hardware and software under consideration of sensor/actuator locations
- computation performance as well as communication performance
- Co-design not only for hardware and software but also function, safety, security

- Metrics and parameters used are domain specific therefore, domain specific system level tools are required interfacing seamlessly with component specific tools (meet in the middle).

A lot of model transformations are required

Architecture Layers in Concept Development

Electronical

Electrical

Physical
Geometrical

HW

SW

Function Architecture
Functions and Subfunctions?
Interaction?

Software Architecture
Software Structure?
Standards?

Network Architecture
How do ECUs communicate?
Performance Characteristics

Component Architecture
How many ECUs, what performance?
Variants, scalability?

Power Supply/Power Distribution
Electrical Power Supply System?
Sources and Sinks, Dynamic Behavior, Idle State
Power Consumption (Ignition Off Drain Current)?

Component Topology
Where are EE Components Located?
Assembly/Maintenance/Recycling?

Interfaces
Relationships
Mapping
Processes

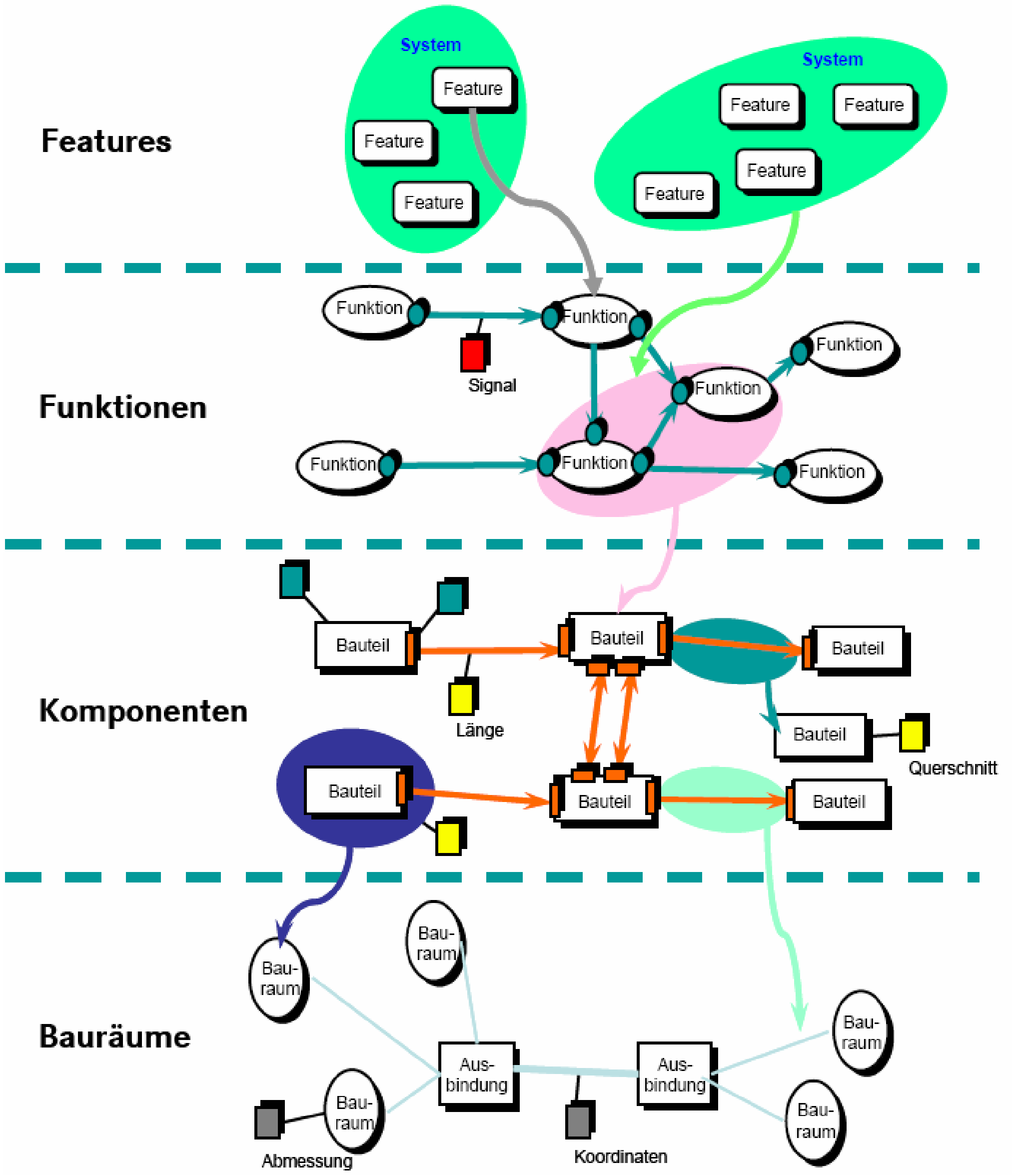
Abstraction Layers

Top-Down-Design

Bottom-Up-Design

List of Features	FL	Which features?
Feature Function Network	FFN	What is the concept behind the features? How do they interact?
Functions Network	FN	Detailed Specification of Functions architecture
Components and Networking	KN	Comp. Architecture Network Infrastructure Power Distribution
Topology	TOP	Topology wiring harness

