FZI
Forschungszentrum Informatik
an der Universität Karlsruhe

13th Monterey Workshop, Paris, 18-October-2006

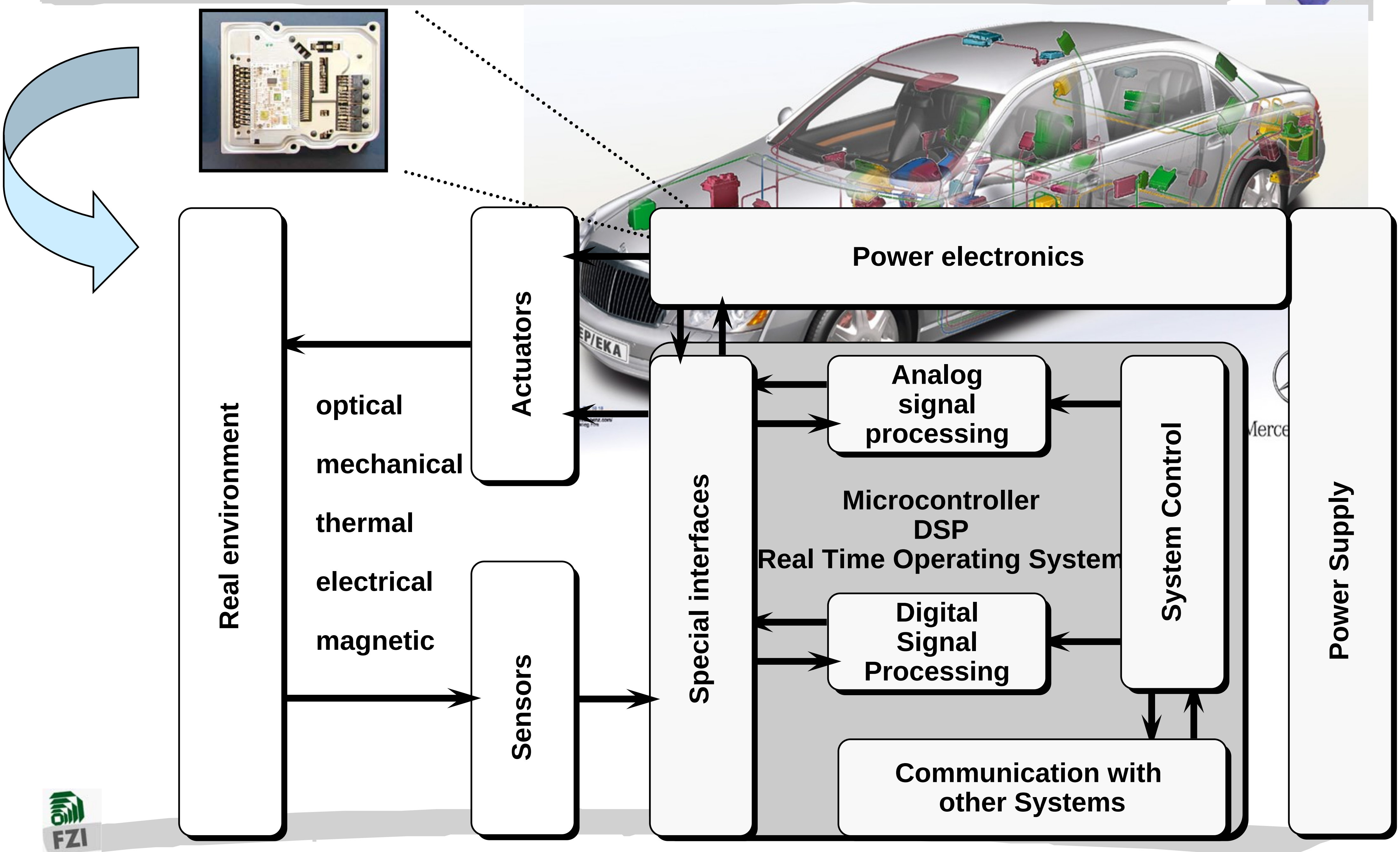
„Supporting System Level Design of Distributed Real Time Systems for Automotive Applications“

Prof. Dr.-Ing. Klaus D. Müller-Glaser
Universität Karlsruhe, Institut für Technik der Informationsverarbeitung (ITIV)

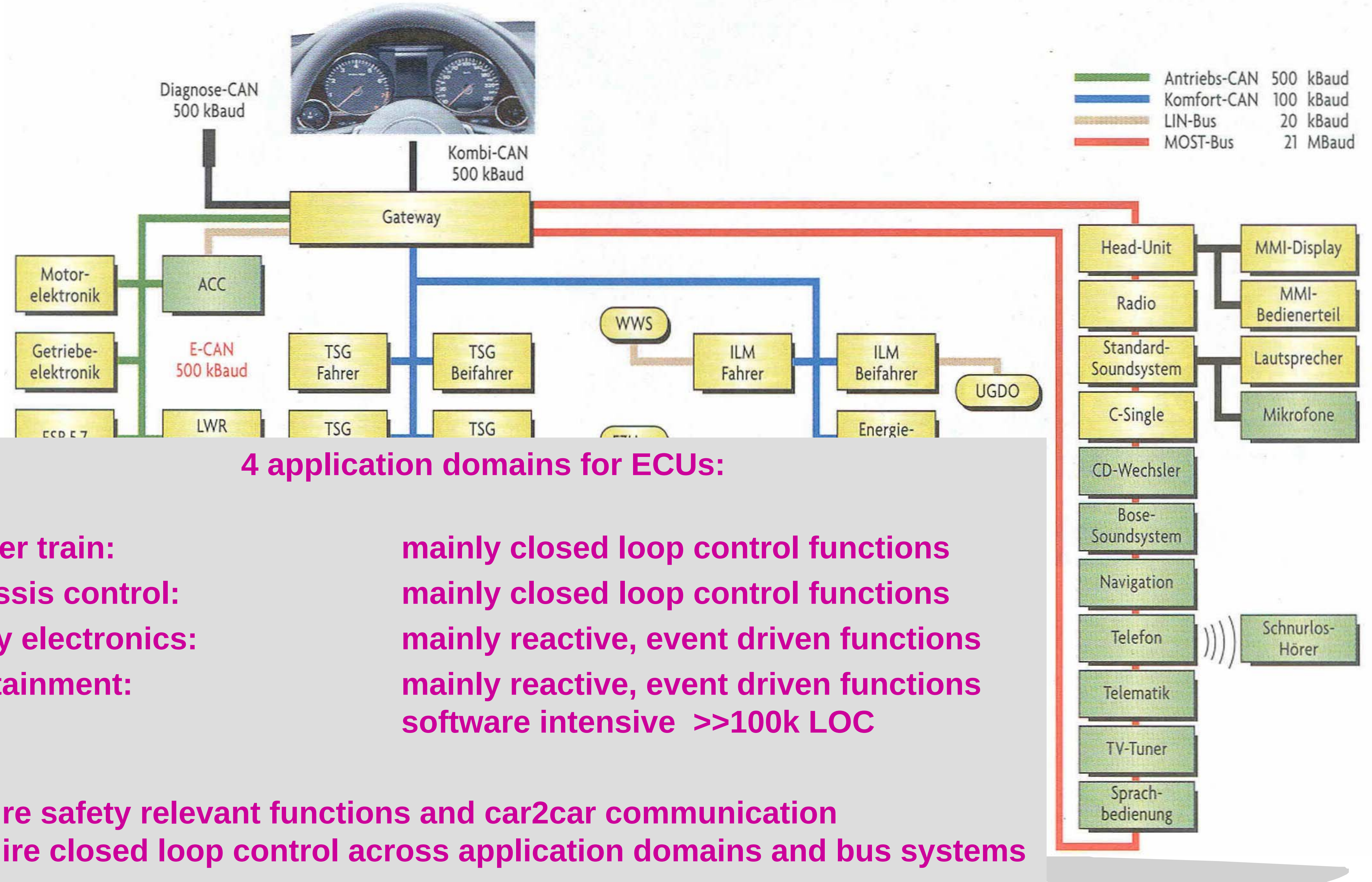
Content

- Automotive Embedded Systems
  - computation and communication
  - distributed and hard real time
  - industrial issues of composition of embedded systems
- Development Process
- Model Based Design
  - heterogeneous models
  - supporting tools, domain specific
  - new system level tools
  - meet-in-the-middle strategy
  - model to model transformation
  - tool support
- Concluding Remarks

Automotive Electronic Control Units ECU



Complex Communication (e.g. Audi A8)



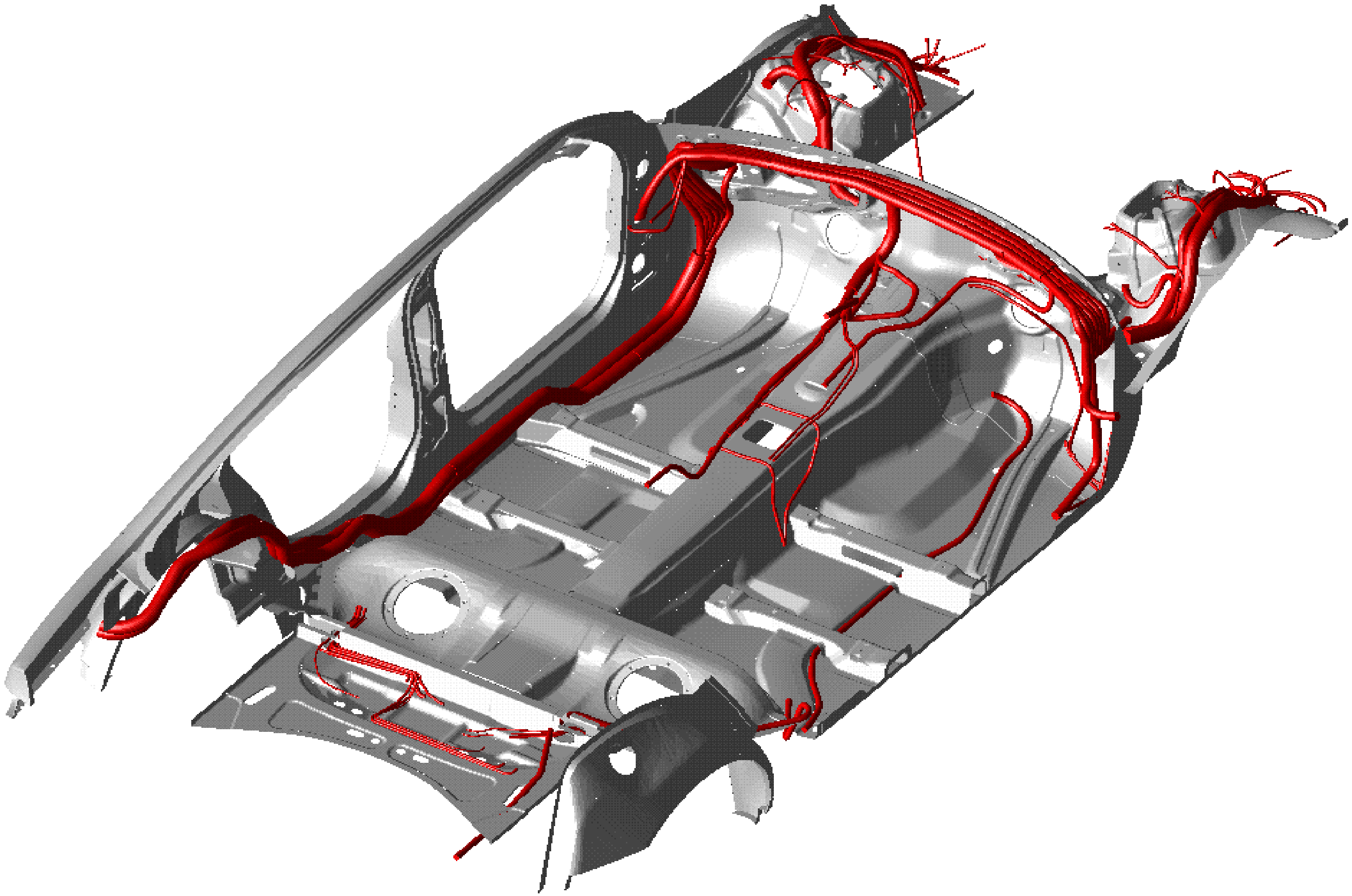
4 application domains for ECUs:

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

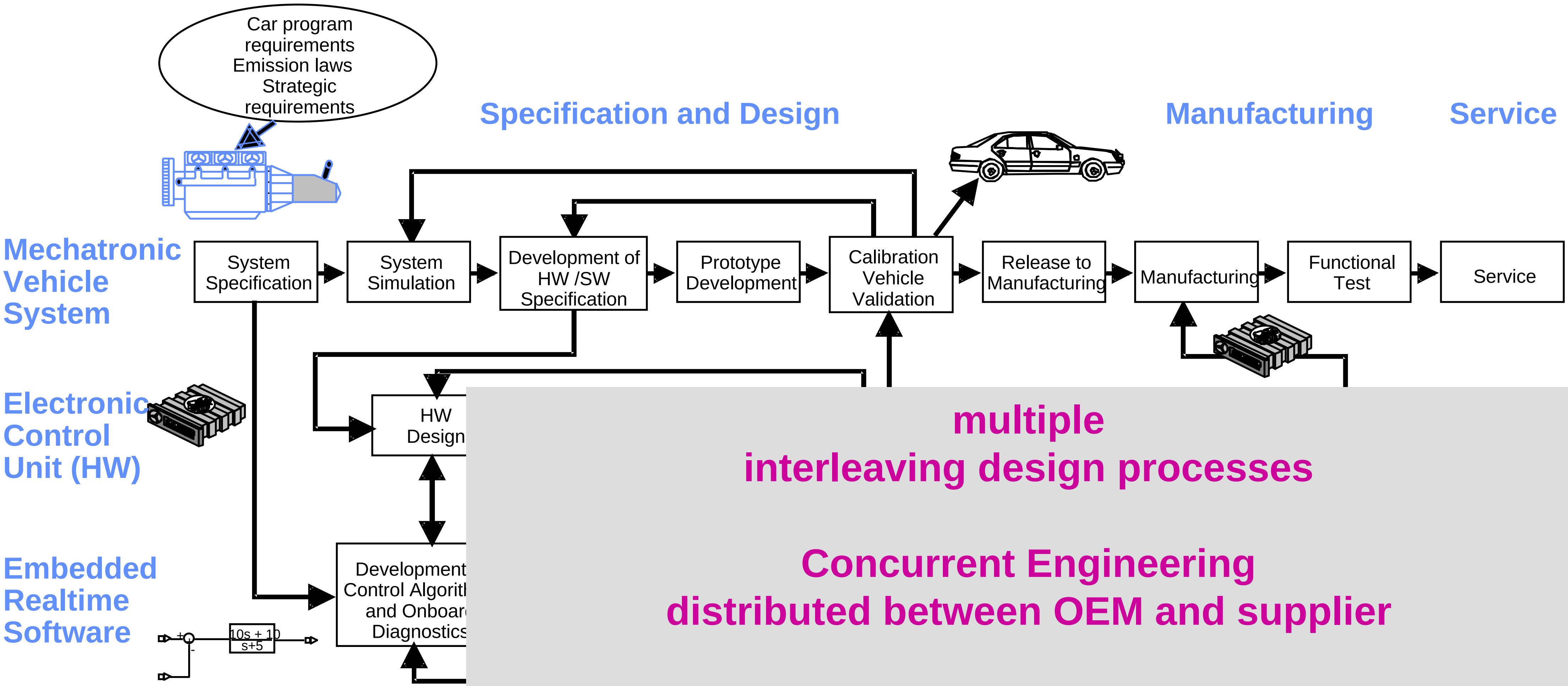
Future safety relevant functions and car2car communication require closed loop control across application domains and bus systems

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, ...)



Hierarchical Organization of Design Processes

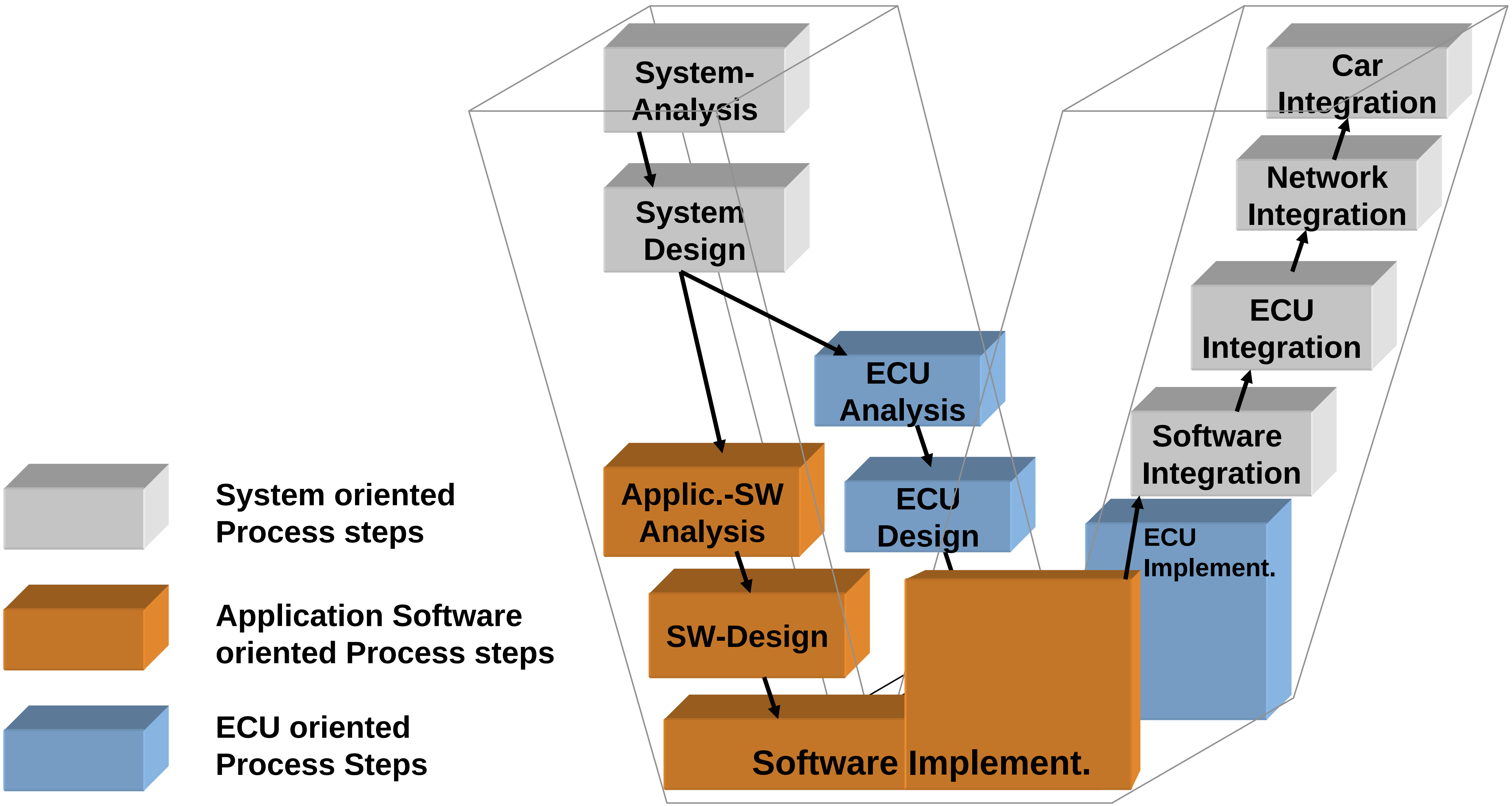


multiple
interleaving design processes

Concurrent Engineering
distributed between OEM and supplier

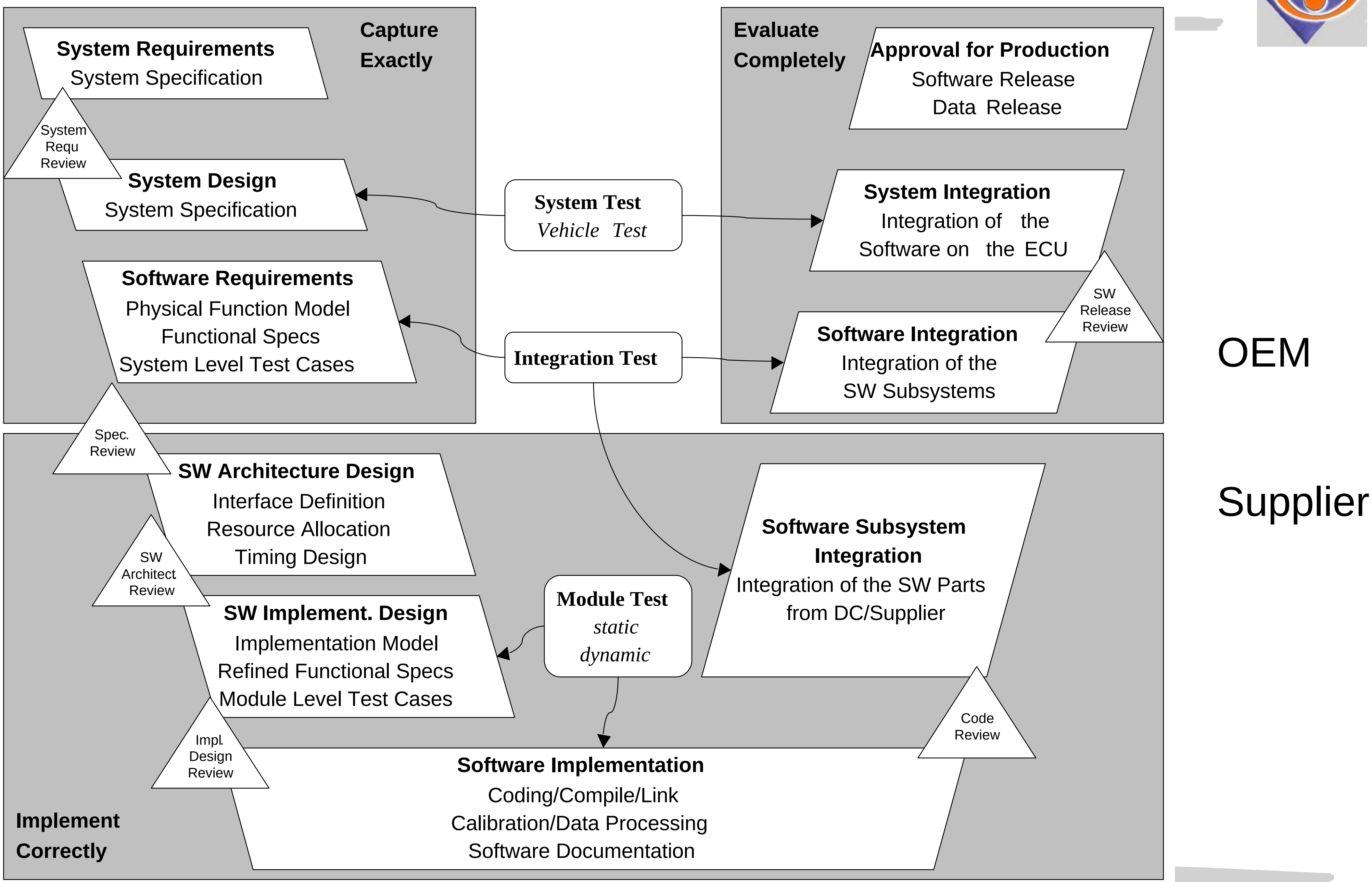
requires
comprehensive life cycle model (V-Model)
strictly controlled design methodology
supporting computer aided design tools

V-Model for automotive ECU's



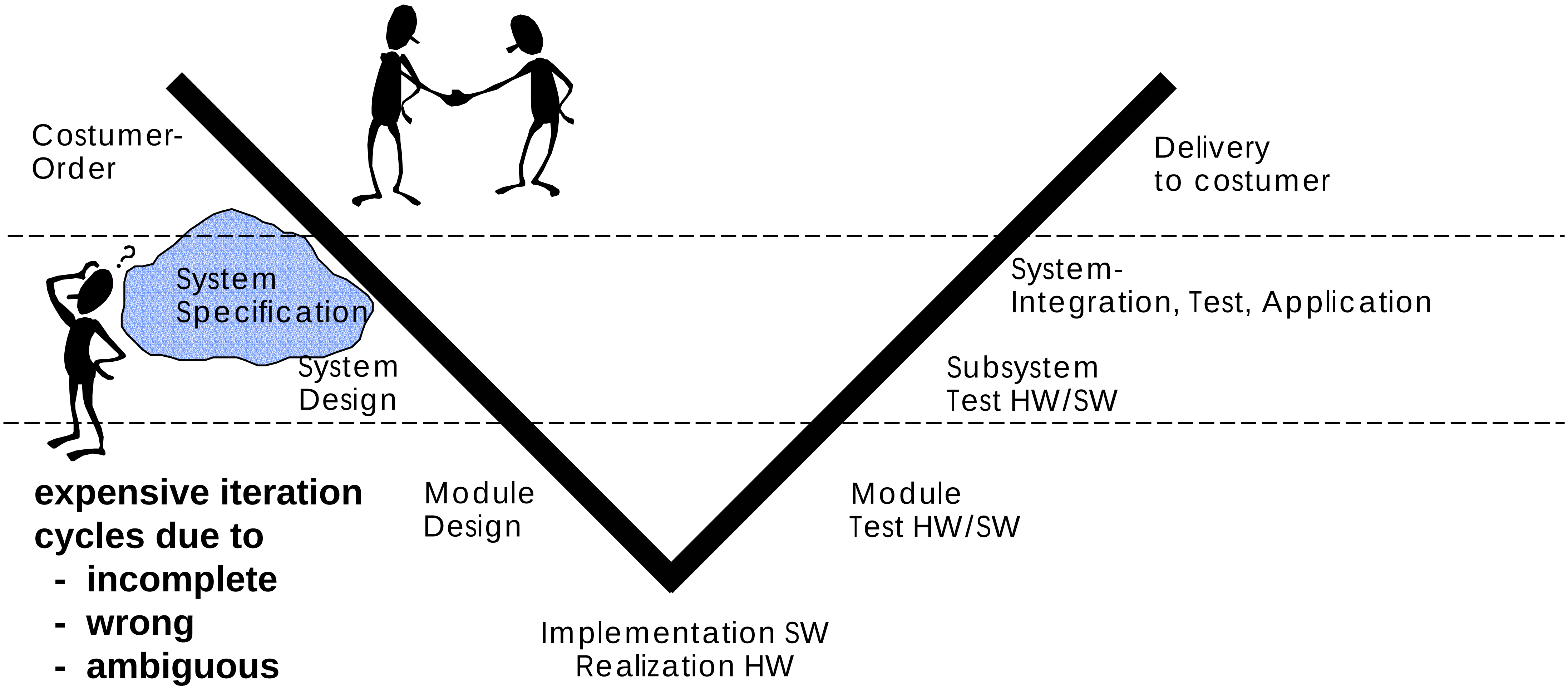
Courtesy ETAS

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

→ **Executable Specification, Model Based Design**

Modeling

Modeling 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

Use of characterized libraries (reuse, variant design)

