"Challenges for reliable software design in automotive electronic control units"

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

Characteristics of Automotive Electronic Control Units (ECU)

State of the art in ECU design
- Typical Design Flow, V-Model
- Manufacturer Supplier Relationship

Model Based Design
- Heterogeneous models
- CASE tool integration platform
- Tool chains

ECU Design Challenges
- Complexity, Flexibility
- Open Systems and Standards
- Software Redistribution
- New System Level Design Tools

Conclusions
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
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

- **Microcontroller**
- **DSP**
- **Real Time Operating System**
- **Communication with other Systems**
- **Power Supply**
- **Actuators**
- **Sensors**
- **System Control**
- **Special interfaces**
- **Power electronics**
- **Analog signal processing**
- **Microcontroller**
  - DSP
  - Real Time Operating System
- **Digital Signal Processing**
- **Communication with other Systems**
- **Real environment**
  - optical
  - mechanical
  - thermal
  - electrical
  - magnetic
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
The State-of-the-Art DMU technology provides the basis for the mechanical integration and optimization of EE components (ECU’s, batteries, wiring harness, ...)

Mechanics/Electrics-CoDesign (Digital Mockup - DMU)
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: << 1 ppm/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
Automotive Electric/Electronic Systems

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

Car program requirements
Emission laws
Strategic requirements

Specification and Design

System Specification
System Simulation
Development of HW/SW Specification
Prototype Development
Calibration Vehicle Validation
Release to Manufacturing
Manufacturing
Functional Test
Service

Mechatronic Vehicle System

Electronic Control Unit (HW)

Embedded Realtime Software

Development of Control Algorithms and Onboard Diagnostics
AutoCode Prototyping
SW Coding
Static and Dynamic Test

Design Verification
Release to Manufacturing
Manufacturing
Functional Test
Service

HW Design
HW Simulation
Prototype Assembly

Concurrent Engineering
distributed between OEM and Tier 1 Suppliers

Multiple interleaving design processes
Complex Manufacturer Supplier Relationship

Car manufacturer controls system design and system integration different „Business-Models“ for software and hardware development by tier 1 suppliers

<table>
<thead>
<tr>
<th>Requirement Specification</th>
<th>Function Analysis, Design &amp; Modelling</th>
<th>Logical System Integration</th>
<th>ECU Developm. Integration</th>
<th>Physical System Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW / HW Supplier</td>
<td>SW / HW Supplier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Supplier</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
V-Model

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

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
V-Model for automotive ECU’s

System oriented Process steps

Application Software oriented Process steps

ECU oriented Process steps

Courtesy ETAS
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

- Requirements: 42%
- Design: 28%
- Coding: 6%
- Unit Test: 5%
- Acceptance Test: 2%
- Maintenance: 7%
- Other: 6%
- Human: 6%
- Environment: 5%
- Data: 5%
- Interface: 6%
- Documentation: 2%

Sheldon, et al.
IEEE Software, July 1992

Relative cost to repair a defect at different lifecycle phases

OEM  Supplier  OEM
Typical Design Flow

**System Integration**
Calibration, Application
Transition to Utilization

**Typical Design Flow**

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

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

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

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

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

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

**Idea System-Analysis**
executable Specs
**Preliminary HW/SW-Design**
model based
Customer Requirements
Interface Description
Technical Requirements
Real Time Requirements
System Architecture
Simulation, Verification

**Interface Description**
Frei
Bremsen
Rad 1
ASR Kontrolle
Bremsen
Frei
Rad 2
ASR Kontrolle
Bremsen
Frei

**Customer Requirements**
**Technical Requirements**
**Real Time Requirements**

**Simulation, Verification**

**HW-Component, HW-Module**
**HW-Realization Documents**

**System Integration**
Calibration, Application
Transition to Utilization

**System Integration**
Calibration, Application
Transition to Utilization
ECU development for passenger cars: 3 Prototypes

- Requirements-Analysis
- System Specification
- System Design
- Subsystem Design
- Module Design
- Prototype
- System-Implementation
- Prototype
- Prototype
- Prototype
- System Test
- Subsystem Test
- Module Test
- Life Cycle-Analysis
- System Delivery
- concept-oriented Rapid Prototyping (A-Muster)
- architecture-oriented Rapid Prototyping (B-Muster)
- Implementation-oriented Rapid Prototyping (C-Muster)
# Verification and Validation

