FZI
Forschungszentrum Informatik
an der Universität Karlsruhe

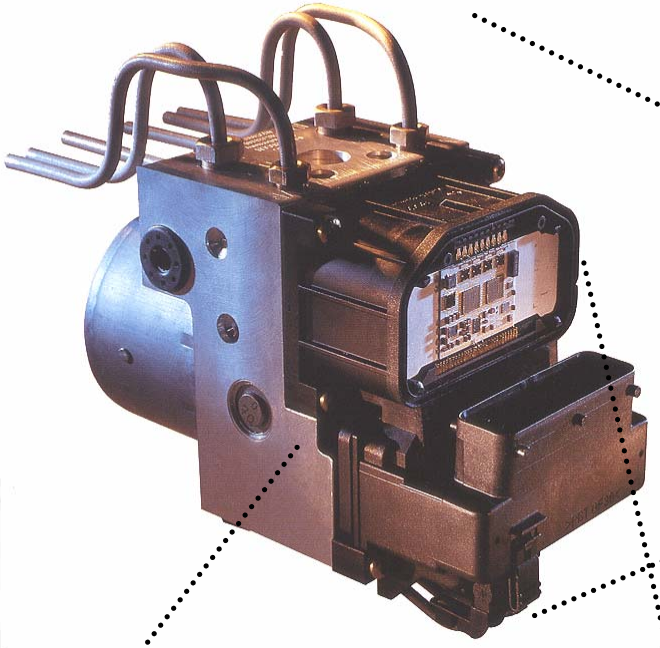
HLSLA‘07, 14-January-2007

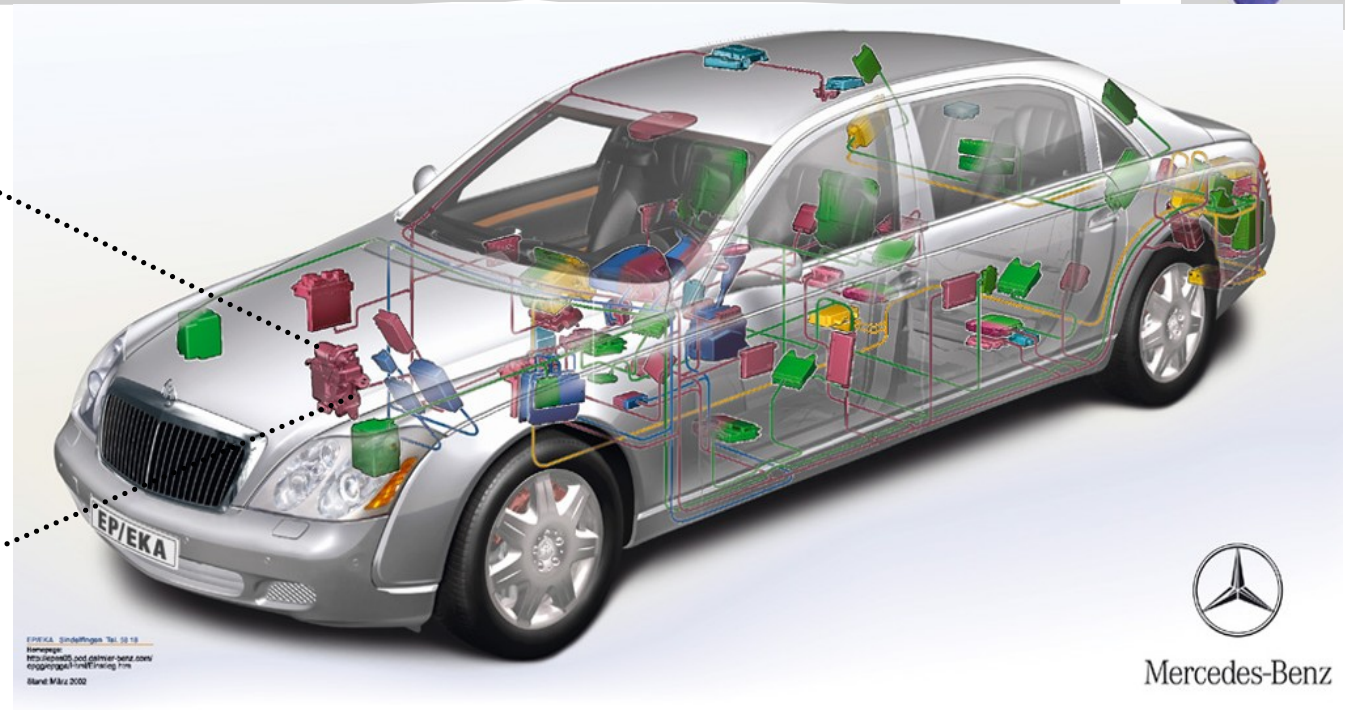
„Supporting system level design of distributed electronic control units for automotive applications“

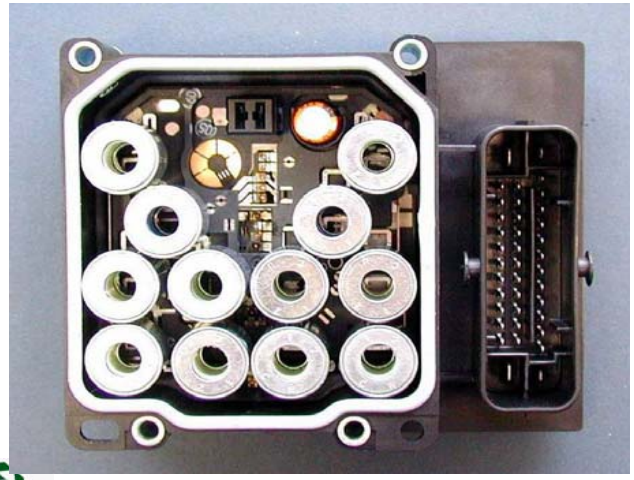
Prof. Dr.-Ing. Klaus D. Müller-Glaser
Universität Karlsruhe, Institut für Technik der Informationsverarbeitung (ITIV)

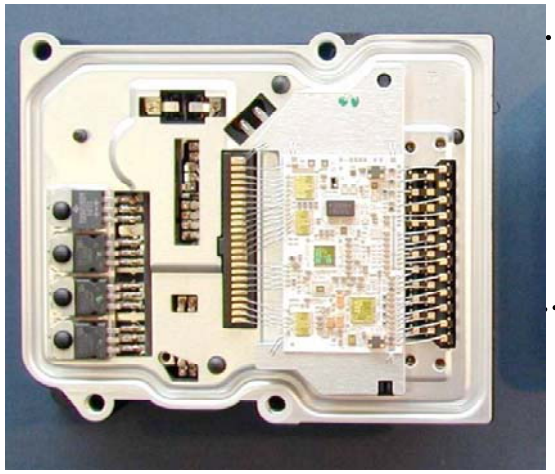
Dr.-Ing. Clemens Reichmann, Markus Kuehl
Aquintos GmbH

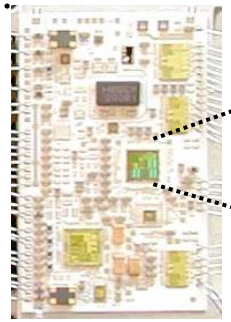
Characteristics: distributed, mechatronic, hard real time