Model verification through extensive testing

Model characterization

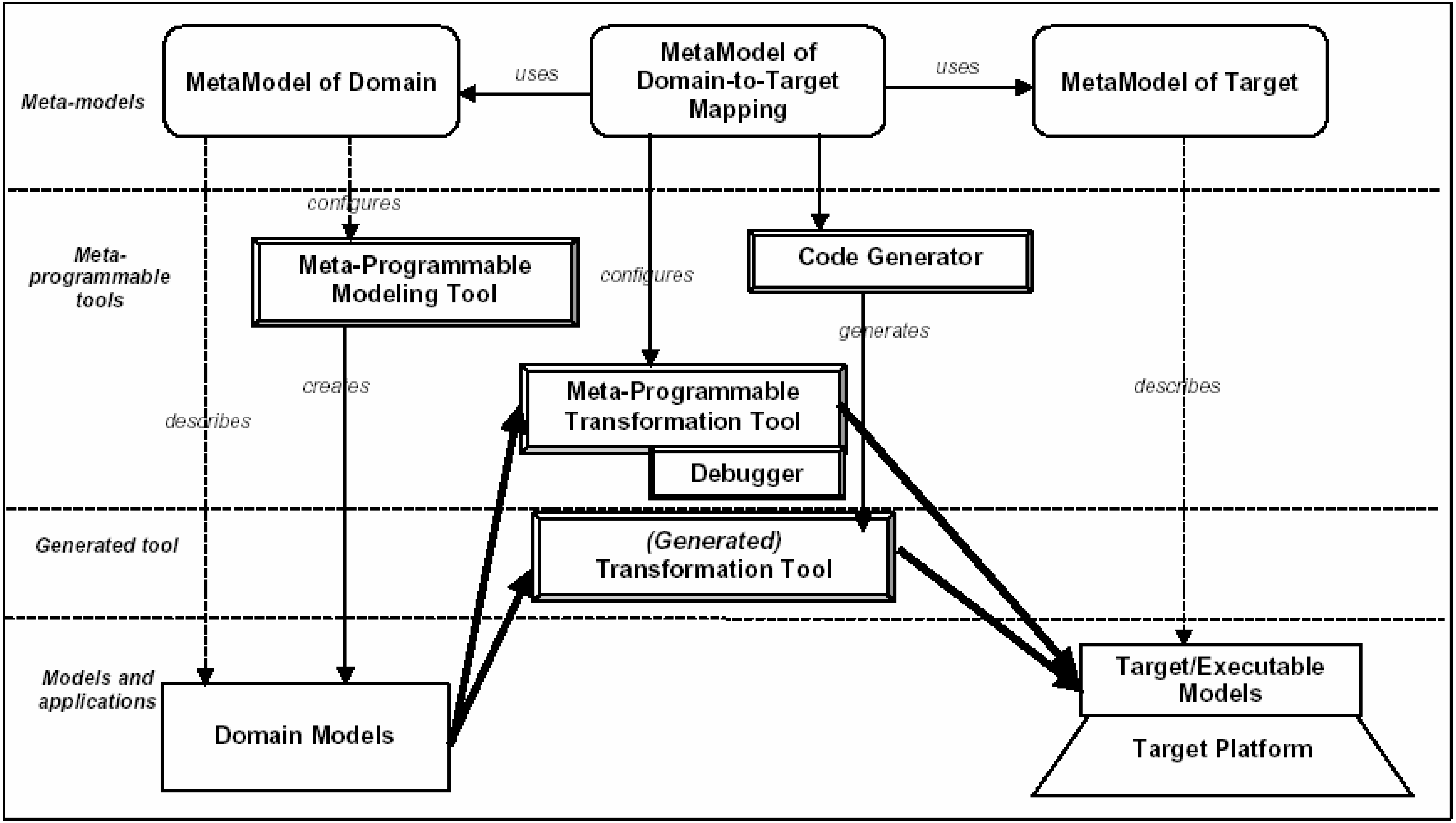
Model documentation

Macro modeling

Meta modeling

FZI

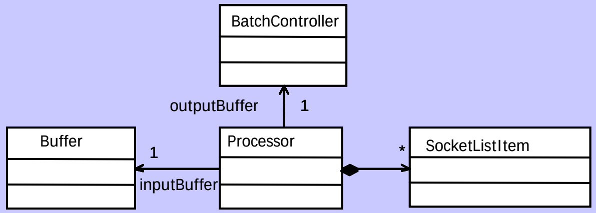
Meta-model based design (Sztipanovits, Karsai Vanderbilt University)



Modeling for automotive ECU's

Architecture

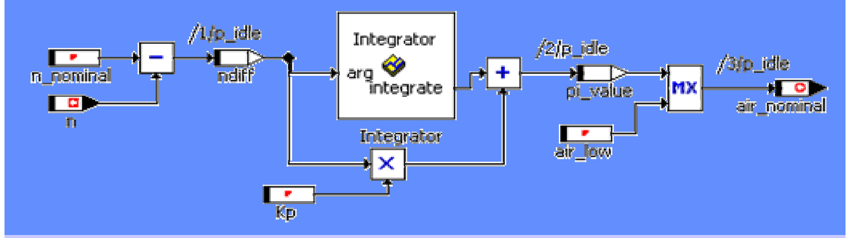
Modelling with UML



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

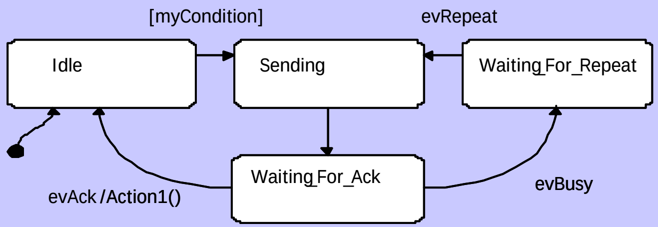
Modelling with block diagrams



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

Event driven

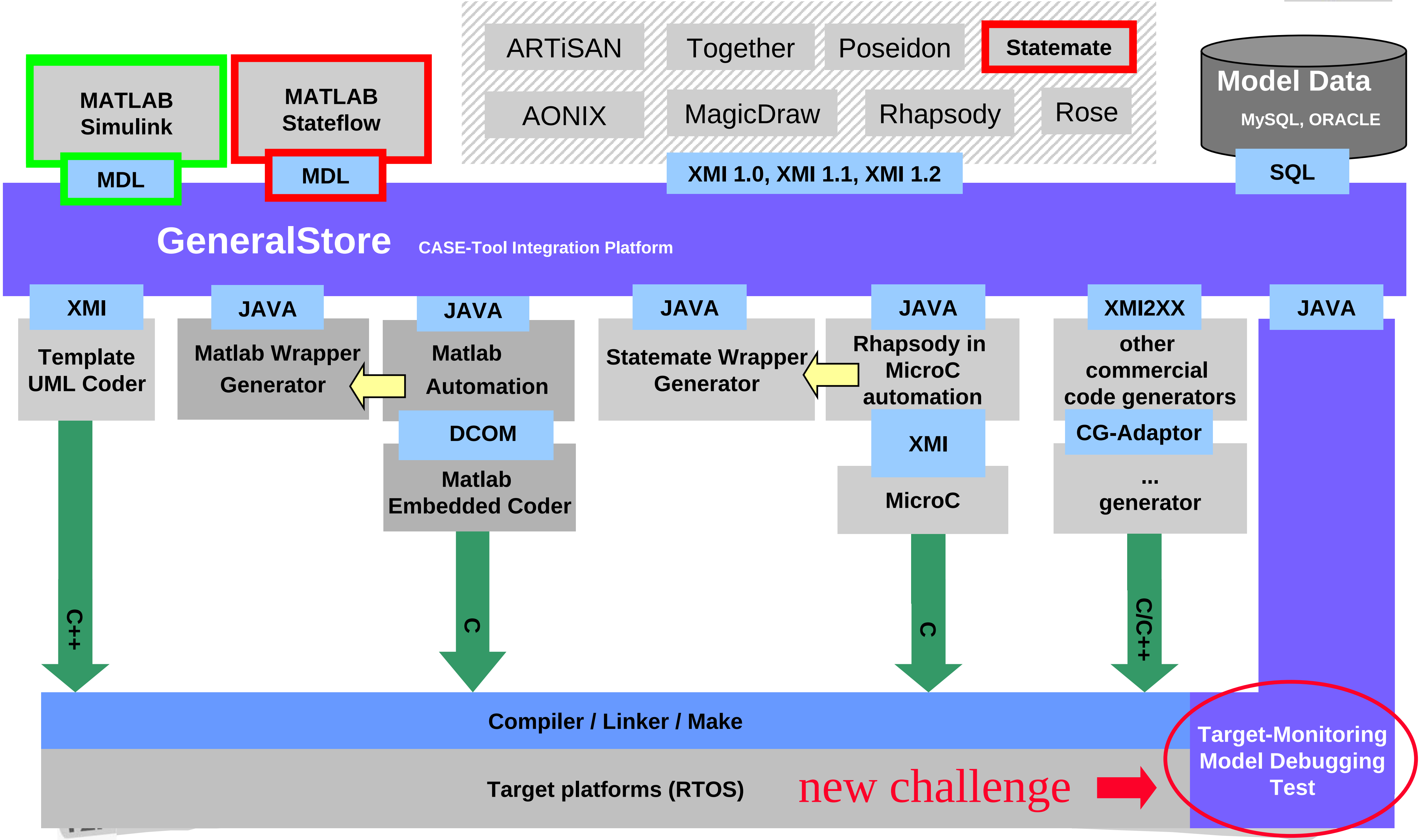
Modelling with state charts



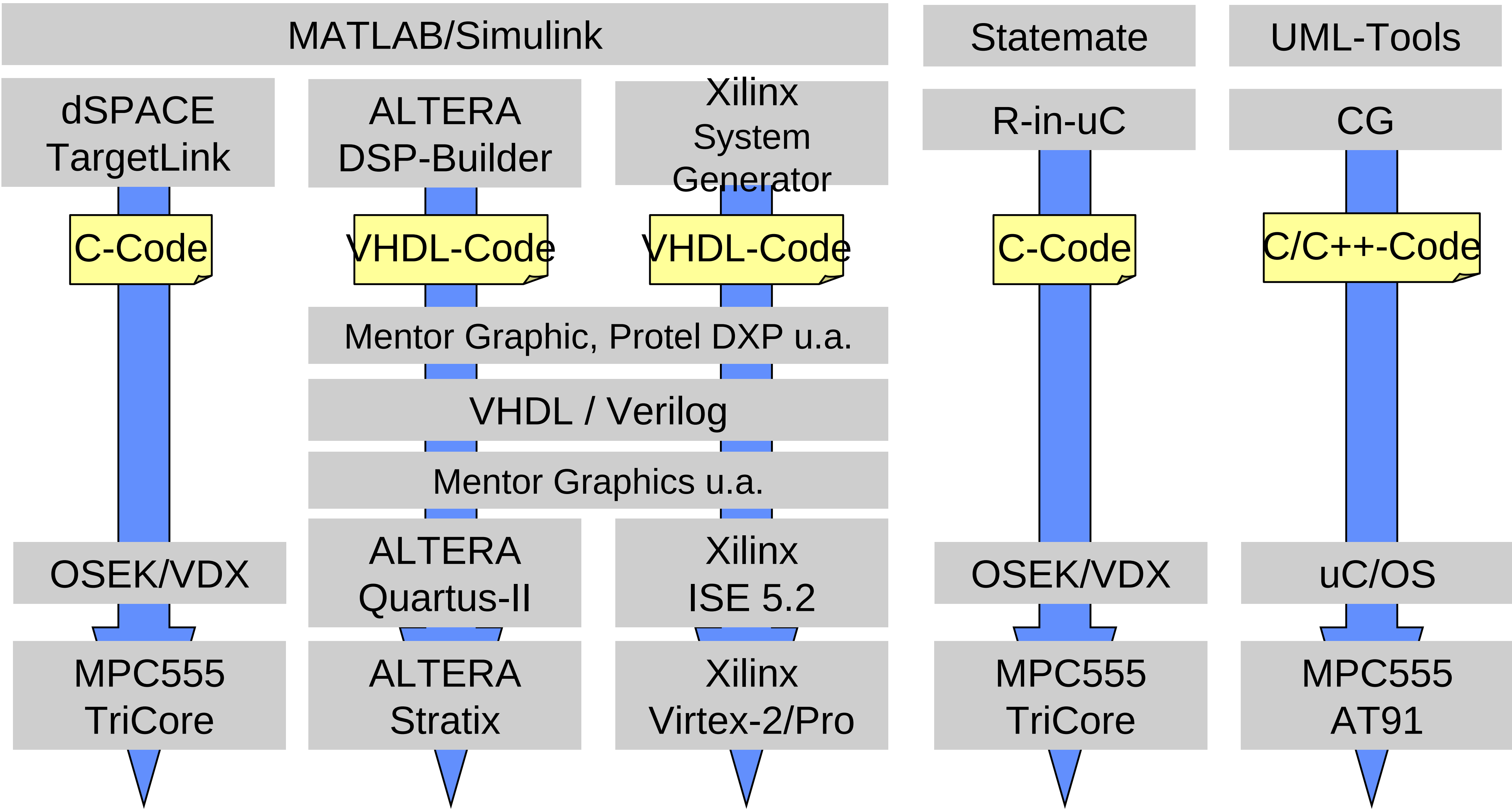
Rhapsody in C++ (i-Logix)
Statemate (i-Logix)
Stateflow (The MathWorks)
ASCET (ETAS)