Abstraction Layers



Typical domain specific views

Features

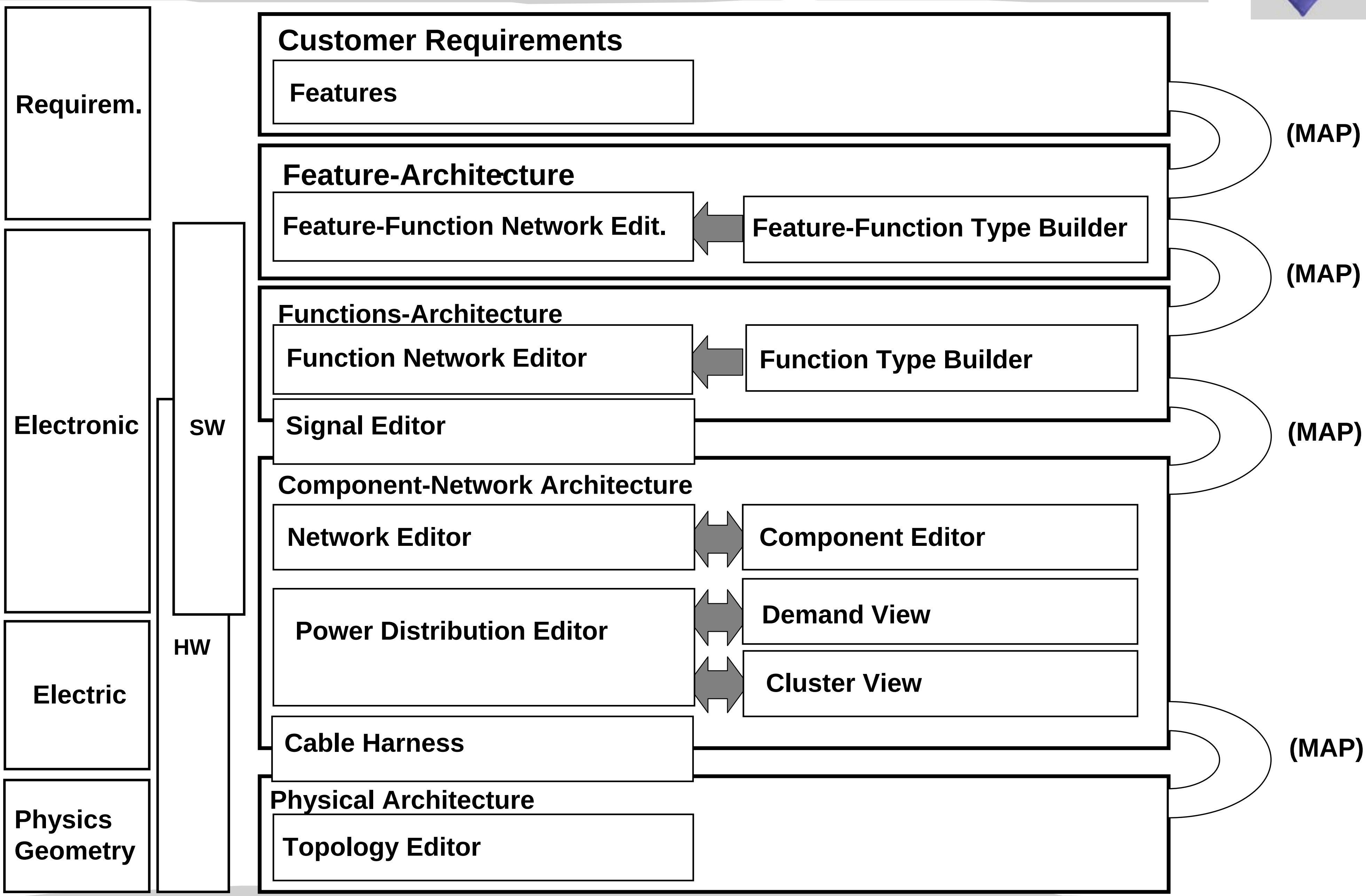
Functions

Components

Component locations and wiring

Design space exploration needs domain specific metrics and parameters

Abstraction layers of new EE-concept tool



Overview EECT

EECT is development name for “Electric/Electronic Concept Tool”

A prototype of EECT was developed in co-operation of FZI and DaimlerChrysler AG

Commercial version by aquintos GmbH

Release 1.0 was released December 2006**, availability to** General Market