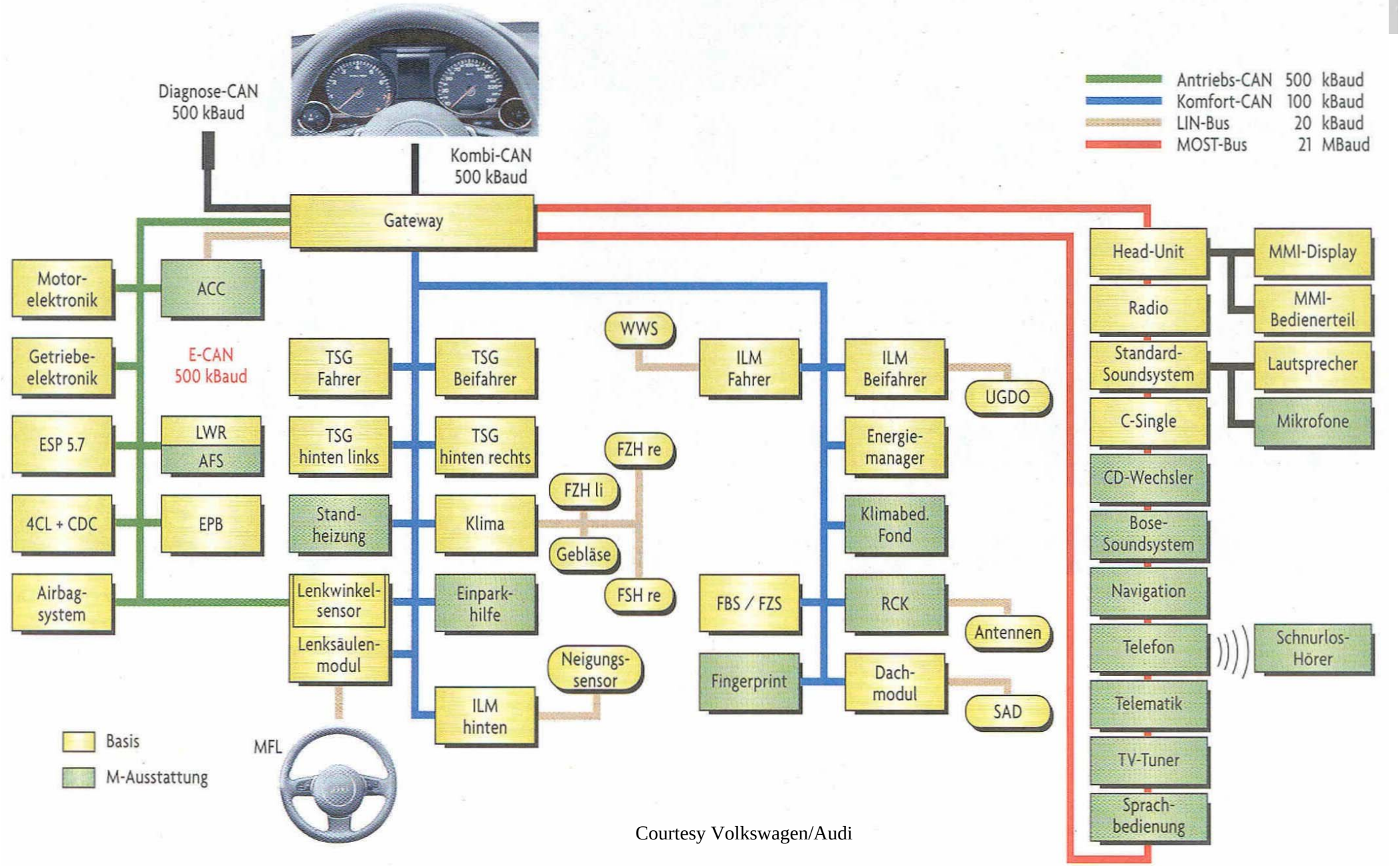




Software is part of ECU

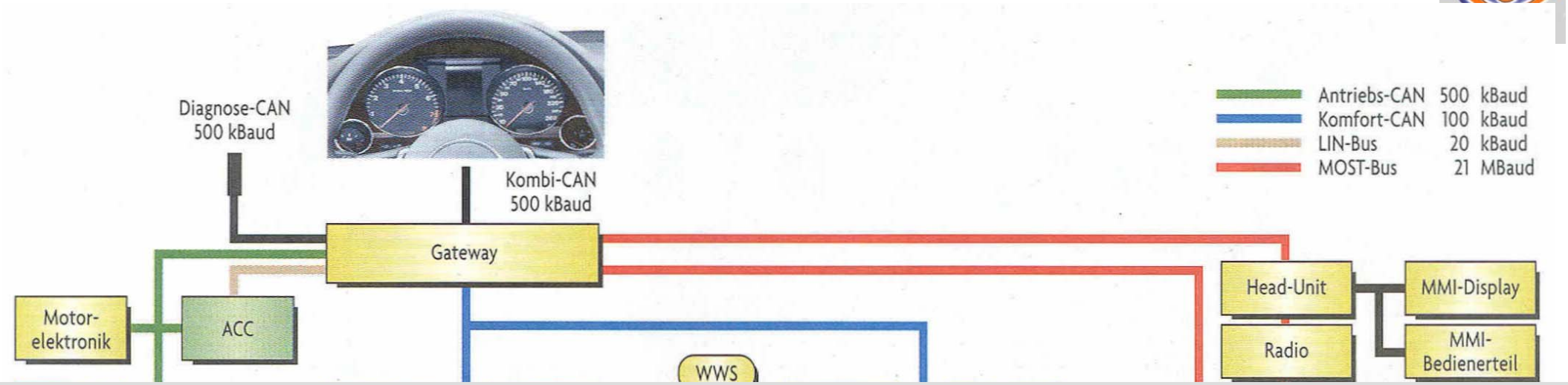
ECU's are part of mechatronic systems for measurement and control

Complex Communication (e.g. Audi A8)



Courtesy Volkswagen/Audi

Complex Communication (e.g. Audi A8)