## Software Quality Control

### Verification

**Am I Building the Product Right?**

<table>
<thead>
<tr>
<th>Static Techniques</th>
<th>Review</th>
<th>White Box Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Walkthrough, Fagan Inspection, Peer Review, Argument etc.</td>
<td>Static Analysis, Formal Proof, Control and Data Flow, etc.</td>
</tr>
</tbody>
</table>

### Validation

**Am I Building the Right Product?**

<table>
<thead>
<tr>
<th>Animation</th>
<th>System/Acceptance Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formal Specification, CASE Modeling, Rapid Prototyping, Virtual Reality etc.</td>
<td>Functional Performance, Stress Testing etc.</td>
</tr>
</tbody>
</table>

### Dynamic - Module/Integration Test

<table>
<thead>
<tr>
<th>Black Box Test</th>
<th>White Box Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional Performance, Stress Testing etc.</td>
<td>Structural, Path, Branch, Condition Decision Coverage etc.</td>
</tr>
</tbody>
</table>
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
Modeling for heterogeneous electronic embedded systems

Architecture
Modelling with UML

- BatchController
- Buffer
- Processor

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)

**Model Data**
- MySQL, ORACLE
- SQL

**GeneralStore**
CASE-Tool Integration Platform

- MATLAB Simulink Stateflow
- MDL
- XMI
- Template UML Coder
- JAVA
  - Matlab Wrapper Generator
  - Matlab Embedded Coder
  - Matlab Automation
  - DCOM
- JAVA
  - Rhapsody in C automation
  - XMI
  - MicroC
- XMI2XX
  - other commercial code generators
  - CG-Adaptor
  - ... generator

**Compiler / Linker / Make**
Target platforms (RTOS)

**Target-Monitoring Model Debugging Test**

new challenge
<table>
<thead>
<tr>
<th>Layer</th>
<th>M3 layer</th>
<th>M2 layer</th>
<th>M1 layer</th>
<th>M0 layer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>abstract</td>
<td>UML 1.5</td>
<td>UML 2.0</td>
<td>MATLAB Simulink</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Simulink Model</td>
<td>Statechart Model</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Objects</td>
<td>Data</td>
<td>Source code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Real artefacts</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Copyright © 2007 FZI Karlsruhe kmg**
Tools Chains used at ITIV/FZI

MATLAB/Simulink
- dSPACE
  - TargetLink
- ALTERA
  - DSP-Builder
- VHDL-C ode
  - Mentor Graphics, Protel DXP u.a.
  - VHDL / Verilog
  - Mentor Graphics u.a.
- OSEK/VDX
  - MPC555
  - TriCore

Xilinx
- System Generator
- ISE 5.2
- Virtex-2/Pro

State mate
- R-in-uC
- C-Code

UML-Tools
- CG
- C/C++-Code

ALTERA
- Str atix
- Quartus-II

Xilinx
- Virtex-2/Pro
- System Generator

MPC555
- TriCore
- TargetLink

uC/OS
- AT91

MATLAB/Simulink
- dSPACE
  - TargetLink
- ALTERA
  - DSP-Builder
- VHDL-C ode
  - Mentor Graphics, Protel DXP u.a.
  - VHDL / Verilog
  - Mentor Graphics u.a.
- OSEK/VDX
  - MPC555
  - TriCore

Xilinx
- System Generator
- ISE 5.2
- Virtex-2/Pro

State mate
- R-in-uC
- C-Code

UML-Tools
- CG
- C/C++-Code

ALTERA
- Str atix
- Quartus-II

Xilinx
- Virtex-2/Pro
- System Generator

MPC555
- TriCore
- TargetLink

uC/OS
- AT91
ITIV/FZI Tool Chains (Automotive) Verification Support

MATLAB/Simulink

dSPACE
TargetLink

C-Code

OESEK/VDX

MPC555
TriCore

ETAS ASCET-SD

TIP

C-Code

OESEK/VDX

MPC555
TriCore

Statemate

R-in-uC

C-Code

OESEK/VDX

MPC555
TriCore

POLYSOURCE C-Verifier (MISRA-C, DO-178B)
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