Heterogeneous modeling requires integration platform
e.g. ETAS Integrio, Vector DaVinci

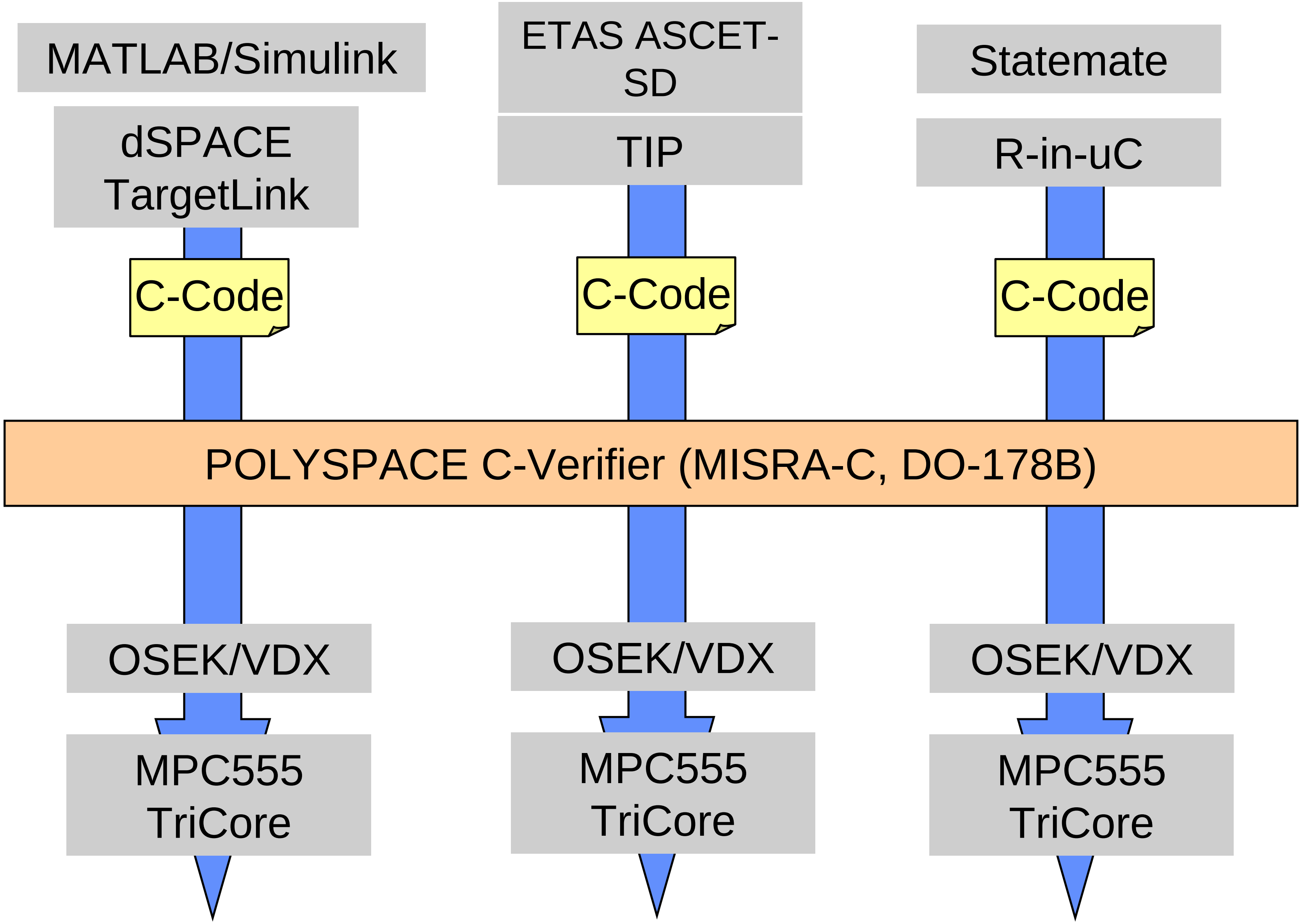
ITIV/FZI Tool integration platform (model transformation)



Tools Chains used at ITIV/FZI



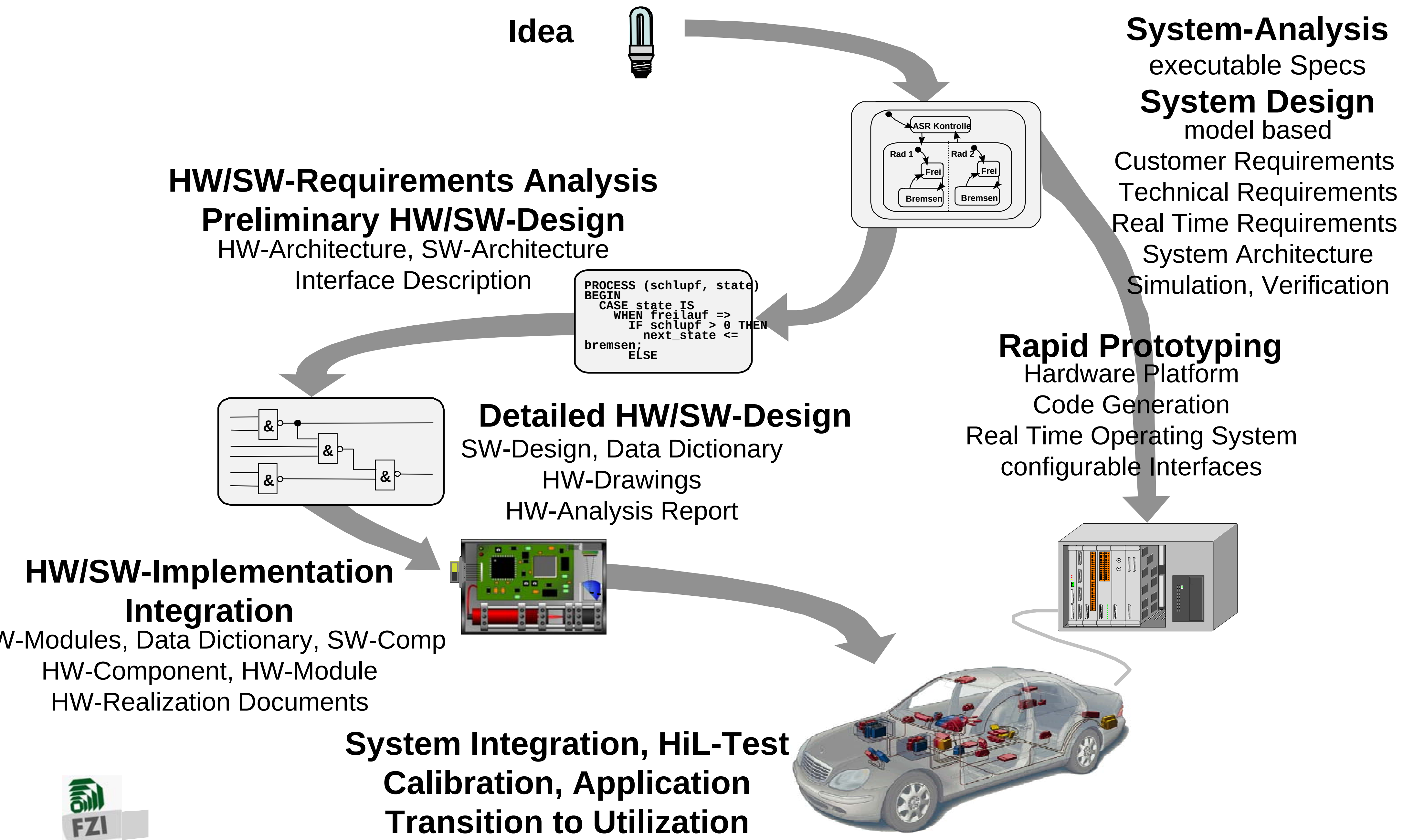
ITIV/FZI Tool Chains (Automotive) Verification Support



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)

Typical Design Flow



Distributed ECU's in cars - design challenges

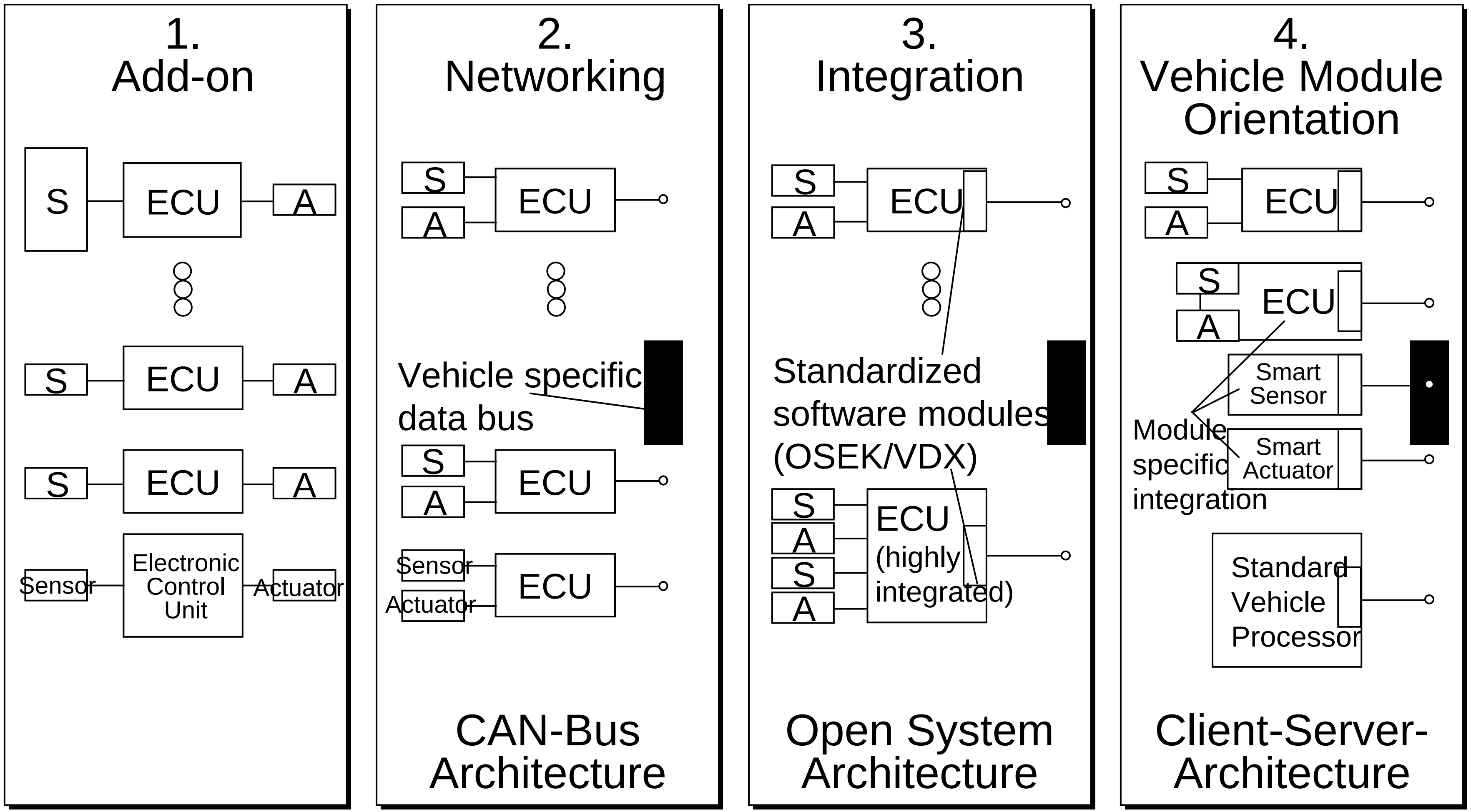
Still increasing complexity (more comfort and safety functions coming)