Some Benefits of the EECT

- Support for concept evaluation of E/E-Systems in early design phases
- Complete meta-model for the description of automotive E/E-Systems
- Special diagram notations for Layers
  - Feature List, Feature Functions Network, Function Network, Components, Topology, Cable harness
- Metrics interface for calculation of E/E-architectures
- Variant Management
- Interfaces to different industrial standards: Fibex, DBC, etc.
- Documentation

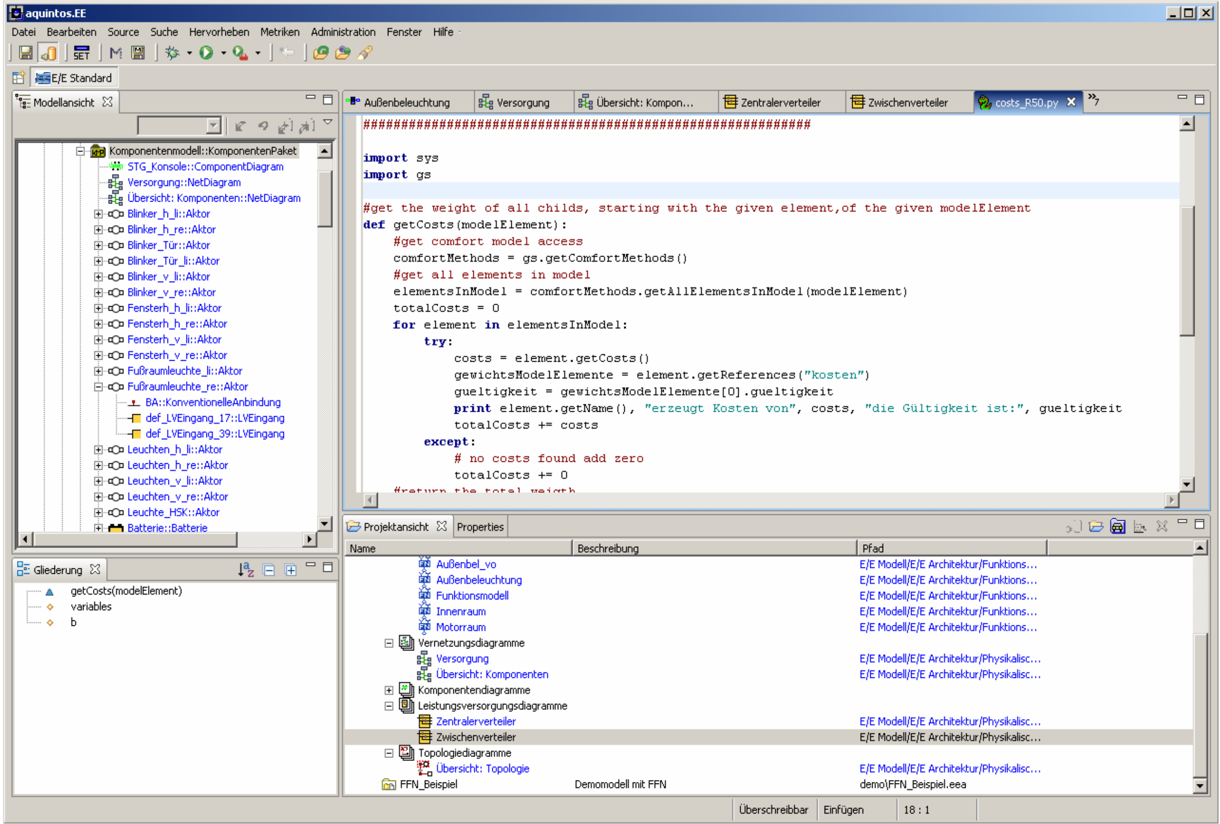
Evaluation / Calculation of the EE Concept

