Developing software architectures for autonomous driving vehicles

Sebastian Ohl - Senior Expert Driver Assistance, Erlangen, Germany

April 6th, 2016
Challenges

Developing a DAS system from scratch is hard, but the core tasks are often similar:

- Fuse, improve, and cross-check different types of sensor data
- Derive scene elements from the fused results (driving corridor, obstacles)
- Plan a full trajectory or parts of it
- Execute the trajectory by controlling the vehicle’s motion components
- Qualified and experienced personal is rare
Challenges

Step from development to embedded is a challenge:

- Computing performance may drop significantly
- Software components on ECU and a Car-PC differ (up to complete reimplementation)
- Automotive standards need to be taken into account (e.g. AUTOSAR, UDS, SomeIP, ...)
- Safety standards may break your architecture
- „Hacked“ code from pre-development may be system depended
- Supervision of execution flow and algorithmic results necessary
How to Jump Start Your Development?

- Sharing SWCs
  - Algorithmic Cores
- SWCs for HAD
- Safety Consulting
  - Tools
  - Test Automation
  - Well defined interfaces
  - Training
  - AUTOSAR
- Software
- Scalable Reference Architecture
  - High Performance ECUs
- Integrated Simulation
- Tools
- Development Process
Approach: HAD – from development to embedded

Development

- Rapid prototyping

Series production

- Rapid embedding

Application layer architecture and modules

- ADAS development framework

- AUTOSAR basic software

Car-PC

Development ECU

Target ECU
Approach: HAD – from development to embedded

- **EB Reference architecture for HAD**
  - Application-Layer Architecture and SWCs for Automated Driving Development
  - Rich Abstraction for DAS tasks like fusion, planning, control, multi-function coordination, and functional safety

- **EB tresos AD**
  - Full AUTOSAR BSW and Safety products on Safety µC
  - RTE on Linux on High-performance µC
  - OS-to-OS communication via Ethernet + SPI

- **Hardware**
  - NVIDIA DRIVE PX for prototyping or similar
  - Conti CDCU or similar for production

- **Scope**
  - Generic platform for ADAS/HAD Applications
  - Safety & Security solutions for High-Performance Systems
Different Hardware Architectures

Full AUTOSAR

for
Safety or
Performance

Microcontroller
Partitioning

for
Safety and
Performance

EB tresos AD for
NVIDIA DRIVE PX

Core1 | Core2 | Core3
---|---|---
Application SW-C | AUTOSAR SW-C | Application SW-C
RTE | Safety OS, BSW | RTE

Core1 | Core2 | Core3
---|---|---
Application | AUTOSAR SW-C | Application
Linux/QNX/ AUTOSAR... | RTE | Safety OS, BSW

Core1 | Core2 | Core3
---|---|---
Application | AUTOSAR SW-C | Application
Linux/QNX/ AUTOSAR... | RTE | Safety OS, BSW

Core1 | Core2 | Core3
---|---|---
Application | AUTOSAR SW-C | Application
Linux/QNX... | RTE | Safety OS, BSW

• Software Architectures define the next generation of microcontroller architectures
• AUTOSAR is part of each architecture as a common standard for
  – Basic Software, Safety and Security in ECU
**EB tresos AD for NVIDIA DRIVE PX**

**Architecture**
- AUTOSAR SWCs & Linux Applications
- Functional Safety & Security concept
- Linux Libraries like OpenCV / CUDA
- On Board Communication concept
- AUTOSAR Tool Environment

**Components**
- AUTOSAR SWCs
- AUTOSAR SWCs QM
- Safety RTE
- E2E
- BSW
- TimE
- MCAL
- Safety MCU
- Generic Linux Applications
- Libraries
- RTE
- DsCom
- AUTOSAR OS
- BSW
- TimE
- Linux Kernel
- Performance Microprocessors
- Can | Ethernet | FlexRay | SPI | Lin

**EB tresos Safety Products**
- EB tresos Safety OS
- EB tresos Autocore
- Application Software
Example: 1oo2D System on NVIDIA DRIVE PX

- If one channel fails, the system continues to operate with only one channel for a limited amount of time (see e.g. chemical process industry).
- Controlling this limited time and matching it with the system behavior is crucial.
- **Precondition:** high diagnostic coverage required for each channel.
Example: 1oo2D System on NVIDIA DRIVE PX
Example Application: Automated Valet Parking by Elektrobit

Automated Valet Parking

Project overview

December 2015
Reference architecture for highly automated driving functions

EB Reference architecture for HAD

- **Sensor Data Fusion**
  - Ego Fusion
  - Object Fusion
  - Grid Fusion
  - Lane Fusion
  - Sign Fusion

- **Situation and Decision Framework**
  - HAD Function
  - Situation Monitoring (SM)
  - Task Planning (TP)

- **Motion Manager**
  - Path Planning
  - Longitudinal Control
  - Lateral Control

- **Motion SW Services**
  - HMI
  - Motors
  - Braking System
  - Steering System

- **Function Specific Views**
  - Functional Error Management
  - Safety Supervisor

- **EB Sensor Cloud**
  - Electronic Horizon
  - Perception SW Services
Reference architecture for HAD: safety path for your system
Reference architecture for HAD: Structure of a module

Scalable inputs
- US Sensor Interface
- Lidar Sensor Interface
- Radar Sensor Interface

Grid Fusion
- Algorithm core

Scalable, open interfaces
- Interface 1: high volume for central ECU use
- Interface 2: mid volume for FlexRay / Ethernet
- Interface 3: low volume for CAN

Base system independence
- ADTF OS Adapter
- AUTOSAR OS Adapter
- Emb. Linux OS Adapter
Logical fusion architecture

• Separation into four basic elements
  – Selection
  – Abstraction
  – Fusion
  – Datastore

• Mapping to
  – Grid based sensor fusion
  – Object hypothesis sensor fusion by specialized instances
Occupyancy grid

• Hierarchical sensor fusions by using multiple layers

• Mapping of the logical architecture to the core elements:
  – SensorModel
  – Filter
  – Layer
  – View
  – Modifier
Occupancy grid in action
Reference architecture for HAD: Structure of a module

Various inputs
- Road Graph
- Occupancy Grid

Path Planning
- Algorithm core

Scalable, open interfaces
- Trajectory
- Center Line...

Base system independence
- ADTF OS Adapter
- AUTOSAR OS Adapter
- Emb. Linux OS Adapter

Motion Manager
- Path Planning
- Longitudinal Control
- Lateral Control
- Motion SW Services

EB Sensor Cloud

HMI
Motors
Braking System
Steering System
Path planning

• Separating the path planning process into several models:
  – Data Source
  – Algorithm
  – Conductor
  – Visualization
  – Post Processing
Industry is facing the transition from assisting to fully automated driving functions

New software/hardware architectures are needed to make this transition a success

New systems are very complex, computationally intensive, and related to safety/security

EB’s Platform for Autonomous Driving is one solution to face these upcoming challenges