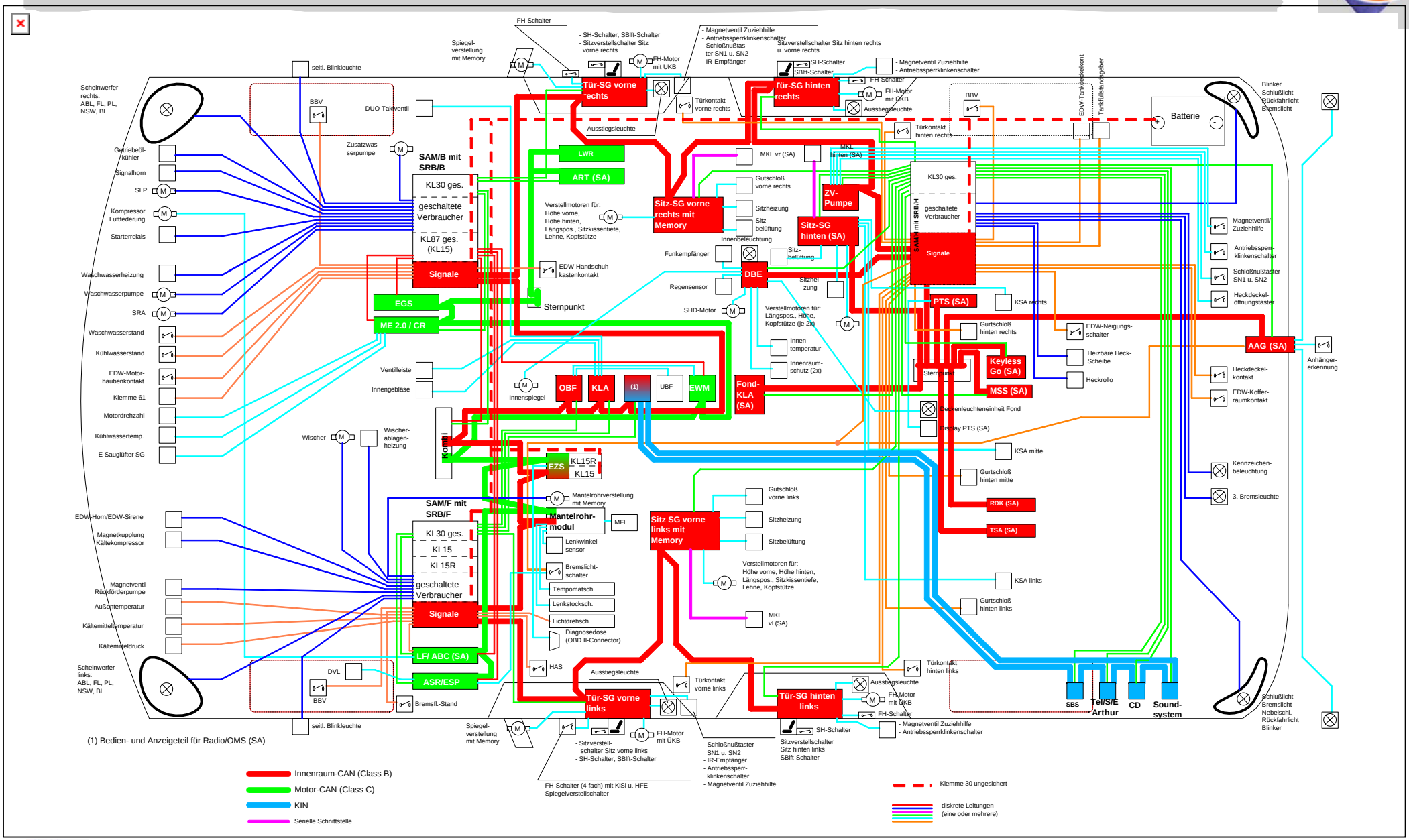


4 application domains for ECUs:

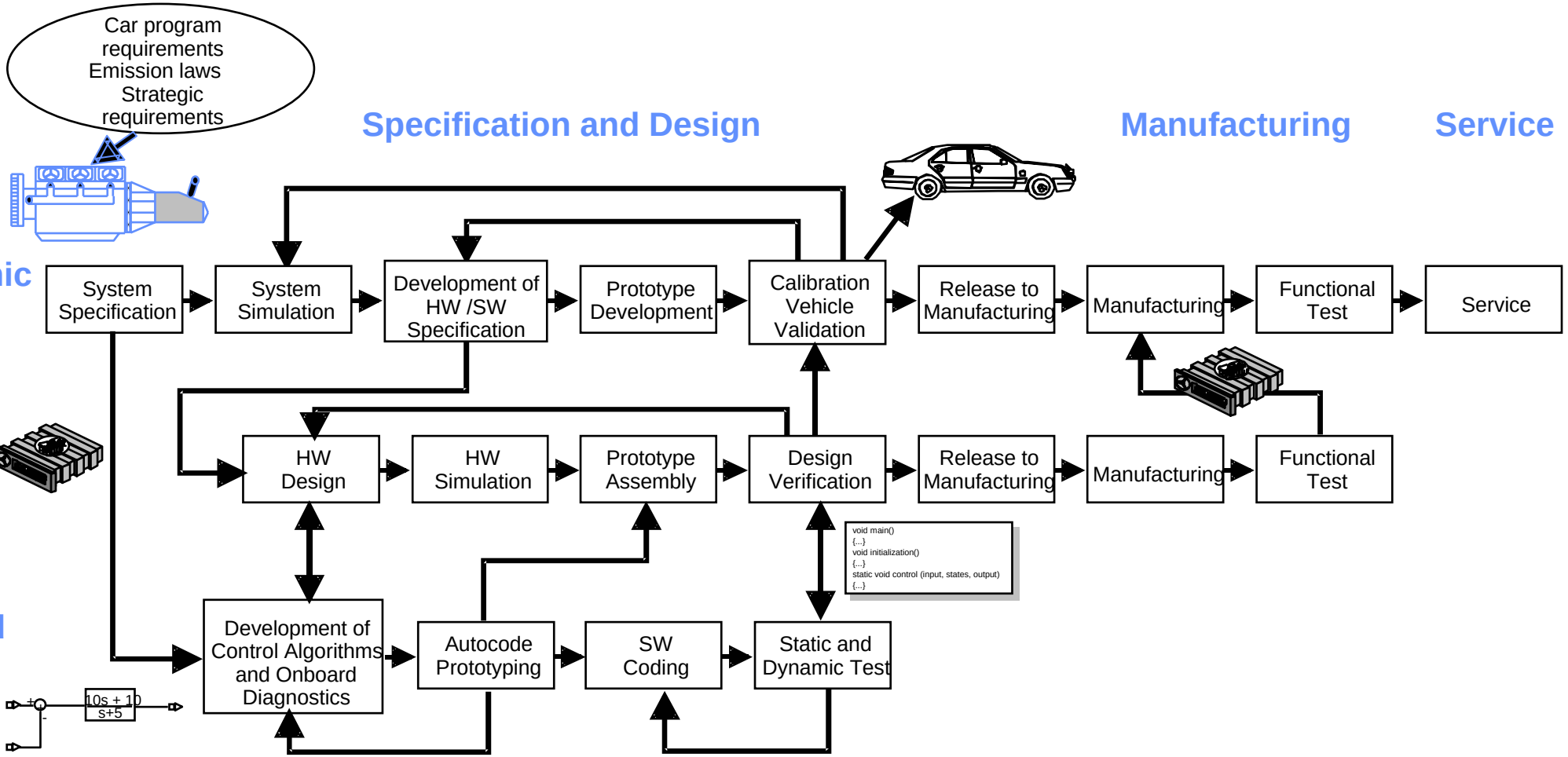
Power train:	mainly closed loop control functions
Chassis control:	mainly closed loop control functions
Body electronics:	mainly reactive, event driven functions
Infotainment:	mainly reactive, event driven functions