User defined metrics are supported

Metrics are implemented in Python

Metrics examples:

- ◻ Count metrics
  - Weight
  - Volume
  - Space
  - Networking Complexity
- ◻ Costs
- ◻ Power calculation



Highlights of Model-to-Model-Technology

Optimized Transformator-Engine with Interfaces to

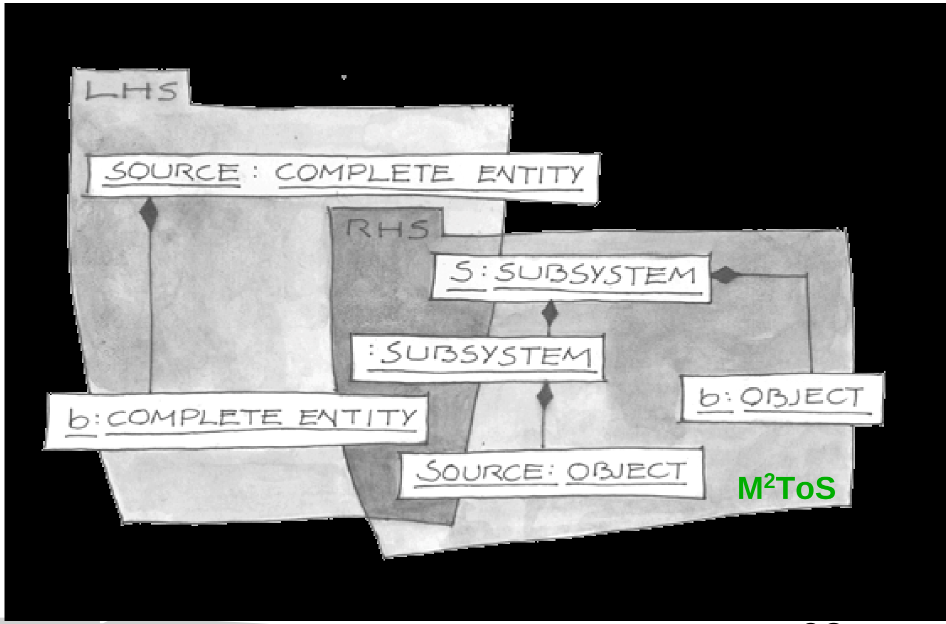
- ETAS **ASCET**® (>= 5.1)
- The Mathworks MATLAB®/**Simulink**®/Stateflow® (R13 – R16)
- Fully integrated in PREEvision (for model consistency checks, variant propagation…)