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?!)**

- **requires redistribution (mapping) of software onto fewer hardware platforms**

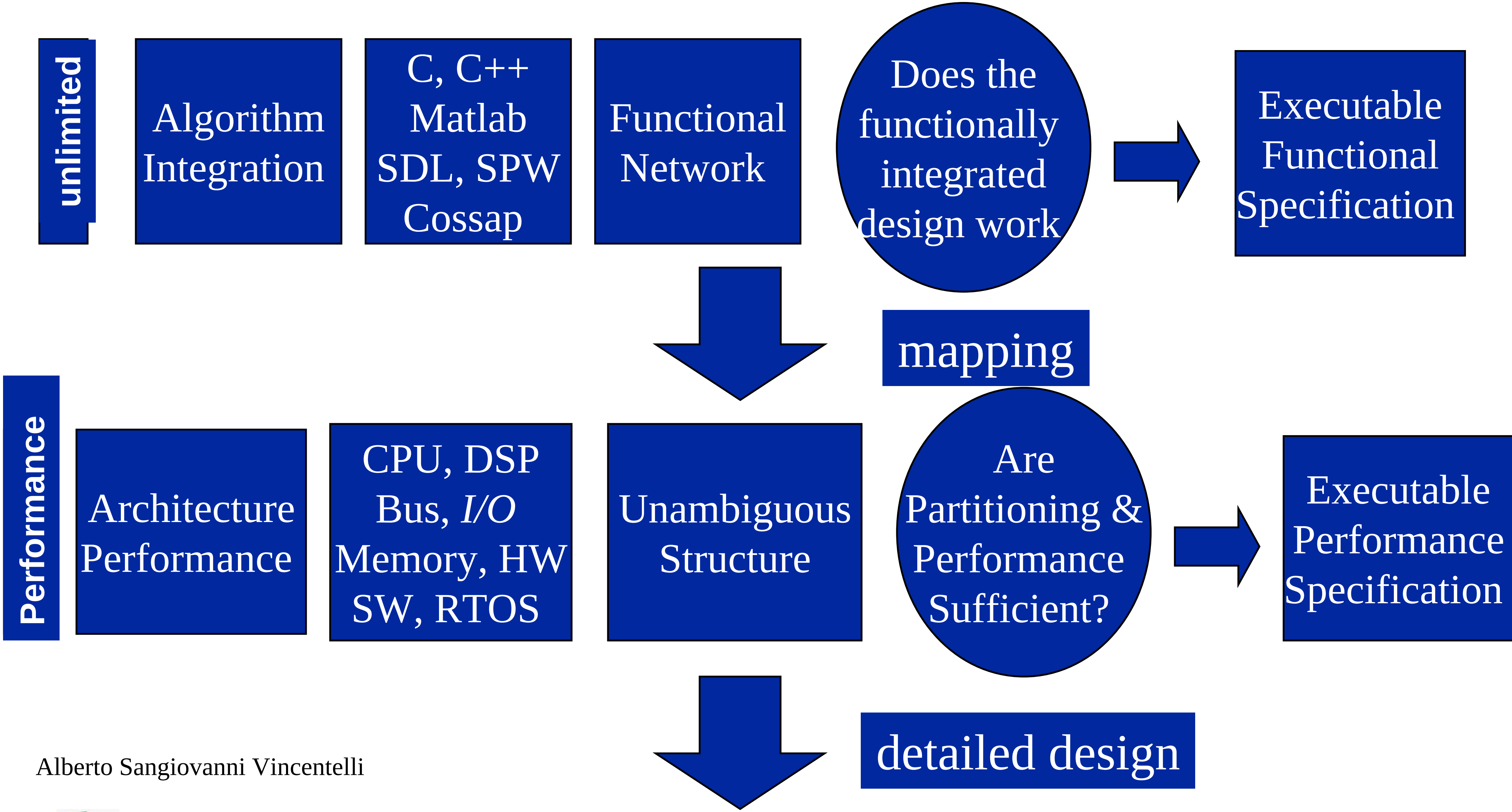
Evolution of hardware/software architectures in a car



Evolution led to open system architectures with modular software architecture:

Milestones: CAN, OSEK/VDX (AUTOSAR)

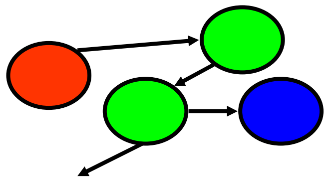
Challenge



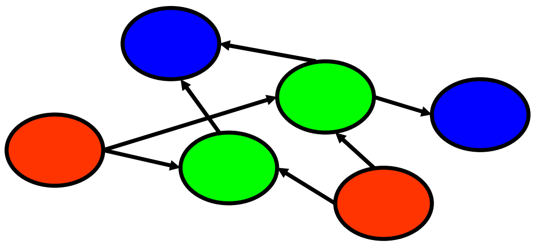
Alberto Sangiovanni Vincentelli

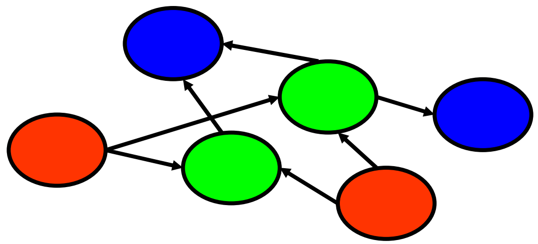
Desired

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

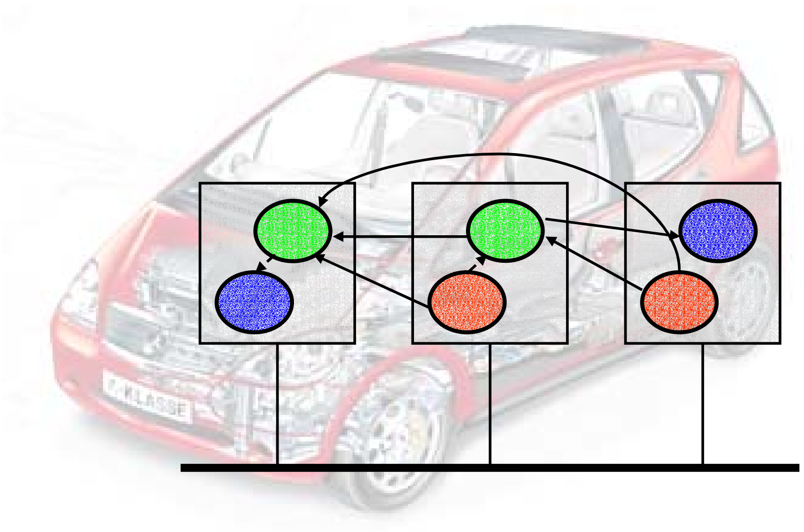


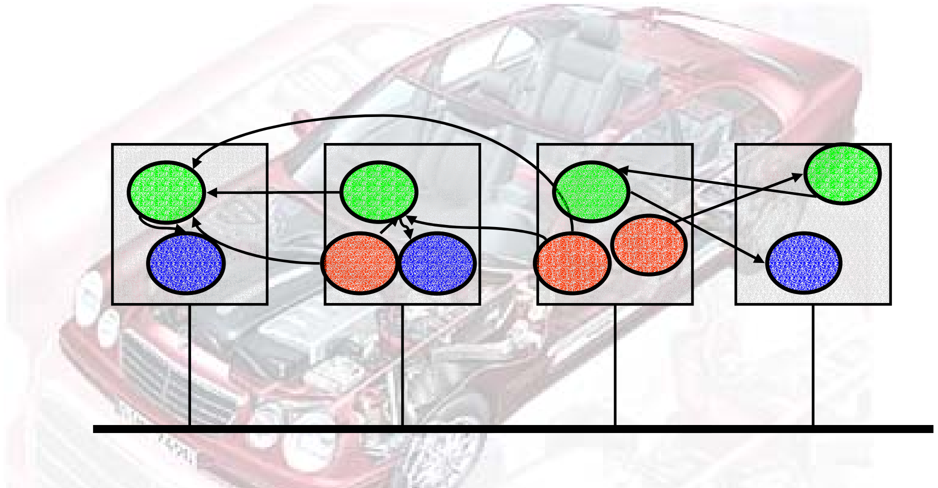
Additional Functions





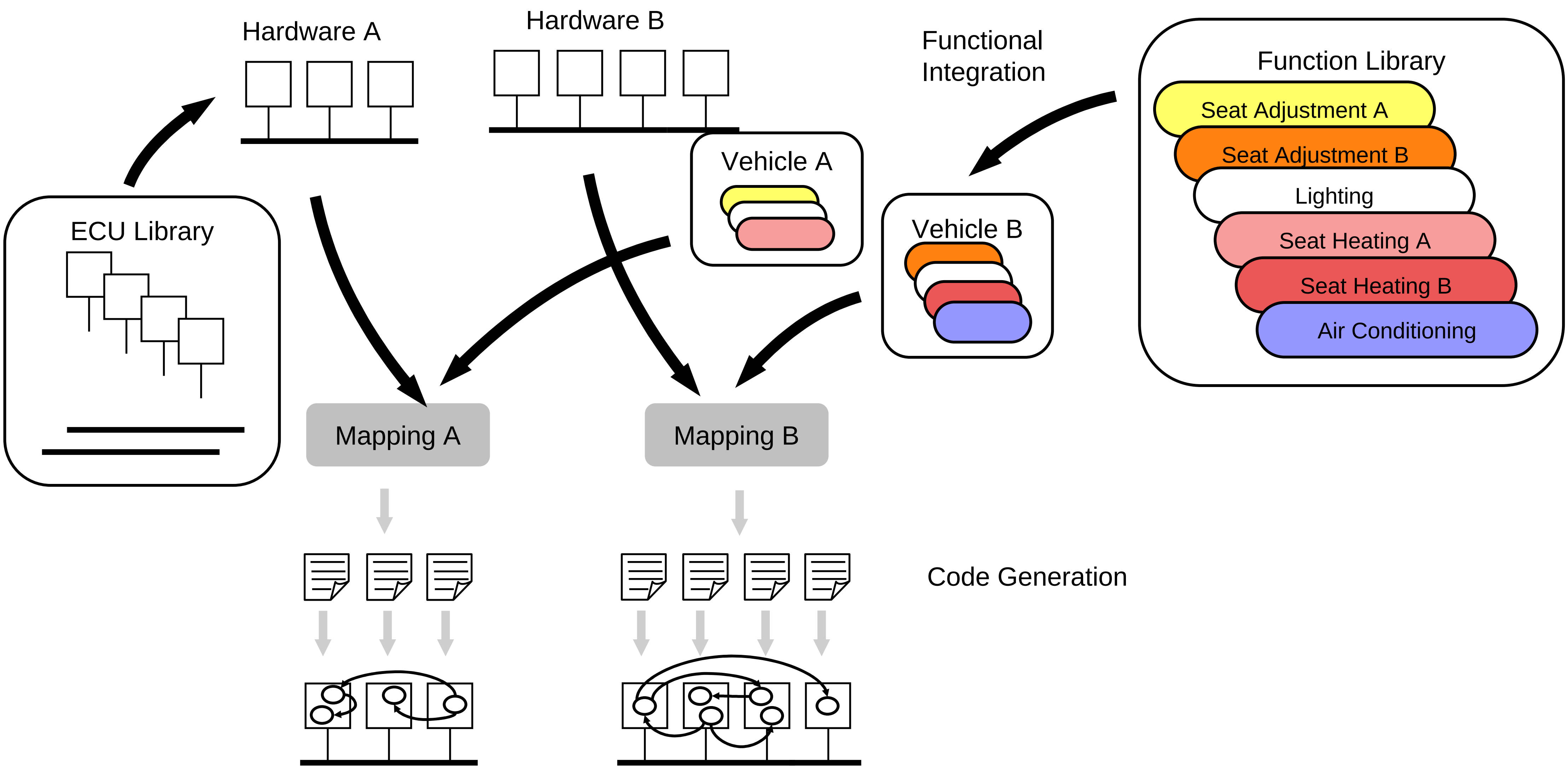
Basic Functions





Courtesy ETAS GmbH

Goal (AUTOSAR)