software intensive >>100k LOC

Electrics/Electronics in a car



Hierarchical Organization of Design Processes



Hierarchical Organization of Design Processes

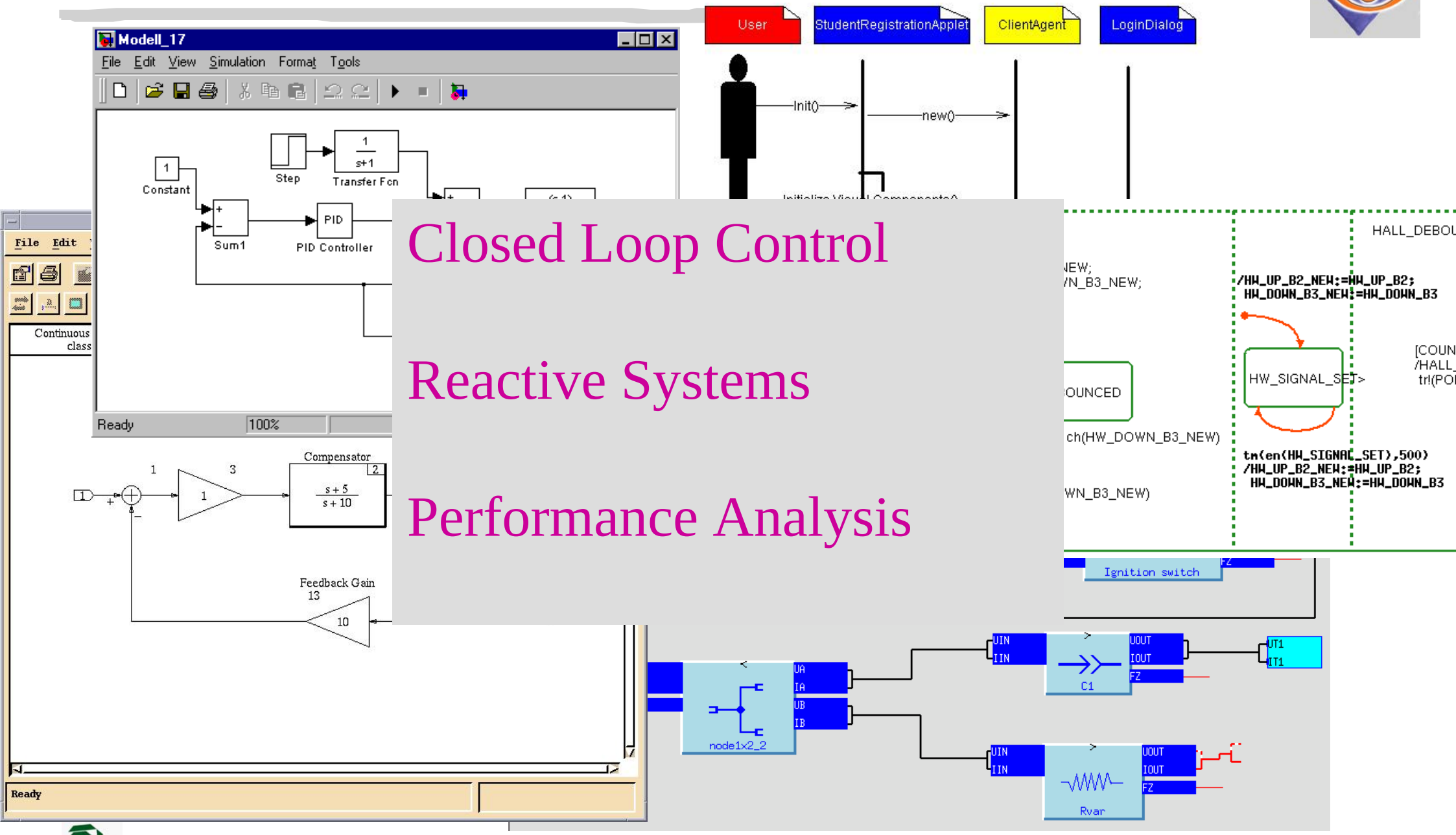
Specification and Design Manufacturing Service

multiple
interleaving design processes

Concurrent Engineering
distributed between OEM and supplier

requires
strictly controlled design methodology
supporting computer aided design tools
model based design

Model Based Design - graphical descriptions preferred



Closed Loop Control

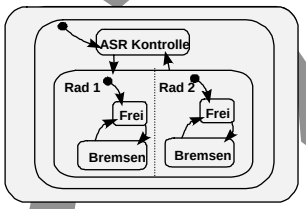
Reactive Systems

Performance Analysis

Typical Design Flow

Idea

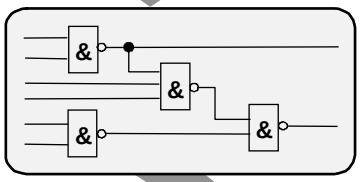
System-Analysis
executable Specs
System Design
model based
Customer Requirements
Technical Requirements
Real Time Requirements
System Architecture
Simulation, Verification



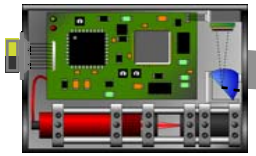
HW/SW-Requirements Analysis
Preliminary HW/SW-Design
HW-Architecture, SW-Architecture
Interface Description

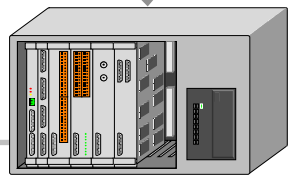
```
PROCESS (schlupf, state)
BEGIN
  CASE state IS
    WHEN freilauf =>
      IF schlupf > 0 THEN
        next_state <=
bremsen;
      ELSE
```