Model-based Specification of Transformation Rules

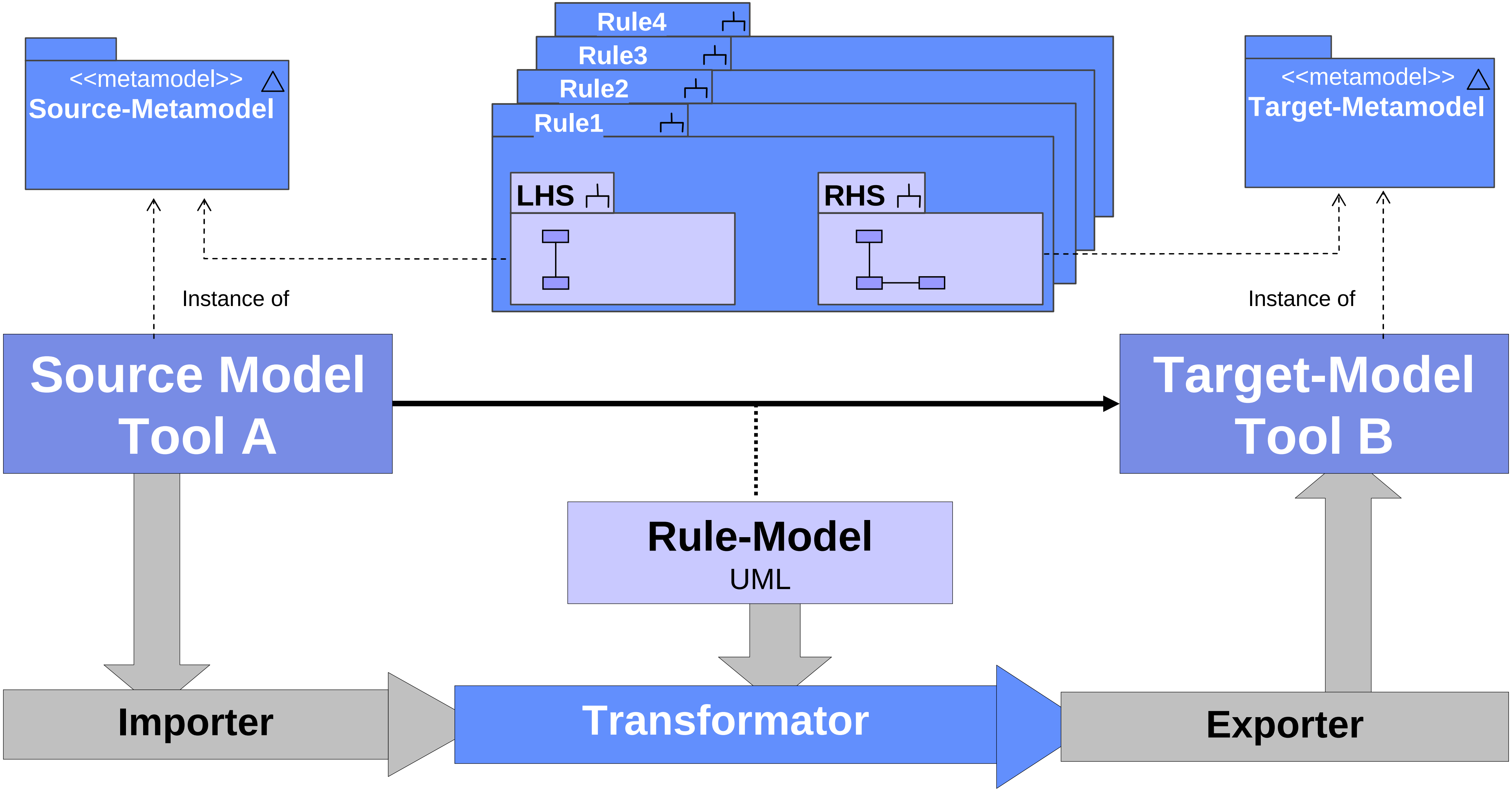
- **Rule Set modeled with UML**
- Maintainability, Readability
- Automated Code Generation of the Rule-Set, no manual design process behind

Purpose of M2M Transformation

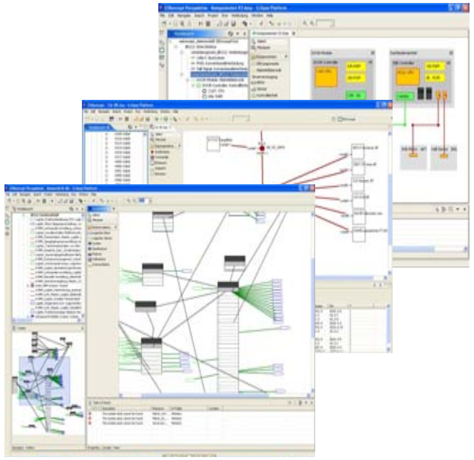
- Model data migration
- Model-**Refactoring**
- Model-**Optimization**
- Model-Verification