Holistic Safety Approach

Passive Safety

Active Safety

Crash Probability

Crash

1. Normal Driving

2. Warning Systems

3. Assistance Systems

4. Automatic Safety Systems

5. Safety Systems

6. Safety Systems soft level

7. Safety Systems hard level

Safety Systems after Crash

Collision Avoidance

Pre-Crash Phase

Basic Vehicle Safety

Occupant Protection

<table>
<thead>
<tr>
<th>ADASE</th>
<th>ACC S&amp;G etc</th>
<th>Lane departure warning</th>
<th>Brake assistant</th>
<th>Emergency braking system, Collision avoidance</th>
<th>Pedestrian airbag</th>
<th>Crash severity sensing for ignition levels and belt tension</th>
<th>Emergency/ Mayday Systems</th>
</tr>
</thead>
</table>

Copyright © 2007 FZI Karlsruhe kmg
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?!)

Copyright © 2007 FZI Karlsruhe kmg
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 TMS470R1
Mitsubishi El. M32R
Dynamic reconfiguration of a FPGA module slot

Slot 0 Slot 1 Slot 2 Slot 3

Bus system

CAN-Anbindung

MicroBlaze (Runtime system)

Arbiter

ICAP/Decompressor
Dynamic reconfiguration of a FPGA module slot

Bus system

Slot 0 Slot 1 Slot 2 Slot 3

MicroBlaze (Runtime system)

CAN-Anbindung

Arbiter

ICAP/Decompressor
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)
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

OSEK RTOS

Application specific software

Standard software modules

Standard CS Middleware

Diagnosis module

CS Middleware

Transport layer

Network management

Communication layer

CAN Driver

Other Driver

Hardware (µC, I/O, CAN, other communication interfaces)
**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

 Courtesy ETAS GmbH
Goal (AUTOSAR)

Vehicle A
- Seat Adjustment A
- Seat Heating A
- Air Conditioning

Vehicle B
- Seat Adjustment B
- Seat Heating B
- Lighting

Hardware A
- ECU Library
- Mapping A
- Quality Assured Hardware Platforms Library

Hardware B
- ECU Library
- Mapping B

Function Library
- Seat Adjustment A
- Seat Adjustment B
- Seat Heating A
- Seat Heating B
- Lighting
- Air Conditioning

Goal:
- Automatic Code Generation
Challenge

Algorithm Integration
C, C++, Matlab
SDL, SPW
Cossap

Functional Network

Does the functionally integrated design work

Executable Functional Specification

mapping

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?
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 Implantation conforms to the architecture

Analyze lessons learned

Functional analysis
(behavioral, performance, 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 processes 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**
- **HW**
- **Electrical**
- **Physical Geometrical**

**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?
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
Abstraction layers of new EE-concept tool

- **Customer Requirements**
  - Features

- **Feature-Architecture**
  - Feature-Function Network Edit.
  - Feature-Function Type Builder

- **Functions-Architecture**
  - Function Network Editor
  - Function Type Builder
  - Signal Editor

- **Component-Network Architecture**
  - Network Editor
  - Component Editor
  - Power Distribution Editor
  - Demand View
  - Cluster View

- **Physical Architecture**
  - Topology Editor

- **Electronic**
  - SW

- **Electric**
  - HW

- **Physics Geometry**
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 Transformer-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

Source Model Tool A

Rule-Model UML

Importer

Transformer

Exporter

Target-Model Tool B

Source-Metamodel

Rule4

Rule3

Rule2

Rule1

Target-Metamodel

Instance of

Instance of

<<metamodel>>

<<metamodel>>
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.

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

Model-Data Backbone

→ Multi-User (DBMS) / Single-User (XML-File)

Open API & M2M

MySQL®

Oracle®

Copyright © 2007 FZI Karlsruhe kmg
System Level Tool Support

- Actuators
- Sensors
- Optical
- Mechanical
- Thermal
- Electrical
- Magnetic
- Real environment
- Special interfaces
- Analog signal processing
- Microcontroller
- DSP
- Real Time Operating System
- System Control
- Communication with other systems
- Digital Signal Processing
- Power electronics
- Communication with other Systems

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