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

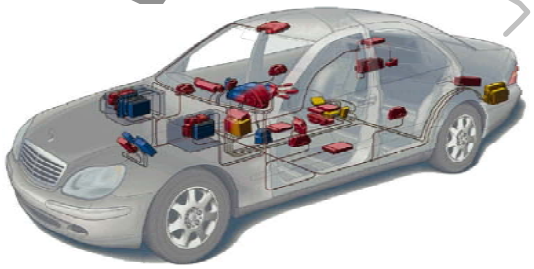


Detailed HW/SW-Design
SW-Design, Data Dictionary
HW-Drawings
HW-Analysis Report



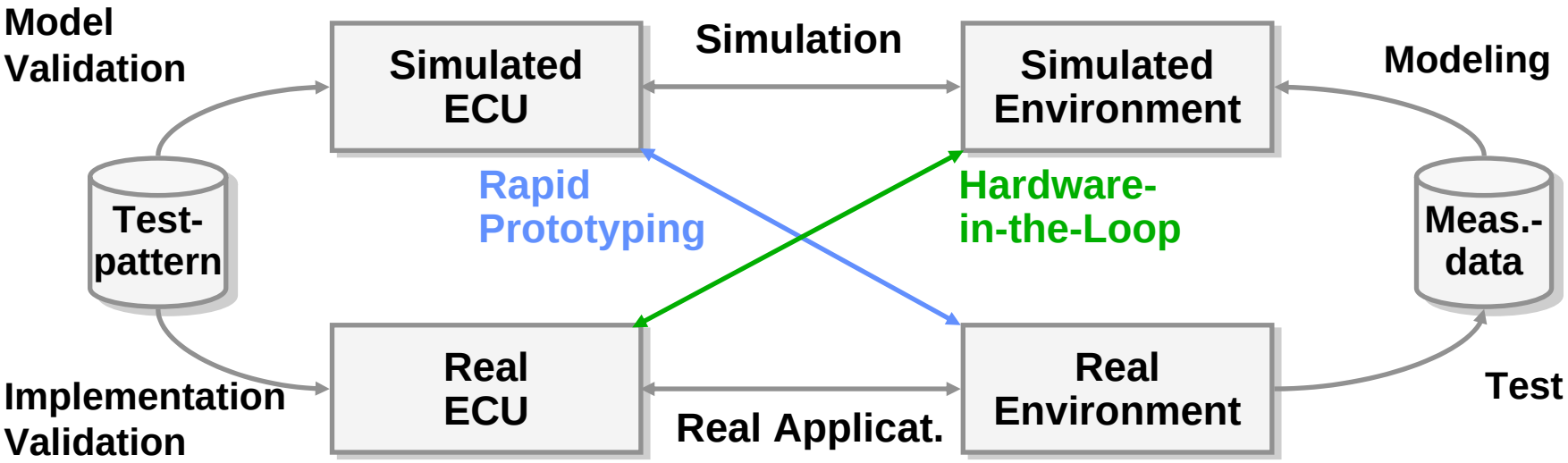


HW/SW-Implementation
Integration
W-Modules, Data Dictionary, SW-Comp
HW-Component, HW-Module
HW-Realization Documents



System Integration
Calibration, Application
Transition to Utilization

MiL, SiL, RP, HiL, Test



MiL Model-in-the-Loop

SiL Software-in-the-loop

RP Rapid Prototyping

HiL Hardware-in-the-Loop

Test

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

FZI

Challenges for Modeling

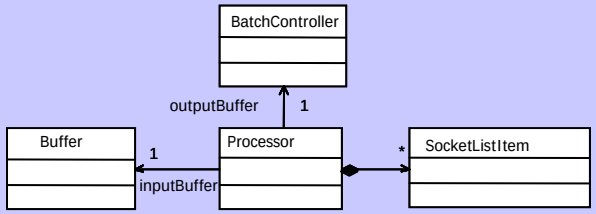
Domain Specific Modeling Languages
Model Synthesis
Model Validation
Model Transformation
Executable Models
Automatic Generation of Product Artefacts

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

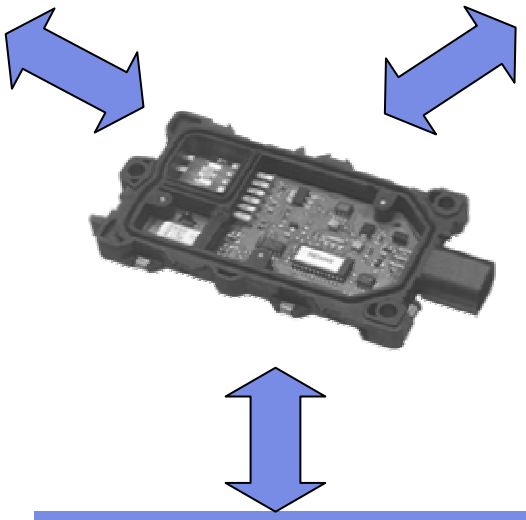
Generic Modeling-Platforms

Modeling for heterogeneous electronic embedded systems

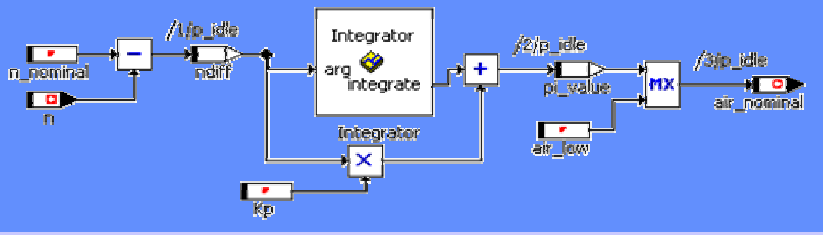
Architecture and Software
Modeling with UML



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

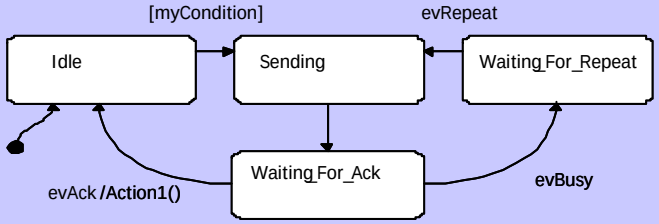


Signal flow oriented
Modeling with block diagrams



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

Event driven
Modeling 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)

ARTiSAN | Together | Poseidon | **Statemate**

AONIX | MagicDraw | Rhapsody | Rose

XMI 1.0, XMI 1.1, XMI 1.2

Model Data
MySQL, ORACLE

SQL

MATLAB Simulink — **MDL**

MATLAB Stateflow — **MDL**

GeneralStore CASE-Tool Integration Platform

XMI	JAVA	JAVA	JAVA	JAVA	XMI2XX	JAVA
Template UML Coder	Matlab Wrapper Generator	Matlab Automation	Statemate Wrapper Generator	Rhapsody in MicroC automation	other commercial code generators	
		DCOM		XMI	CG-Adaptor	
		Matlab Embedded Coder		MicroC	... generator	
C++		C		C	C/C++	

Compiler / Linker / Make

Target platforms (RTOS)

new challenge ➡

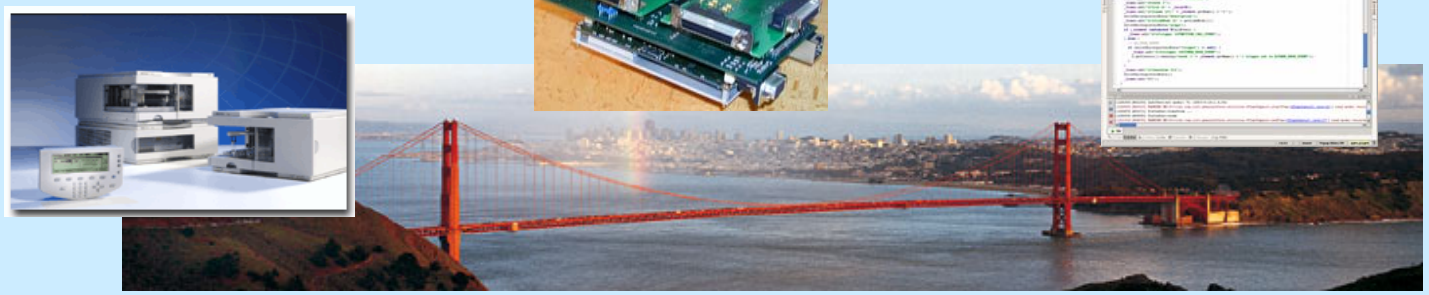
Target-Monitoring
Model Debugging
Test

Meta-Modeling 4 Abstraction Layers (OMG standard)