M2M Engines Architecture



EE-Architecture Concept Tool PreeVision (www.aquintos.com)



Graphical Editors
Variants Management

Excel
Word

Documentation
Analysis,
Metrics

E³.cable

Microsoft EXCEL

Telelogic Doors

Simulink u.w.

Open API & M2M

eclipse 3.0

Model-Data Backbone
→ Multi-User (DBMS) / Single-User (XML-File)

MySQL® Oracle®

Tool-Framework for Development using Eclipse-Basis
- Extensibility
- Open API

Supports Model Exploration

Model Management
- Multi-User (Database)
- Single-User (File-based)

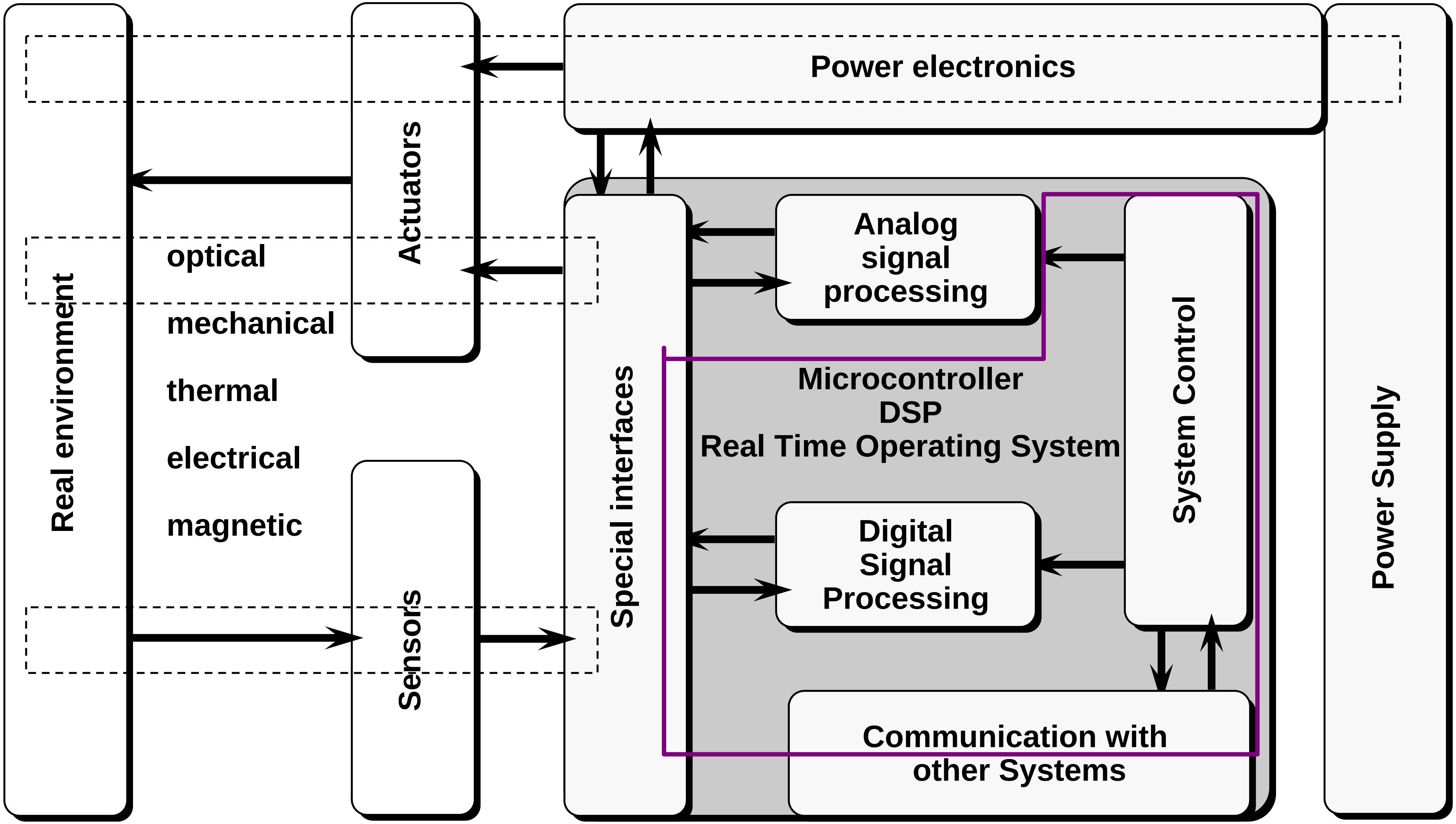
Variant Management
- Kernel based on pure:systems technology