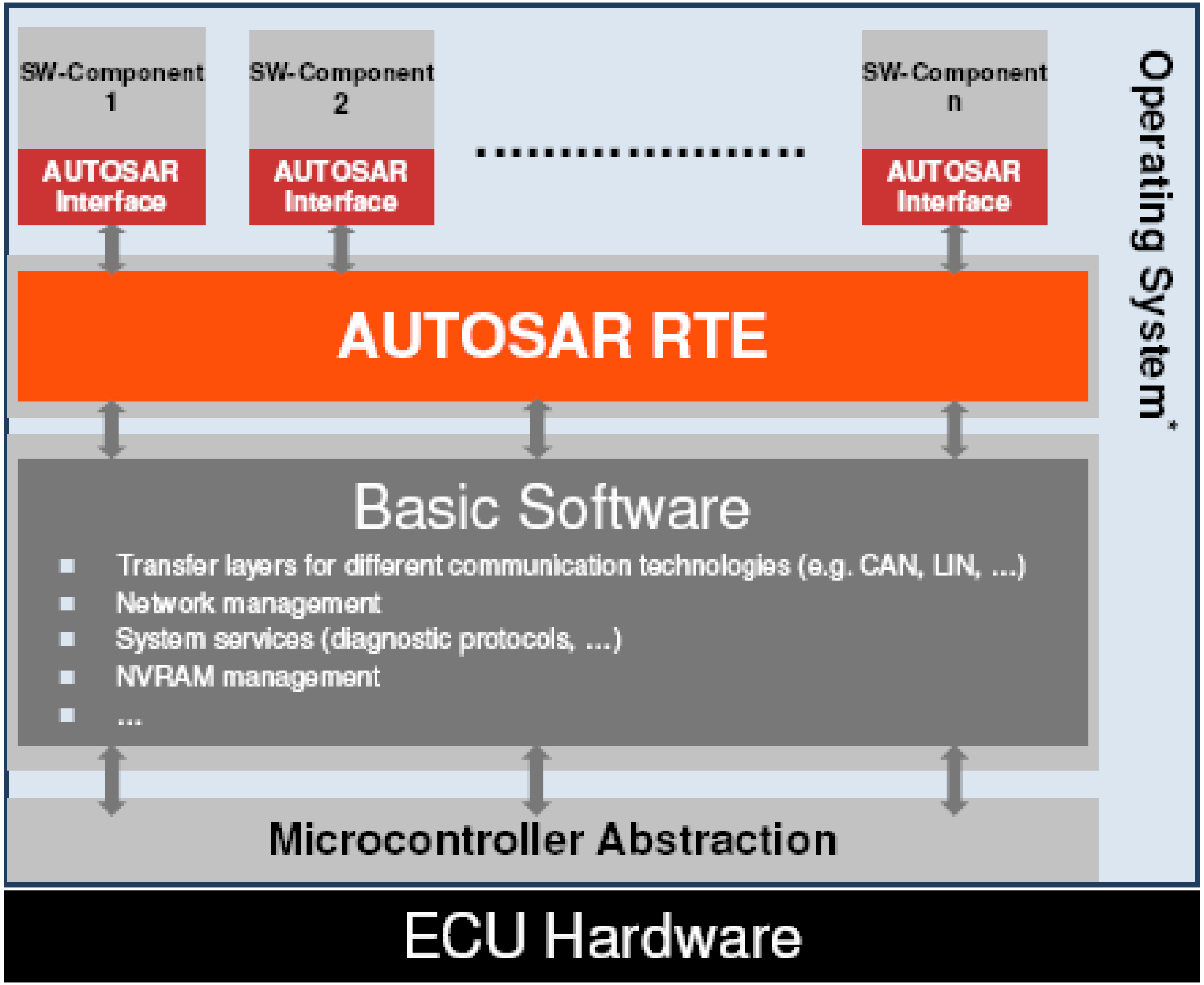
AUTOSAR

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, …

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?!)
- requires redistribution (mapping) of software onto fewer hardware platforms

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

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

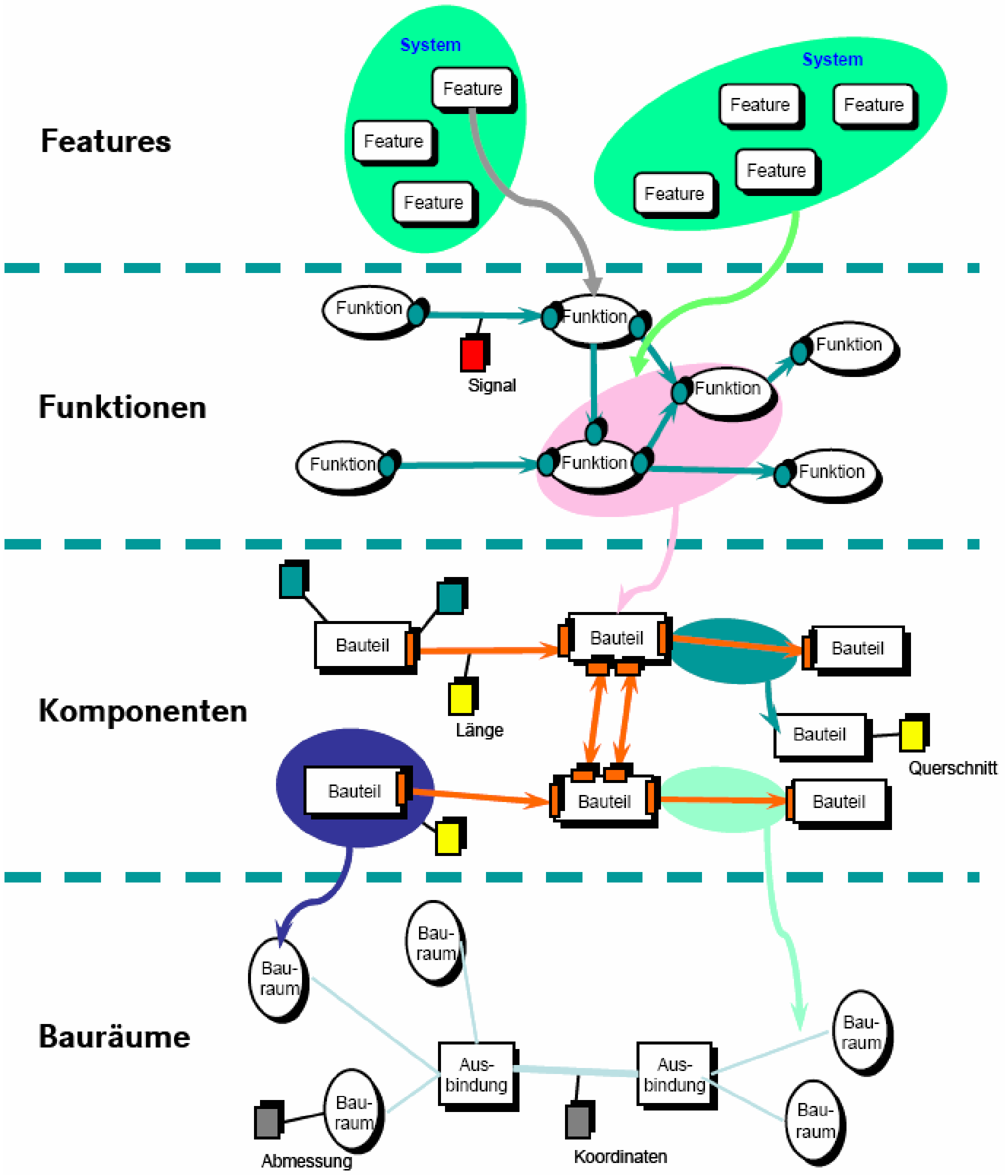
computation performance as well as communication performance

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

Abstraction Layers



Typical domain specific views

Features

Functions

Components

Component locations and wiring

Design space exploration needs domain specific metrics and parameters

⇨ **Electrics/Electronics Concept Tool**

Architectural Layers available within E/E meta model

Electronics

Electrics

Geometry
Physics

HW

SW

Functional Architecture
Functions and Subfunctions
Interaction

Software Architecture
Software Structure
Company Standards

Networking Architecture
Communication of ECU
Performance

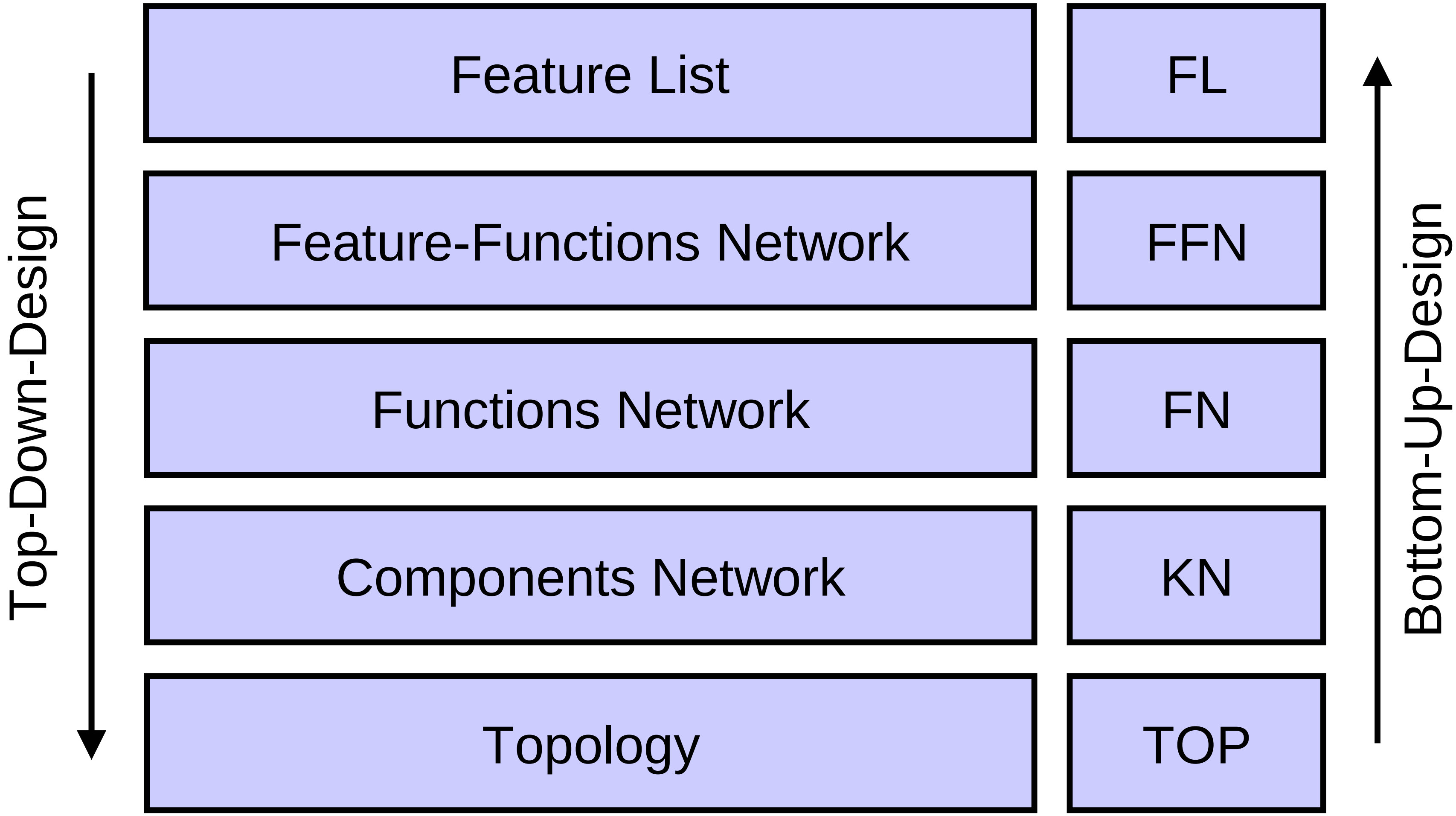
Component Architecture
ECUs, Variants,
Performance, Memory etc.

Power Supply
Electrical Power Supply and distribution
Generator, Electrical load

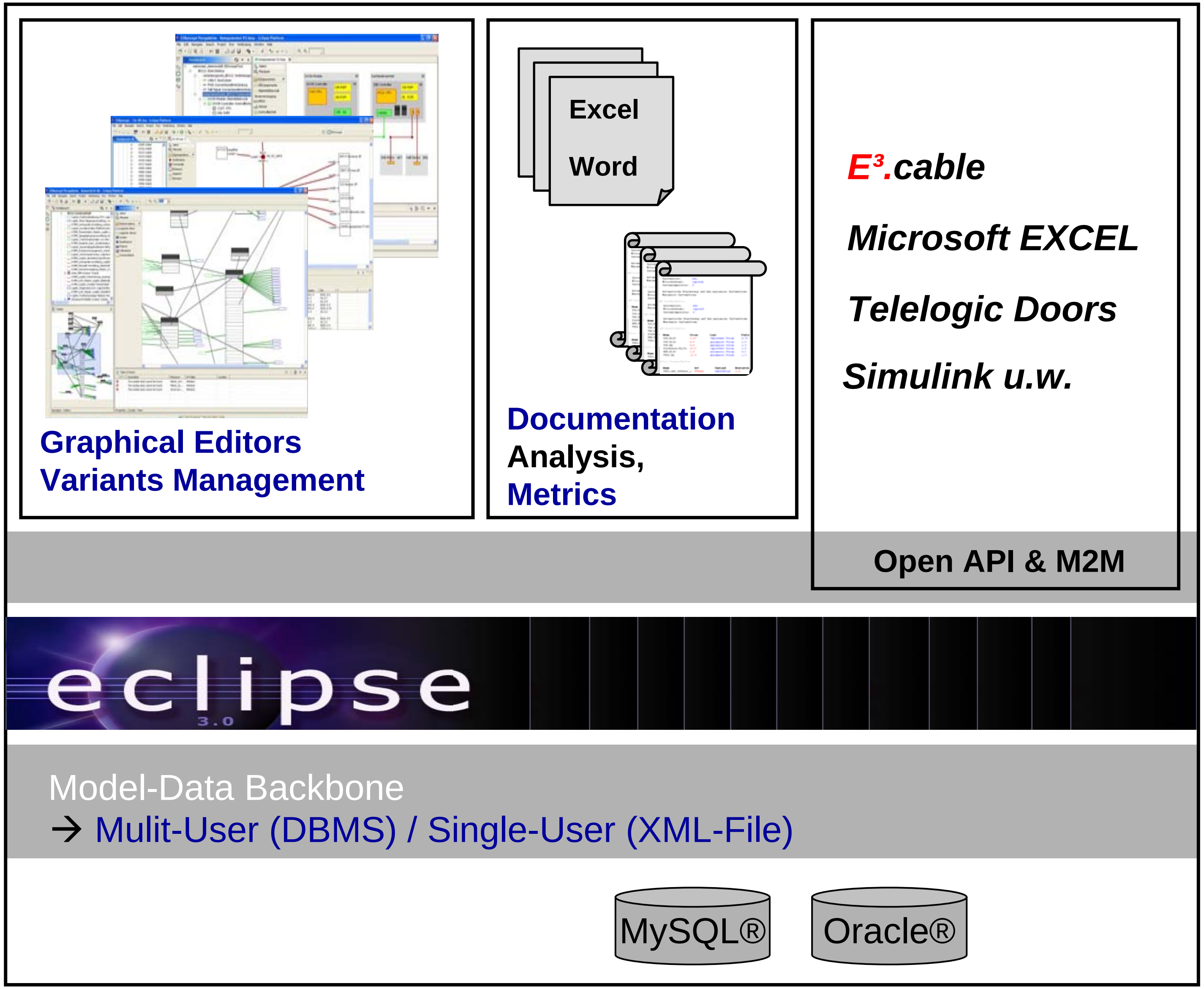
Component Topology
Location of the ECUs
Construction, Maintenance, Recycling