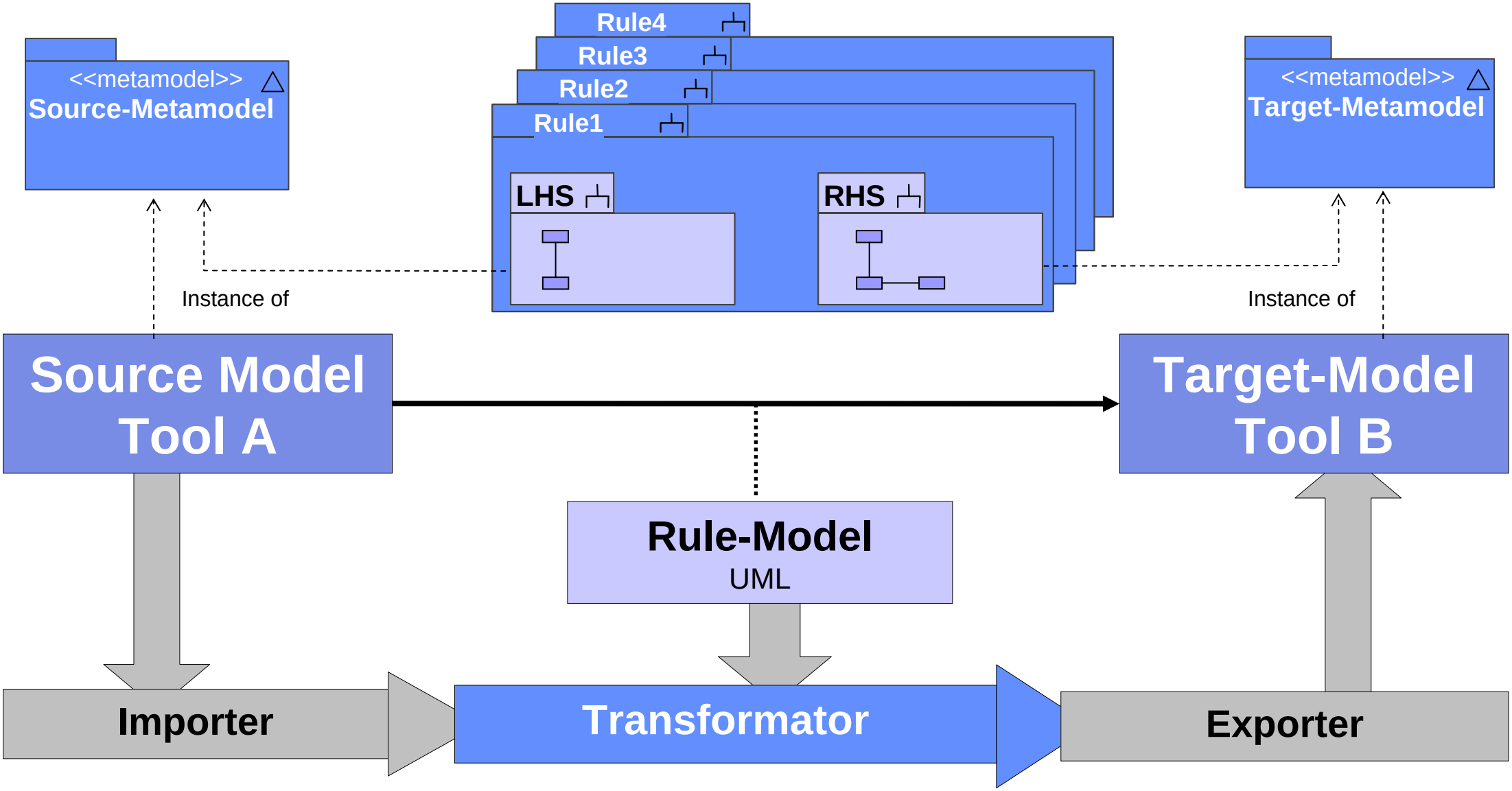
abstract

real

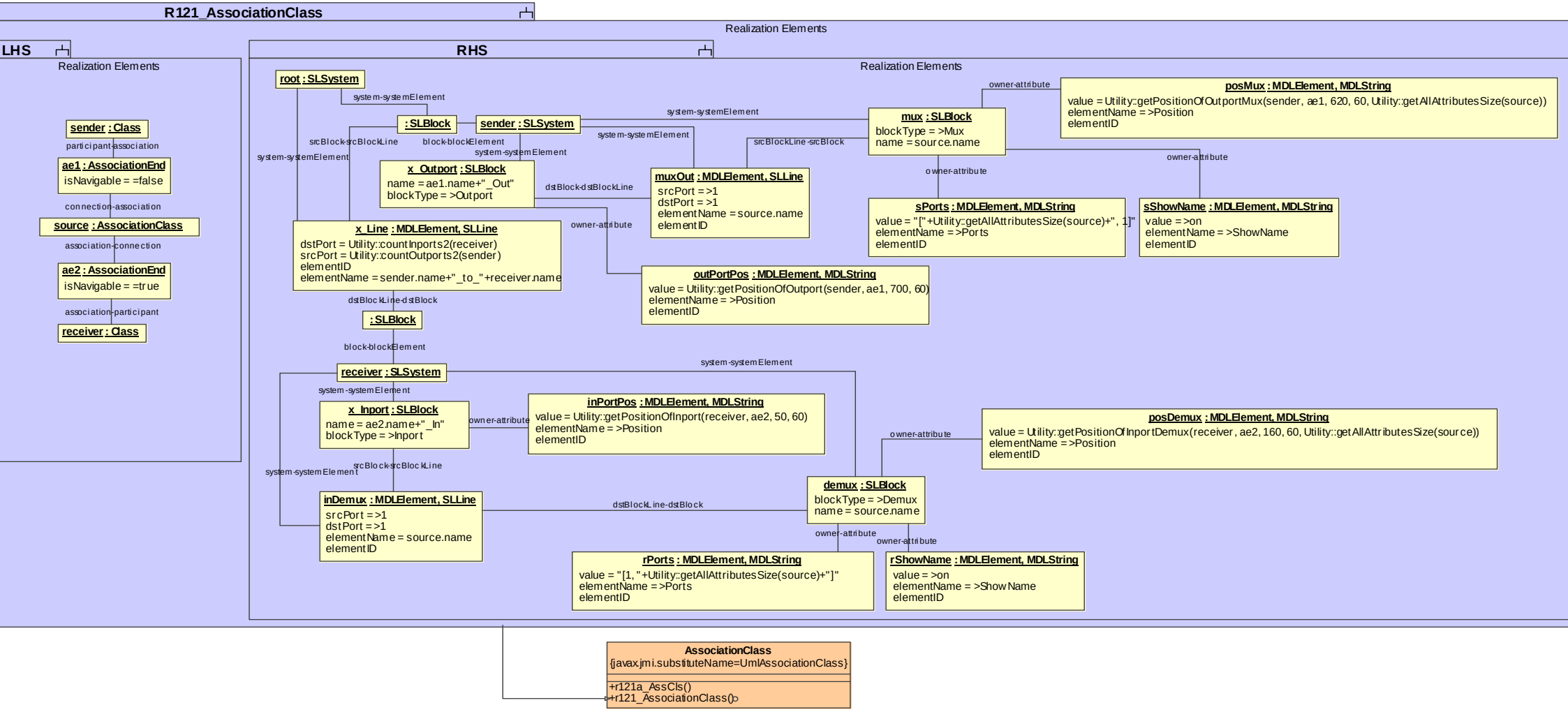
M3 layer		**MOF**		
M2 layer	**UML 1.4**	**UML 1.5**	**MATLAB Simulink**	**Statechart (D. Harel)**
M1 layer		**UML Model**	**Simulink Model**	**Statechart Model**
M0 layer	**Objects**	**Data**	**Source code**	
Real artefacts				



