One Click from Model to Reality

Joint work with
Markus Herrmannsdoerfer (TUM), Stefan Kugele (TUM),
Michael Tautschnig (TU Darmstadt), and Martin Wechs (BMW)

Symposium on Automotive and Avionics Systems 2009
Contents

• Introduction
• Problem statement and vision
• Integrated modeling using the Component Language (COLA)
• Integrated tool
• Integrated development process
• Case study
• Future work
Introduction

• More that 98% of all microprocessors are assembled in embedded systems
• In many cases they form safety-critical control systems
  – Faults of any kind may be fatal
  – Lead to large warranty costs or even human injury
• Embedded System pose high demands on:
  – Reliability,
  – Robustness, and
  – Safety
• Model-driven development (MDD)
  – Is an emerging paradigm and a de-facto standard
  – Helps tackling the enormous complexity of these systems
Introduction

- 40% of a car’s costs are due to electronics / software.
- The **software** represents 50 - 70% of development costs of an ECU (electronic control unit)
- Premium class cars contain up to 100 ECUs connected by various bus systems (e.g. CAN, FlexRay, MOST, LIN, ...)

  - Increasing system **complexity**
  - Very complex **dependencies**
  - **Product costs** are a major issue
The Problem

- Lots of different MDD approaches have been proposed
- A multitude of tools has been implemented for different abstraction layers/development stages
- Most of these approaches/tools are not interoperable
- Conversion of modeling artifacts is error prone
  - Non-future-proof tool adapters
  - Incompatible semantics
  - Implementation overhead
- Optimizations are only applied locally, not globally
The Problem

Requirements

Adapter

Tool D

Adapter

Tool A

Adapter

Tool C

Manual conversion

Tool B

Manual conversion
The Vision

Integrated Language

Integrated Tool

Integrated Process

Case Study
The Vision

Integrated Language

Integrated Tool

Integrated Process

Case Study
Integrated Modeling in COLA

- Intended for automotive systems development
- Core is a data-flow language similar to Lustre or Simulink
- COLA allows an integrated development approach
  - Requirements definition
  - Functional modeling
  - Hardware platform modeling (deployment-oriented)
  - Partitioning and distribution specification
- Refinement and reuse ease modeling of complex systems
- Graphical editors and code generators have been implemented
COLA Architectural Layers

Requirements

Feature Architecture

Logical Architecture

Technical Architecture

Implementation
## COLA Architectural Layers

<table>
<thead>
<tr>
<th>Feature Architecture</th>
<th>Logical Architecture</th>
<th>Technical Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Feature Tree</strong></td>
<td><strong>Sub-unit</strong></td>
<td><strong>Cluster</strong></td>
</tr>
<tr>
<td><strong>Feature</strong></td>
<td><strong>Channel</strong></td>
<td><strong>Allocation</strong></td>
</tr>
<tr>
<td><strong>Feature Dependency</strong></td>
<td><strong>Network</strong></td>
<td><strong>Bus</strong></td>
</tr>
</tbody>
</table>

- **Automaton**: State, Transition
- **Hardware Topology**: ECU
COLA (Core) – Graphical Syntax

- **Basic blocks**
  - arithmetical functions (+, -, *, /)
  - Boolean operators (\(\land\), \(\lor\), \(\neg\))
  - comparison operators (\(=\), \(!=\), <, \(<=\), ...)

- **Data-flow networks**

- **Communication between typed ports via channels**

- **Delays** are used to store data for a clock tick

- **Mode-automata**
  - express distinct system behaviors (e.g. ignition on, ignition off)
  - decouple the modeling process
The Vision

Integrated Language

Integrated Tool

Integrated Process

Case Study
Integrated Tool

Task-list controlled front-end

Eclipse Plug-ins
- Process Engineering
- Requirements Engineering
- System Design
- Quality Management
- After-Sales Service
- Product Management

client network

LAN/WAN

backbone

Batch-mode Back-end
- Deployment Cluster
- Verification Cluster
- Simulation Cluster

Process Engine
Central Model Database
Wiki System

Wolfgang Haberl
SAASE, San Diego 2009

Fakultät für Informatik, TU München
Lehrstuhl für Betriebssysteme
Integrated Tool – Front-end
The Vision

- Integrated Language
- Integrated Tool
- Integrated Process
- Case Study
Integrated Development Process

1. System Modeling
2. Model Analysis
3. System Partitioning
4. Code Generation
5. Performance Estimation
6. System Distribution
7. Scheduling
8. System Synthesis
9. Runtime Logging
10. Runtime Data Logging
11. Checked Model
12. Partitioned Model
13. C Code Files
14. Resource Usage Figures
15. Distribution Decision
16. System Schedule
17. Executable System
18. Software & Hardware Model
19. Runtime Trace

START

Wolfgang Haberl
SAASE, San Diego 2009
COLA Clustering

- Clustering may be either manual or automatic
- Allocation of clusters onto ECUs is calculated depending on NFRs
- Connecting channels are mapped to communication middleware
System Distribution - Dependencies
Calculating System Schedules

Schedule 1

Schedule 2

Schedule 3

Schedule 4
Code Generation

- C Files
- C Files
- XML File

ECUs

<<include>>
Communication at Runtime

- Inter-cluster channels are mapped to logical middleware addresses
- Middleware also takes care of clusters internal state
- Interfacing sensors and actuators is also done by middleware
- Address „1“ is reserved for global clock
Model-level Debugging

Classical Debugging

Level of Debugging

Model-level Debugging

Model-level

Generation

Executable Code

Platform-level

Generation

Executable Code

Tracing

Captured Data

Wolfgang Haberl
SAASE, San Diego 2009

Technische Universität München
Fakultät für Informatik, TU München
Lehrstuhl für Betriebssysteme
Model-level Debugging
The Vision

Integrated Language

Integrated Tool

Integrated Process

Case Study
Case Study

Wolfgang Haberl
SAASE, San Diego 2009

Fakultät für Informatik, TU München
Lehrstuhl für Betriebssysteme
Case Study – In Action
Conclusion and Future Work

- MDD is the state-of-the-art in the design of cyber physical systems
- An integrated language along with proper tool-support is crucial for:
  - system quality
  - traceability
  - verification
- The presented approach worked well for our case study
  - Yet it is to prove, whether it scales for an actual automotive system
- A future version of our code generator shall produce AUTOSAR compatible code
Thank you for your attention!

Q & A