Time-aware architecture and its contribution to ADAS/AD systems

Eric Schmidt
Director Lead Engineering & Safety
TTTech Auto AG
The trend towards software-defined, highly modular, converged and service-oriented platforms

How time-aware architectures accelerate the development and pave the way to autonomous driving

How to build stable, hard real-time systems that fulfill all computation chains
The automotive industry is evolving at high speed

<table>
<thead>
<tr>
<th>Change #1</th>
<th>Change #2</th>
<th>Change #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move away from one-function hardware appliances to a software-defined architecture</td>
<td>Brings freedom to OEMs or Tier1s to select the best-in-class solution elements</td>
<td>With the decoupling of application software from hardware, faster innovation cycles are enabled</td>
</tr>
<tr>
<td>Highly modular, converged, centralized, service-oriented platform is needed</td>
<td>High transparency is achieved, as opposed to the old approach with sourcing a complete closed-box system</td>
<td>Serving as a competitive advantage</td>
</tr>
<tr>
<td>Built on generic high-performance computing hardware enabling software function re-use</td>
<td>High transparency is achieved, as opposed to the old approach with sourcing a complete closed-box system</td>
<td>Maximizes solution efficiency and value</td>
</tr>
</tbody>
</table>

Software functions serving as a differentiator for OEM’s car brands
The automotive industry is evolving at high speed

<table>
<thead>
<tr>
<th>YESTERDAY &amp; TODAY</th>
<th>2020 – 2025</th>
<th>2025 +</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed E/E architecture</td>
<td>1 ECU: 1 function</td>
<td></td>
</tr>
<tr>
<td>Domain E/E architecture</td>
<td>5-7 ECUs: n functions grouped by functional domains</td>
<td></td>
</tr>
<tr>
<td>Centralized E/E architecture</td>
<td>2 ECUs: n functions</td>
<td></td>
</tr>
</tbody>
</table>

Service-oriented architecture with real time guarantees
We are at the intersection between the physical world and the cyber world

SAFETY
REAL-TIME
SECURITY
ROBUSTNESS
RELIABILITY
FAIL-OPERATIONAL
Building synergies across safety-critical industries

SAFETY
REAL-TIME
SECURITY
ROBUSTNESS
RELIABILITY
FAIL-OPERATIONAL

Building strong integration platforms

Automotive
Aerospace & Space
Off-Highway
Industrial & IoT
Fundamentals of (smart/electronic) systems

MotionWise

Sense = See
Think = Compute
Act

Time-aware safety software platform

Sensors
ADAS ECU
Actuators

But how is the timeliness of the system achieved?
How is real-time behavior guaranteed?
This is quite „hidden“ in the design decisions of the architects
Hard real-time requirements for automated driving

Braking distance with processing time in ECU is too long

Braking distance with guaranteed short processing time in ECU
What is time-aware architecture?

The "actions" of the system are majorly triggered by the progression of time (not by external events or interrupts).

Prerequisites needed to build a time-aware architecture:

- Planning (a system-wide schedule)
- Global time (time synchronization)

This applies to:

- The execution of software
- The sending and receiving of messages
Every-day examples for time-aware architectures

Please do not speak to the driver while the vehicle is in motion.
The orchestra – an example for a complex time-aware architecture
MotionWise – The safety software platform

Real-time orchestration
- Time aware architecture
- Data synchronization layer
- Global scheduling

Safety by design
- Freedom from interference
- Error and health management
- Fail-operational requirements

Open integration platform
- Unified APIs
- Heterogeneous SoCs
- Platform services & tools

Processing

Deterministic Ethernet

PCIe
MotionWise – The safety software platform

App

App

App

MotionWise

HPC
High Performance Compute
High performance compute

MotionWise

HPC
High Performance Compute

Safety host           Performance host           Performance host

Deterministic Ethernet

PCIe
MotionWise – The safety software platform
MotionWise - Core services

Extensions
- Scheduling services
- Communication services
- Time synchronisation services
- Safety & health services

Tool suite

Core

HPC
High Performance Compute
MotionWise - Extensions

Core

Automotive Services

Development Services

Support Services

Extensions

HPC
High Performance Compute
MotionWise - Tool suite

MotionWise

Core

Extensions

MotionWise Creator

MotionWise SDK

SIL Environment

Tool suite

HPC
High Performance Compute
Advantages of a time-aware architecture

- **Deterministic behavior** → increased stability and testability
- Supports **complexity reduction** → improved designs, less test cases
- **No overload scenarios** → no messages lost, no extreme latencies
- Applications executed synchronously to communication → **stable real-time behavior** with constant duration of “computation chains”
- **Synchronized global time** → allows data exchange with global time stamps
- **Increased composability** → adding additional functions (in predefined time slots) does not require complete re-evaluation of system
- Allows **offline planning** and **optimization of the schedule** → delivers foreseeable timing behavior of the system and less “experimental surprises”
Scheduled data flow with guaranteed latency for computation chains
Typical distribution of latencies – a comparison of two implementations of an exemplary system

shorter „best case“ latency

uncertain „worst case“ latency

guaranteed and shorter „worst case“ latency + smaller variation
Disadvantages of a time-aware architecture

- The “best case reaction time” (latency) of an event-triggered system is not reached … but the worst-case latency is the critical parameter for safety

- Determining the relevant processing cycle times in advance and designing the system to be “only” this quick in its reaction times (defined maximum latency)

- Defining/developing a schedule (for SW execution and communication) → “frontloading” the development process

- Flexible redistribution of computation resources (CPU execution time) is not supported → system needs to be designed for the worst-case conditions

- Synchronization between different time sources (e.g. SW execution on different hosts) is needed to avoid additional latencies
How time-aware architecture accelerates the development
Conventional integration of complex real-time systems

01 Integration of platform without configuring execution frames.

02 Applications are integrated and tested individually by APP suppliers without any timing restrictions.

03 All applications are integrated by the SW-integrator on the platform; conflicts start immediately as it is not clear who is causing problems and why.

04 Conflicts are reported back to function SW suppliers, applications have to be modified to meet the system’s timing restrictions.
Robust parallel integration process based on a time-aware architecture

01
Platform configuration includes execution boundaries for the applications.

Robustness through clear allocation and monitoring of resources (memory, CPU, comm.)

02
Applications are integrated and tested individually by the APP suppliers into their respective execution boundaries.

Parallel Integration to speed-up software development of multiple software suppliers

03
All applications are integrated and are immediately able to run together; violations by APPs are detected easily.

Complete software integrated for functional testing
This speeds up time-to-market for new functionalities, guarantees safety, and allows software investments to be reused for highly automated driving projects.

A service-oriented architecture with real-time guarantees provides a seamless roadmap to full automation in the future.

MotionWise helps organizations move away from a slow, costly, complex and iterative integration process to a platform approach.

Key take-aways

Eric Schmidt
Director Lead Engineering & Safety
eric.schmidt@tttech-auto.com
www.tttech-auto.com