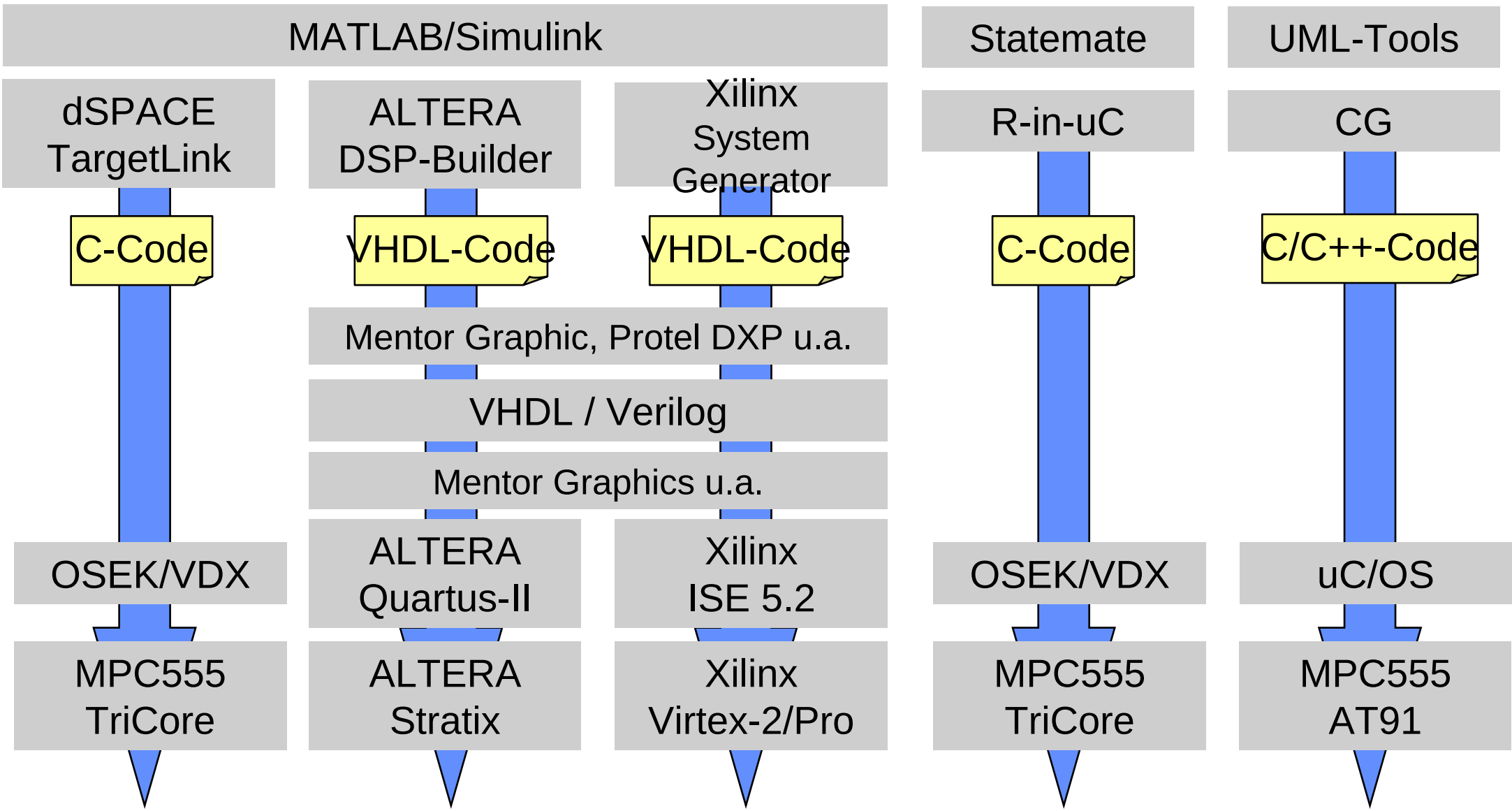
Model Transformation: M2M Engines Architecture



Transformation Rules UML -> Simulink



Tools Chains used at ITIV/FZI



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)

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

FZI

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

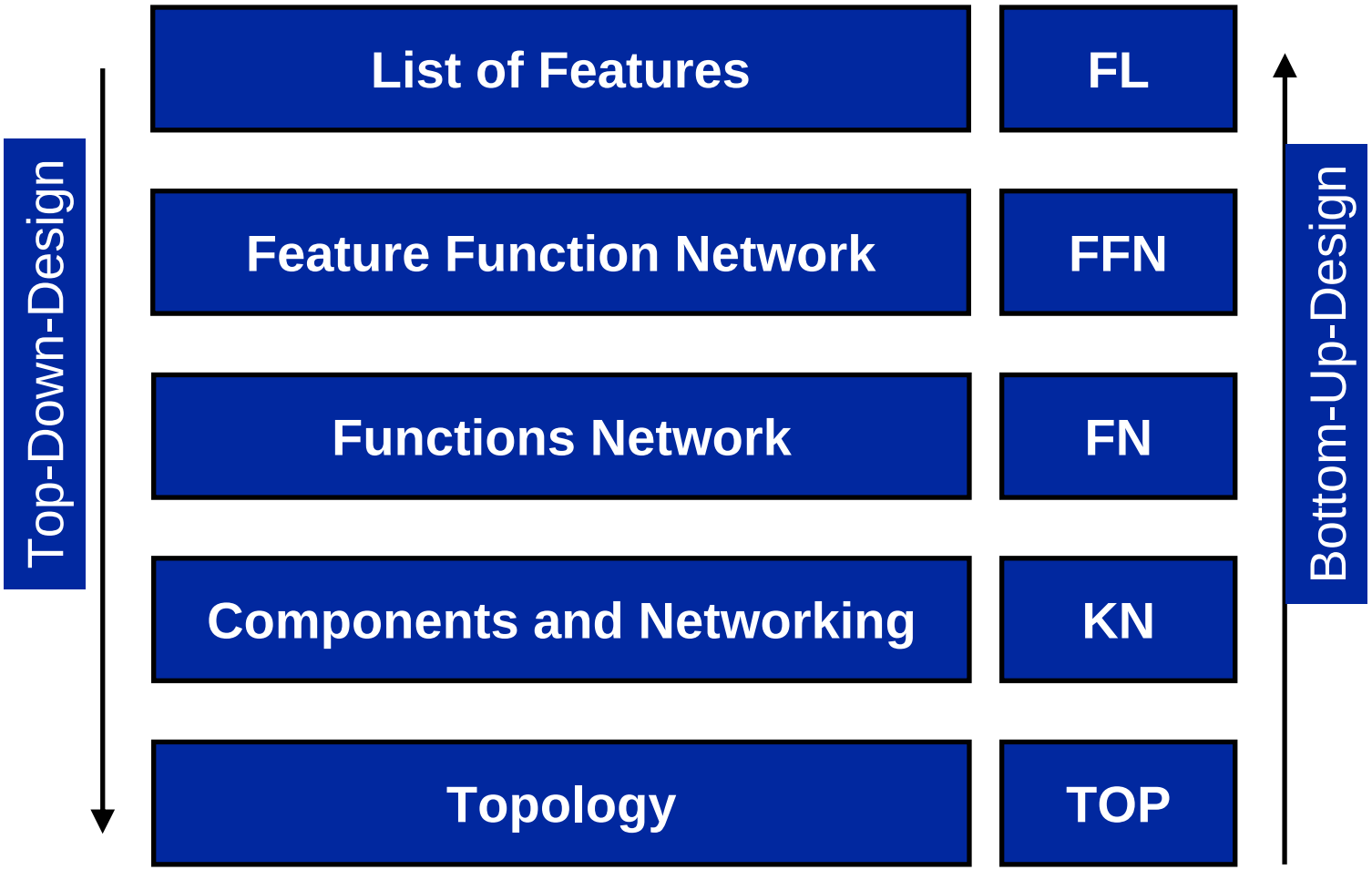
therefore, domain specific system level tools are required