Interfaces
Mappings
Processes
Relationships

Abstraction Layers of Concept Tool



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



Tool-Framework for Development using Eclipse-Basis
- Extendability
- 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 Design Tools

Some general remarks

- **System Level vs. Component Level**
- **Meet-in-the-Middle Strategy**
- **Model Based Design**
- **Model to Model Transformation**
- **Test**

System Design: Meet-in-the-Middle - Strategy

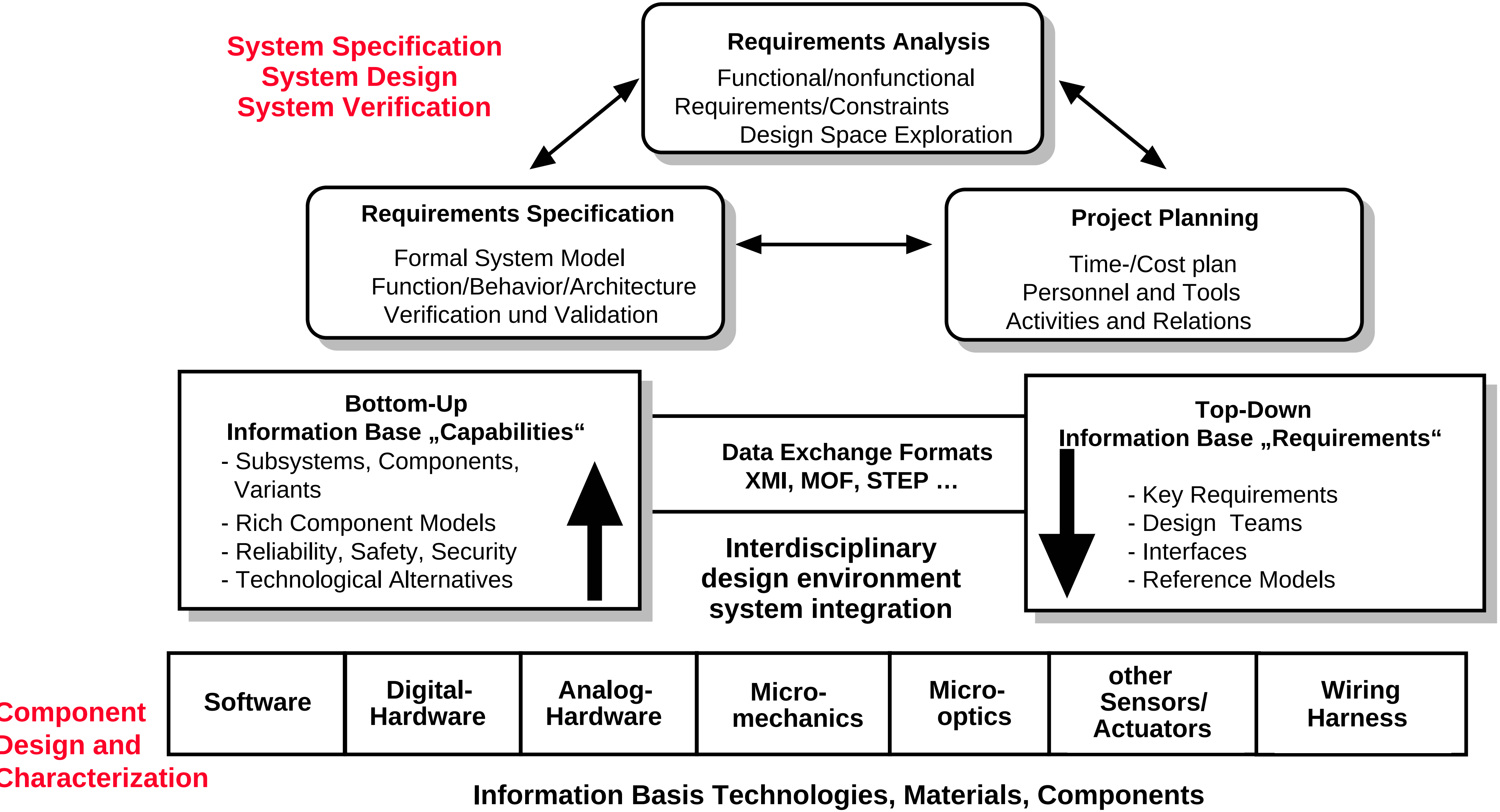
Design-Environment for System Level

- CAE-tools for Systems Engineers
- Evaluate Design Alternatives
- Architectural Synthesis
- High Design Productivity and Efficiency
- Short Development Cycles
- Mostly Manufacturer- and Technology Independent

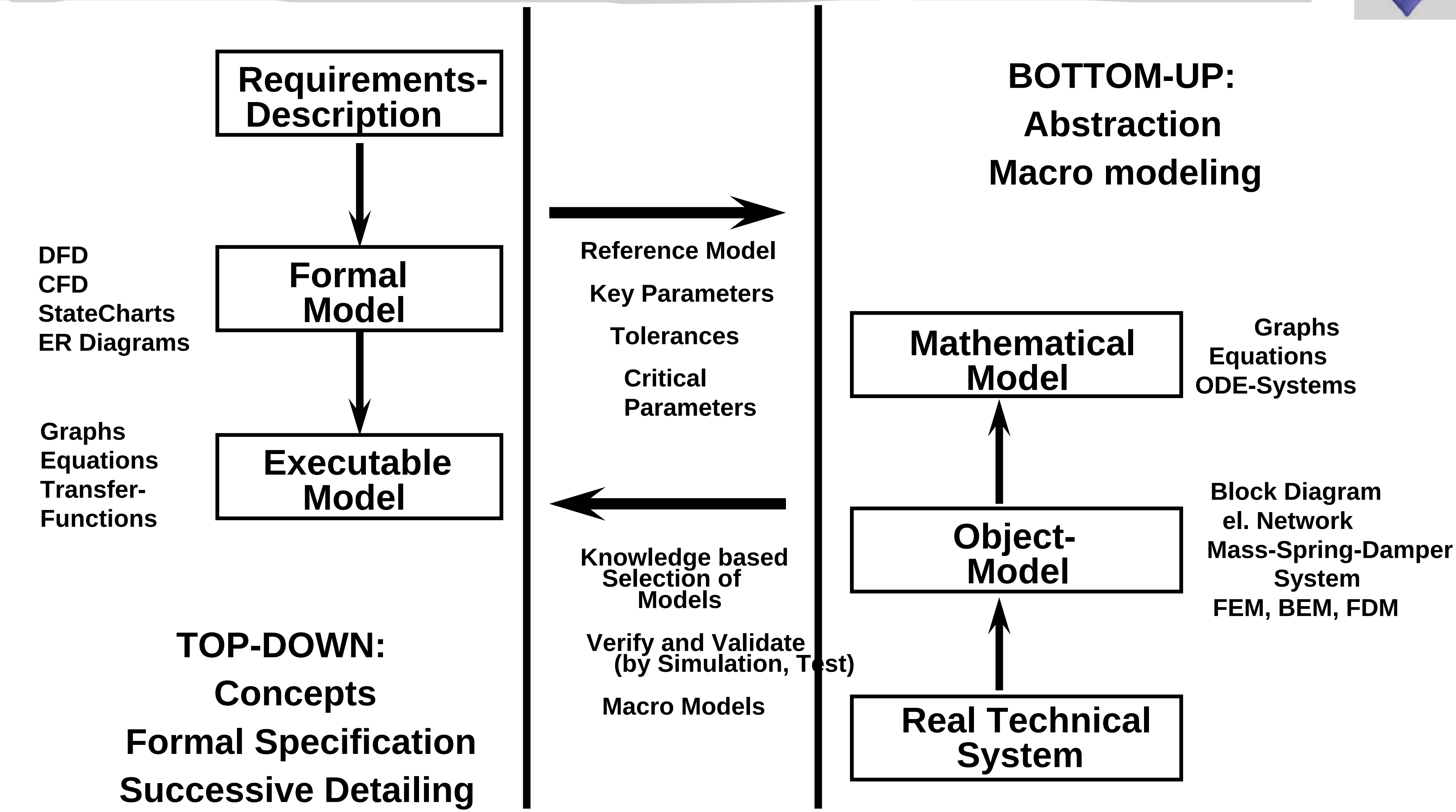
Design-Environment for System-components

- Closely related to Technology
- Mostly Manufacturer- and Technology Dependent
- CA-Tools dedicated and highly flexible
- For highly qualified Specialists only
- Optimization and Characterization of all Parameters
- Result: quality-assured Library for Variant Design

