Tackling the Complexity of Timing-relevant Deployment Decisions in Multicore-based Embedded Automotive Software Systems

Rolf Schneider, AUDI AG
Topics

• Introduction
  • Project ARAMiS
  • ARAMiS Automotive LSSI Demonstrator
  • Deployment Challenges

• CADMOS
  • Basics
  • Modeling
  • Timing-relevant Deployment Decisions

• Tool Extensions

• Development Process Remarks

• Relationship with AUTOSAR

• Concluding Remarks
Introduction

Project Environment, Motivation and Challenge
Project ARAMiS key facts

• Coordinator: Karlsruhe Institute of Technology

• Consortium of about 40 Partners (Industrial and Academic)

• Duration: 40 month

• Start: 12/2011, End: 03/2015

• Funded by: German BMBF (Federal Ministry of Education and Research)
Large Scale Software Integration Demonstrator

From Hardware Integration to Software Integration
Large Scale Software Integration And Integration Platforms

„Block of Flats“ for SW modules

Application A

Application B

Application C

Application D

...
ARAMiS AP6.2 Demonstrator

**Hardware:**
Close-to-Production ECU with a wide range of I/O for Prototyping purposes

**Software:**
Functional integration with standard methods of AUTOSAR 4

---

**infineon AURIX TC277**

- JTG
- DAP
- Ethernet
- FlexRay
- CAN
- ADC
- GTM
- PWM
- SPI
- USB
- Transceiver
- Transceiver
- Transceiver
- PWM
- DCM
- Inertial Sensor
- EEPROM
- Extension Ports
- I/O Switches
Goals of the Demonstrator

- AUTOSAR based System (v4.x)
- Evolution of current prototyping hardware
- Integration of functions with varying safety integrity level (ASIL acc. to ISO26262) → Mixed criticality system
- Gaining experience hands-on in or at least close to normal practice
- Aiming on common practice
  - Using typical development tools and processes if possible
  - Using function software components already in production with known ‘history’
Deployment Challenges
Overwhelming Complexity of Deployment Decisions

What is the impact of a (small) change in deployment on timing?

Possible Static Allocations $O(p^t)$

<table>
<thead>
<tr>
<th>Parallel Cores $p$</th>
<th>Distributable Tasks $t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3125</td>
</tr>
<tr>
<td>6</td>
<td>15625</td>
</tr>
<tr>
<td>7</td>
<td>78125</td>
</tr>
<tr>
<td>8</td>
<td>390625</td>
</tr>
</tbody>
</table>

0,01 0,1 1 10 100 1000

Millions

0.001 0.01 0.1 1 10 100

10.7 Mio.
CADMOS
A Tool based Approach
Approach with Constructive & Analytic Part: Separation of Concerns, Modeling and Extensive Automation

- Reduced complexity of individual engineering subtasks
- Better work division for teams of domain experts
- Increased reusability by low coupling and high cohesion
- Flexible re-deployment with tool support (generators, analyzers, schedulers, ...)

Rolf Schneider, AUDI AG - Fachbereich Fahrwerk, 28.07.2014
CADMOS work flow

Model → Solver Logic

Change Modell → SAT → Deployment Schedule

Z3
CADMOS – Input/Output

- Software Architecture
- Hardware Topology
- Cost Model
- Constraints

Deployment Schedule
Modeling

Using a Basic System Model to Develop Dedicated Views on Architectures
A Basic System Model

- **Components**  Behavior
- **Ports**  Interface Points
- **Channels**  Communication

- Nested composition enables „Systems of Systems“
- Parallel, pipelined and recursive composition
Nested Composition
„Systems of Systems“

- Behavior of parent \( P \) follows from composed behavior of children \( C_1 \) to \( C_n \)
  \[
  C_1 \otimes \cdots \otimes C_n \Rightarrow P
  \]

- Finally, behavior of complete system \( S \) follows from leaves \( L_i \)
  \[
  L_1 \otimes \cdots \otimes L_m \Rightarrow S
  \]
Dedicated Architectures

\[ S = (L, R) \]

Reusing the Basic System Model

- Software Component Architecture \( L \) is a model for software-implemented application logic.
- Platform Component Architecture \( R \) is a model for platform-provided resources.
Timing-relevant Deployment Decisions