Export / Import Filters
- DBC
- FIBEX
- KBL
- MATLAB/Simulink
- UML ARTiSAN Studio

Report Generation
- BIRT Technology
- User Configurable Reports

Metric-Interface
- Python, alternative Java API

System Level Tool Support



Not seamless

somehow satisfying support: standard hardware platforms, software, RTOS, Sensors und Actuators

Conclusion (1)

- **What system level tools should provide**
  - Documentation (readable for men, specific for application domain)
  - Data exchange between all designers across company boundaries
  - Data exchange between computer aided tools supporting distributed databases
  - Intellectual Property, reusable in libraries
  - Parameterized for variant design
  - Supporting standards and guidelines (e.g. HIS, Autosar)
  - Testable (Fault models, automatic Model validation), quality assured (automatic generation of test pattern and test bench) and documented (what is modeled, but also what is not modeled)
  - Seamless in design flow (Analysis, Design, Verification, Integration, Validation, Test, Application, Diagnosis)
  - Reviews, Rule Checking, Simulation, Formal Verification, Model Checking
  - Synthesis, automatic, interactive optimizing (e.g. RP-Code, Production Code)
  - allow access for automatic parameter-extraction

Conclusion (2)

Design studies show:

- **Model based methodologies and tools are well performing and promising**
- **Seamless design flow only partially given (e.g. digital hardware, software).**
- **Interfaces for Modeling, Simulation, Characterization mostly manual**
- **hard problem for design of embedded systems**
  - Cross sensitivity of Components (insufficient characterization)
  - Safety, Security, Function-Codesign
  - According modeling is really time and cost consuming
  - Mixed-Mode, Multi-Level-Simulation required
  - Formal Verification und Validation not possible?!
    - Non functional requirements
    - Time-, frequency- und parameter-domain
  - Module / System-Integration und –Test
  - Cross-sensitivities, EMC, Certification

Model based system design is possible,
but there are many design and analysis steps still missing, especially in early design phases.

Conclusion (3)

Industrial design practice shows:

- **Challenges for the design of embedded systems**
  - many modeling techniques from computer science not adequate: FSM, Hybrid Automata, LSC, MSC, Petri nets, process algebra, Statecharts, Temporal Logic, Timed Automata, Z …
  - Is academic willing to prove their research results for real designs?!
  - Seamless flow required with respect to industrial life cycle processes, therefore support of standard interfaces must be done also by academics
  - There exist large libraries in different description methodologies that can‘t be neglected
  - There exist standard RTOS (OSEK/VDX) and bus systems
  - There exist tight cost boundaries
  - New algorithms and tools must be made commercially available
  - Engineering constraints, adequate description methods according to De-Facto-Standards (tools) must be obeyed: Matlab, ASCET, Statemate, Doors, Saber, VHDL, C, Assembler
  - Formal methods are not yet scaling for many real industrial problems
  - Required from industry: availability of real requirements, constraints, cost numbers etc. for research
- **Required: more close cooperation between system manufacturer, (tier 1) suppliers, EDA companies and academics**

Questions

Thank you very much

for your attention

Contact:

Klaus Müller-Glaser

Universität Karlsruhe, ITIV

kmg@itiv.uni-karlsruhe.de