"Meet-In-The-Middle" - Strategy

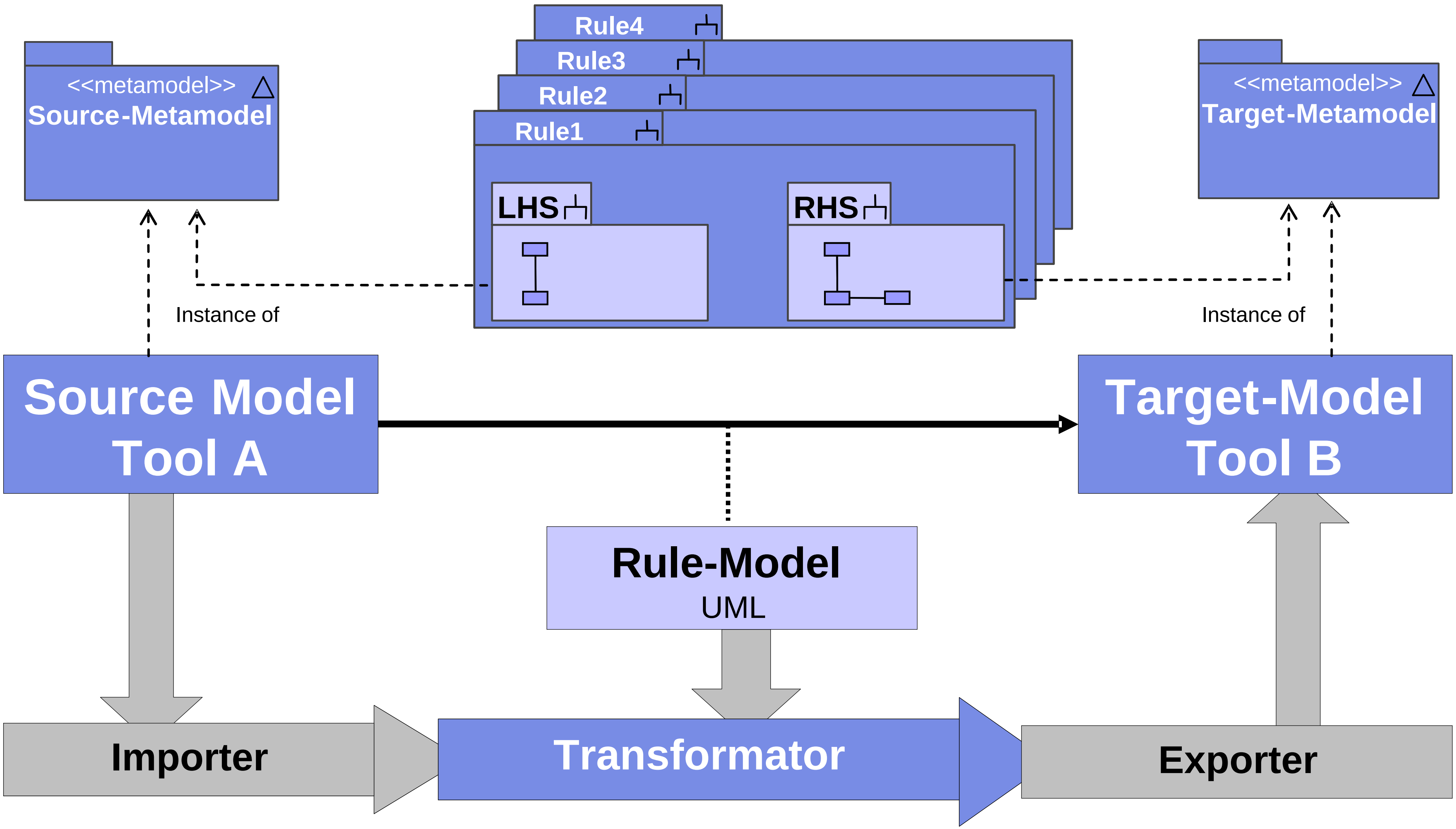


Model Based Design: Meet-in-the-Middle Strategy

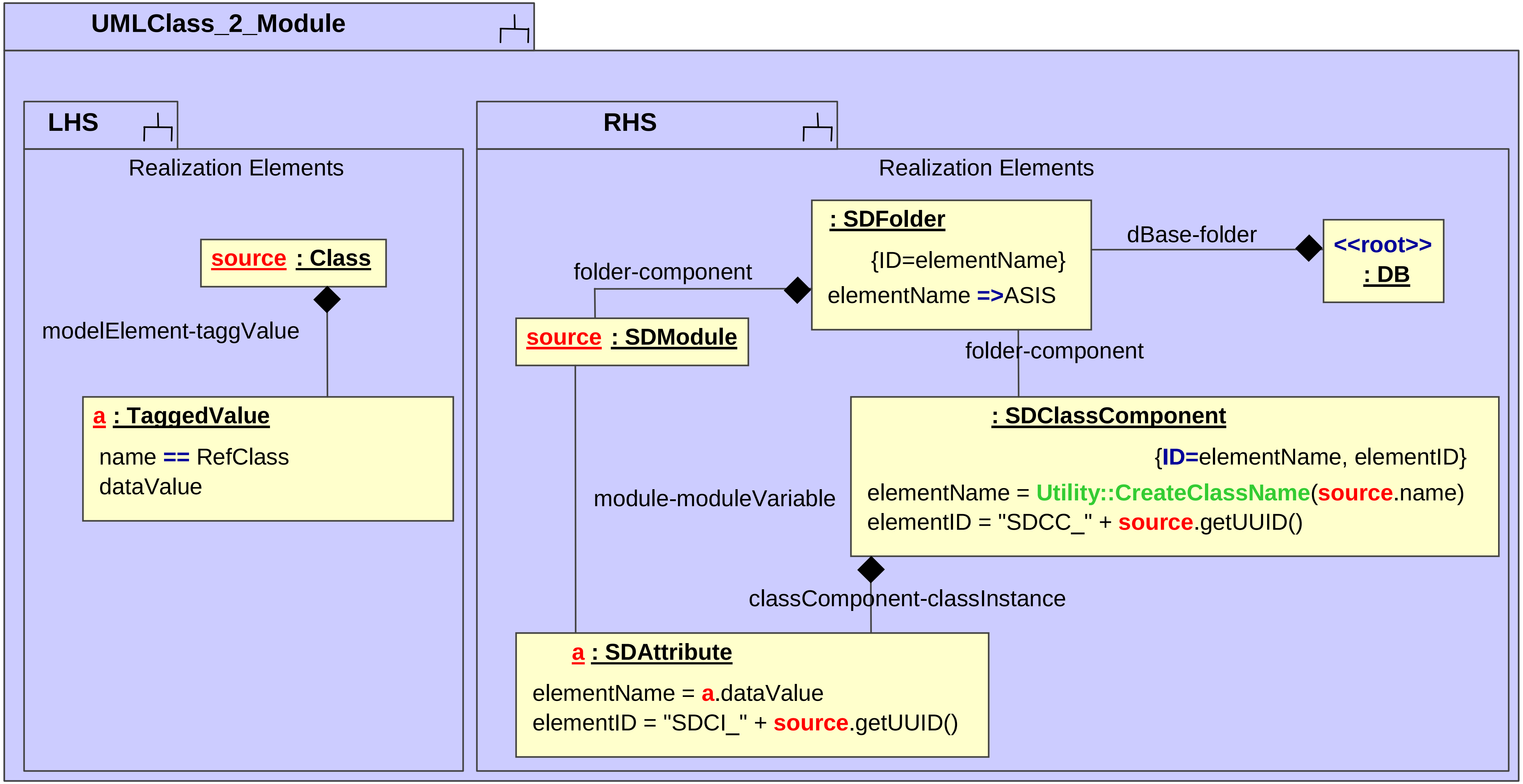


Werner Damm (OFFIS Oldenburg): Rich Component Models
Powerful Model Transformation Technology required

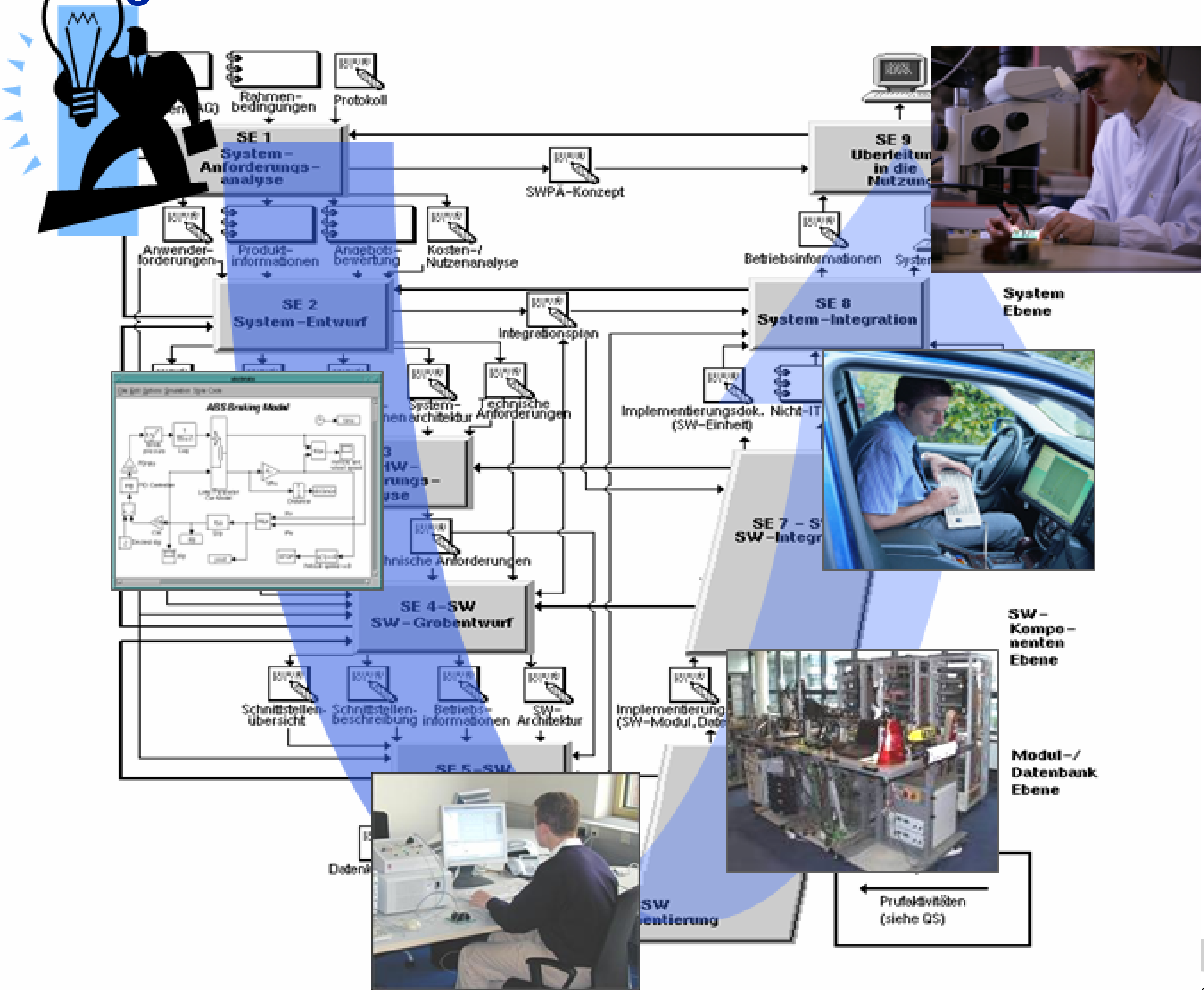
Structure of M2M transformation



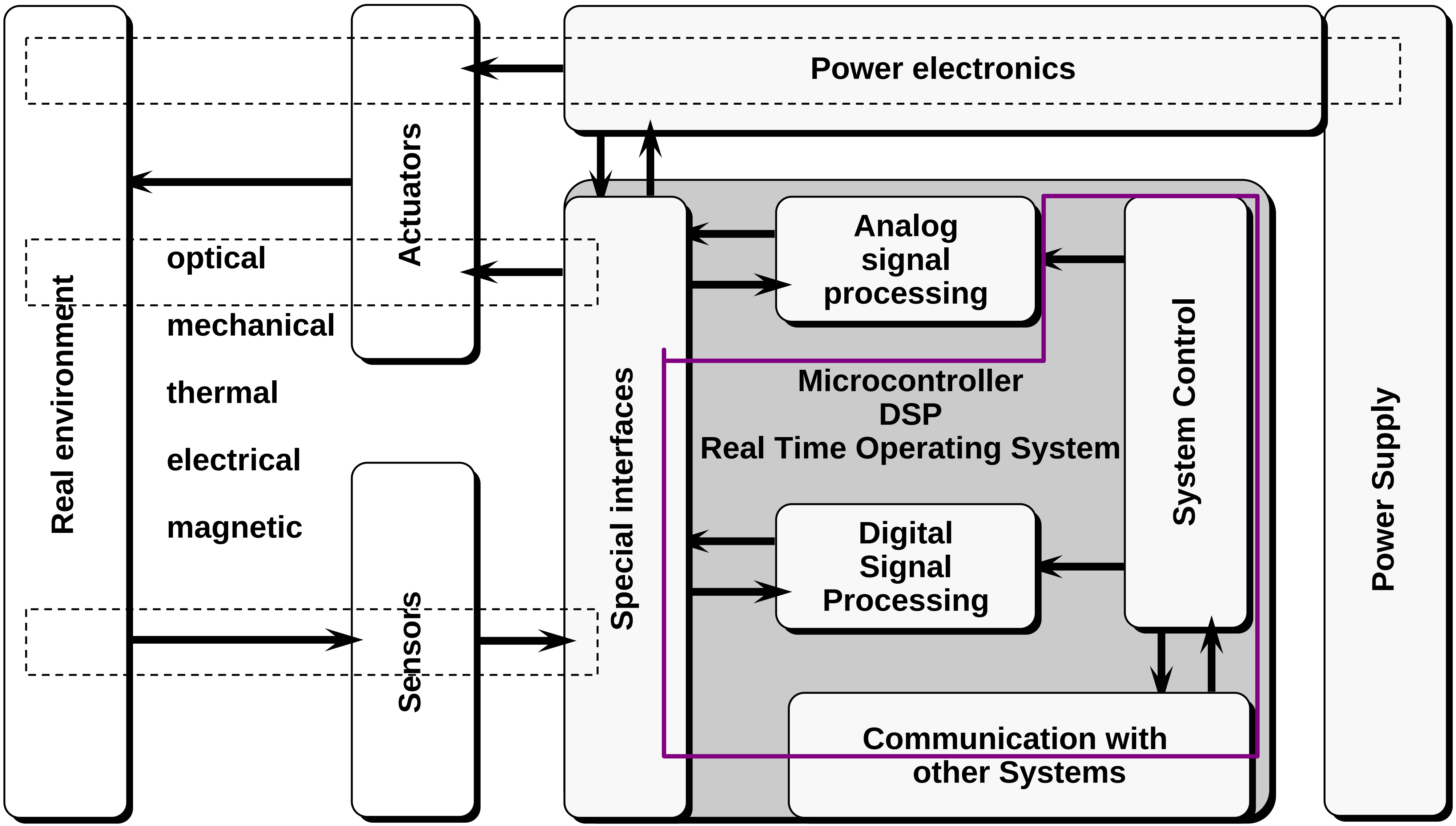
Transformation Rule specified in UML



Supporting test methods and tools



System Level Tool Support



Not seamless

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

 Benz

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
    (Early Design Space Exploration, Analysis, Design,
    Verification, Integration, Validation, Test, Calibration, Diagnosis)
  - Reviews, Rule Checking, Simulation, Formal Verification, Model Checking, Test
  - 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 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
  - Certification

Model based system design according to „Meet in the Middle – Strategy“ is possible,
but there are many design and analysis steps still missing, especially in early design phases.

Conclusion (3)

Industry / Academic Cooperation:

- **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, B, 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: cooperation between system manufacturer, (tier 1) suppliers, EDA companies and academics**

Questions

Thank you very much

for your attention