interfacing seamlessly with component specific tools (meet in the middle).

A lot of model transformations are required

FZI

Abstraction Layers



Which features?

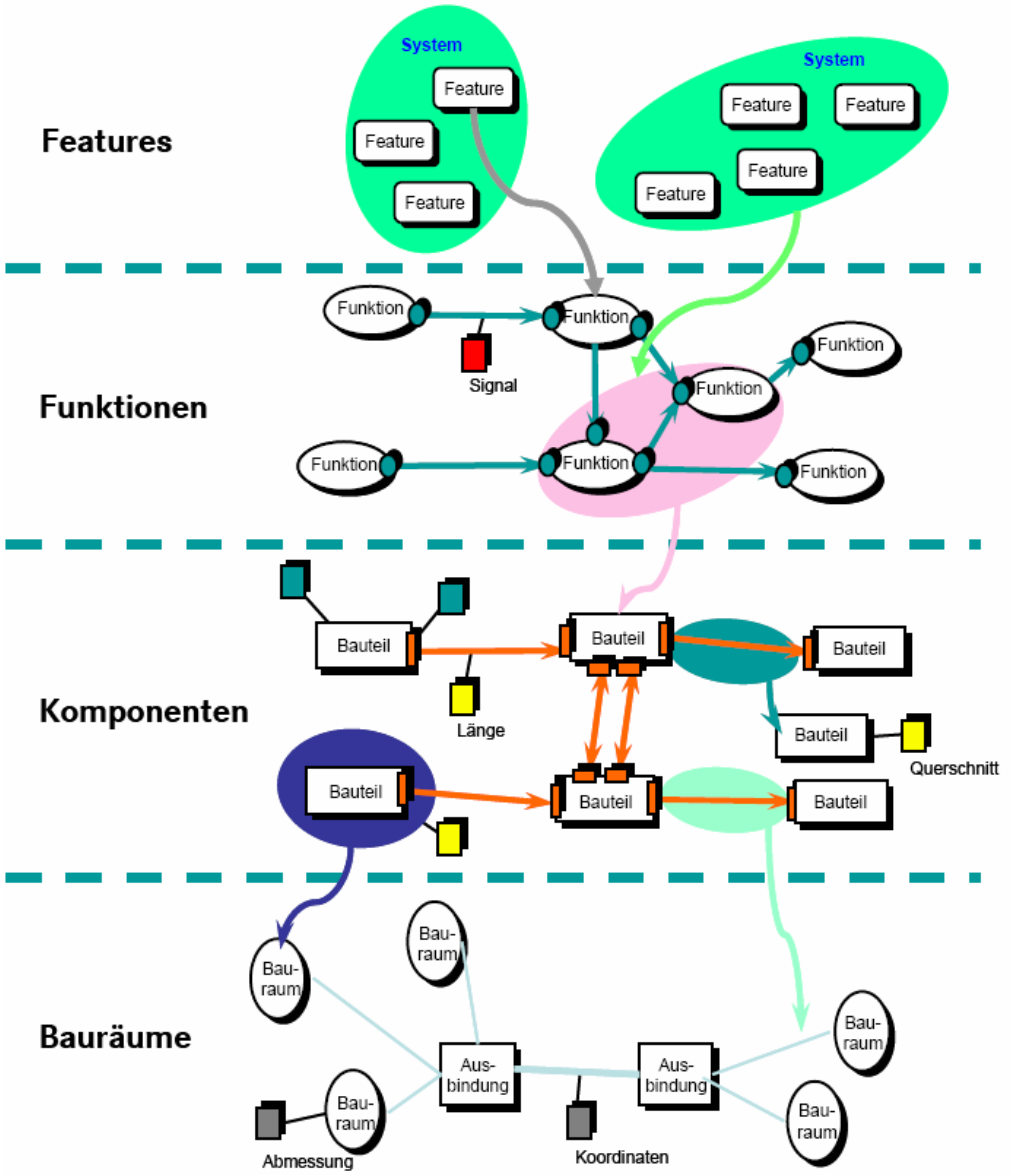
What is the concept behind the features? How do they interact?

Detailed Specification of Functions architecture

Comp. Architecture
Network Infrastructure
Power Distribution

Topology
wiring harness

Abstraction Layers



Typical domain specific views

Features

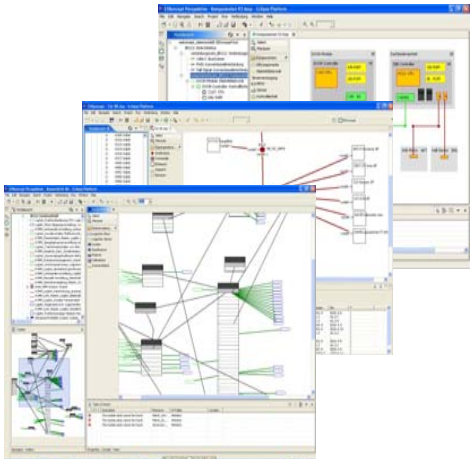
Functions

Components

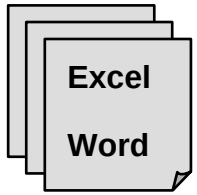
Component locations and wiring

Design space exploration needs domain specific metrics and parameters

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



Graphical Editors
Variants Management



Documentation
Analysis,
Metrics

E³.cable

Microsoft EXCEL

Telelogic Doors

Simulink u.w.

Open API & M2M

eclipse 3.0

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

MySQL®

Oracle®

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

FZI

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

Conclusion (3)

Model based system design is possible and very promising,

tools like Matlab/Simulink/Stateflow are essential

however, there are many design and analysis steps still missing, especially in early system design phases:

- design space exploration
- heterogenous system level modeling and simulation
- model checking, abstract interpretation
- code generation for RP and for production
- test program generation
- connection to 3D mechanical design (mechatronic)
- connection to life cycle product data management systems

Questions

Thank you very much

for your attention