Allocations and Schedules by Formal Constraints
Explicit Deployment Decisions by Dedicated Models

- Deployment \( D = (M, Z) \)
- Mapping \( M \) allocates software onto platform-elements
  \[ M: L \rightarrow R \]
- Schedule \( Z \) defines the start times of software-elements
  \[ Z: L \rightarrow \mathbb{N}_0 \]
- Timing-relevant deployment decisions are precisely documented in allocation mapping \( M \) and the schedule \( Z \)
A Deployment Example

Rolf Schneider, AUDI AG - Fachbereich Fahrwerk, 28.07.2014
Formalized Timing Constraints

Execution Times:

\[ M(l) = r_1 \Rightarrow X(l) = x_1 \]
\[ M(l) = r_n \Rightarrow X(l) = x_n \]

Distances in Time:

\[ \Delta(l_i, l_j) := Z(l_j) - (Z(l_i) + X(l_i)) \]

Precedence:

\[ \Delta(src(e), snk(e)) \geq -delay(e) \cdot T \]

Mutual Exclusiveness:

\[ (M(v_1) = M(v_2)) \land (M(v_1) \in R_{MUTEX}) \Rightarrow (\Delta(v_1, v_2) \geq 0) \lor (\Delta(v_2, v_1) \geq 0) \]
Tool Extensions

Results from current Master Theses
Goals of Master Theses

- Generation of Architecture model
- Annotation of Timing Constraints
- Calculation of Schedules including Deployment
- Find and Resolve Errors and Inconsistencies
- Consistency of Use
- Usability
Categorization of Constraints

- **Timing**
  - Assumptions/Properties
  - Requirements
    - Deployment-Insensitive
      - Schedule is preemptive and priority-based
      - The precedence constraints
      - All runnables of an atomic SWC on same core
      - AUTOSAR tasks...
    - Deployment-Sensitive
      - WCET depending on cores
      - Memory allocation
      - Instruction code allocation
      - WCET depend on nr. of runnables on same core...
    - I/O latency
    - Multi-rates of runnables
    - Robustness for communication
    - Expiry Points...
Retrieving information from the solver where changes could help to find a valid solution
GUI Extensions

Rolf Schneider, AUDI AG - Fachbereich Fahrwerk, 28.07.2014
Development Process Remarks

Step by Step from Early Approximations to Precisely Predicted Product Properties
Iterative Incremental Development Process

User-defined system model and timing constraints

Input: L, R, C, T

Architecture Analysis & Constraint Synthesis

Constraint Solving & Result Synthesis

C' (and L, R)

Output: M, Z

Refinement of system or constraints

Automation

Assessment of allocation and schedule by users

0%
100%
Time

Precision

Rolf Schneider, AUDI AG - Fachbereich Fahrwerk, 28.07.2014
AUTomotive Open System ARchitecture

In Touch with Industrial Relevance
Relation to AUTOSAR Concepts

- **System Perspective**
  - A `System` is the comprehensive AUTOSAR element of the whole vehicle system and owns exactly one Top-Level Software Composition implemented as `RootSwCompositionPrototype`.
  - Every dedicated `SwComponentPrototype` is a part of the TLSwC.

- **Platform Perspective (left-hand side)**
  - The `System` owns several `SystemMappings`.
  - The `SwcToECUMapping` will decompose one or more `SwComponentPrototypes` to an individual `EcuInstance`.

- **Application Perspective (right-hand side)**
  - Software Components (SWC) can either be a composition of other SWCs implemented by a `CompositionSwComponentType`.
  - Or are atomic represented by an `AtomicSwComponentType`.

---

Rolf Schneider, AUDI AG - Fachbereich Fahrwerk, 28.07.2014
Concluding Remarks
Concluding Remarks

• Complexity tackled by mixed constructive/analytic approach
  • Separation of concerns using dedicated models
  • Extensive automated analysis and generation → instant answers
  • In alignment with typical automotive development process (iterative incremental)
  • Static scheduling only for „WCET scenarios“: self-timed or dynamic scheduling also possible

• Future Work / Open Issues
  • Optimization instead of „feasibility only“
  • Finer grained mapping from components to tasks
  • Direct interfaces to AUTOSAR tools
  • Allow interrupts
Thanks to Wolfgang Schwitzer and Christoph Döbber for their Support and Material and Thank You for your Attention!