2020 – Setting the Direction of Automotive EE

2019-10-08  Dr. Günther Heling
Automotive Trends

**Automated Driving**

**Connectivity**

**Electrification**

ACE - New User Experiences

➔ **ACES**
that’s a game changer !

➔ Focusing on the enablers prepares for the future

[https://www.welt.de/sonderthemen/zvei/article116780491/Hier-kommt-das-Superhirn-auf-Raedern.html](https://www.welt.de/sonderthemen/zvei/article116780491/Hier-kommt-das-Superhirn-auf-Raedern.html)
Automotive EE has to adopt new technologies → We need to go IT!

<table>
<thead>
<tr>
<th>EE Trends</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ECU Topology</td>
<td></td>
</tr>
<tr>
<td>ECU HW</td>
<td></td>
</tr>
<tr>
<td>SW Architecture (ECU)</td>
<td></td>
</tr>
<tr>
<td>SW Framework</td>
<td></td>
</tr>
<tr>
<td>SW Platforms</td>
<td></td>
</tr>
<tr>
<td>SW Configuration</td>
<td></td>
</tr>
<tr>
<td>SW Language</td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td></td>
</tr>
<tr>
<td>Work split</td>
<td></td>
</tr>
<tr>
<td>Development pattern</td>
<td></td>
</tr>
<tr>
<td>SW Lifecycle</td>
<td></td>
</tr>
</tbody>
</table>

VECTOR
Automotive EE has to adopt new technologies → We need to go IT!

<table>
<thead>
<tr>
<th>ECU Topology</th>
<th>Domain oriented</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECU HW</td>
<td>μController</td>
</tr>
</tbody>
</table>

- SW Architecture (ECU)
- SW Framework
- SW Platforms
- SW Configuration
- SW Language
- Communication
- Work split
- Development pattern
- SW Lifecycle
Future ECU Types

3 types of ECUs:

1. **Sensor and Actuator ECUs**
   - commodity ECUs
   - basic functionality

2. **Integration ECUs**
   - per domain or per zone
   - lean or rich functionality

3. **Vehicle Brain**
   - high performance computers based on SoCs
   - secure IT-like software acc. ISO 26262
   - focus of functional innovation
   - linked to the cloud
SoCs allowing an optimized partitioning of Software

- Linux
- Android
- QM Application
- QM Adaptive AUTOSAR
- SafePOSIX
- Safe Application
- Safe Adaptive AUTOSAR
- Safe Classic AUTOSAR
- Safe Realtime OS
- Shared Memory IPC

→ allows high performance computing based on LINUX, Android combined with ASIL D and real time support based on AUTOSAR
Automotive EE has to adopt new technologies → We need to go IT!

<table>
<thead>
<tr>
<th>ECU Topology</th>
<th>Domain oriented</th>
<th>Cloud &amp; HPC &amp; Zone orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECU HW</td>
<td>( \mu \text{Controller} )</td>
<td>( \mu \text{Processor, SoC} )</td>
</tr>
<tr>
<td>SW Architecture (ECU)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW Framework</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW Platforms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW Configuration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW Language</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Work split</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development pattern</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW Lifecycle</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Automotive EE has to adopt new technologies → We need to go IT!

### EE Trends

1. **ECU Topology**
   - Domain oriented
   - Cloud & HPC & Zone orientation

   - **ECU HW**
     - \( \mu \text{Controller} \)
     - \( \mu \text{Processor, SoC} \)

2. **Communication**
3. **Work split**
4. **Development pattern**
5. **SW Lifecycle**

- **SW Architecture (ECU)**
- **SW Framework**
  - AUTOSAR Classic
- **SW Platforms**
  - Tier1
- **SW Configuration**
- **SW Language**
AUTOSAR Classic is widely used

Estimation of AUTOSAR Classic penetration for new projects

Source: AUTOSAR OC Nov.2018
Future ECU Types and Usage of AUTOSAR

3 types of ECUs:

1. Sensor and Actuator ECUs
   - commodity ECUs
   - basic functionality
   ➔ Classic AUTOSAR

2. Integration ECUs
   - zone oriented
   - lean or rich functionality
   ➔ Classic & Adaptive AUTOSAR

3. Vehicle Brain
   - high performance computers based on SoCs
   - secure IT-like software acc. ISO 26262
   - focus of functional innovation
   - linked to the cloud
   ➔ Adaptive AUTOSAR
   (supported by Classic for fail-operational)

Seamless interoperability!

Classic
- C
- static
- monolithic / top-down
- model-based
- configuration at compile time

⇒ maximal efficiency

Adaptive
- C++
- dynamic
- modular / parallel
- model-based and software-based
- configuration at runtime

⇒ maximal flexibility
Future ECU Types and Usage of AUTOSAR

Adaptive

... already used in production projects
- VW ID.3 SOP 2019
- BMW iNext (ASIL D) SOP 2021
...

... many more projects are on the way
> 150 prototype and evaluation projects

3 types of ECUs:

1. Sensor and Actuator ECUs
   - commodity ECUs
   - basic functionality
   - Classic AUTOSAR

2. Integration ECUs
   - zone oriented
   - lean or rich functionality
   - Classic & Adaptive AUTOSAR

3. Vehicle Brain
   - high performance computers based on SoCs
   - secure IT-like software acc. ISO 26262
   - focus of functional innovation
   - linked to the cloud
   - Adaptive AUTOSAR
(supported by Classic for fail-operational)
Beyond Adaptive AUTOSAR: SW Runtime Frameworks ("Vehicle OS")

**User Applications**
- components by OEM, Tier1, Tier2, ...
- integrated by OEM or Tier1

**SW Runtime Framework ("Vehicle OS")**
- AUTOSAR Basic Software
- Hypervisor, Libs, Security extensions
- Resource monitoring
- System services (diagnostic, SW update, ...)
- Basic apps (data server, personalization, ...)

➔ Elements provided by OEM, Tier1, Tier2, ...
➔ OEM specifics should be minimized!

**Hardware**
- provided by Tier1
User Applications
- Functional SW components („Apps“)

SW Runtime Framework
- AUTOSAR Basic Software
- System functions (diagnostic, SW update, …)
- Basic apps (data server, personalization, …)

SW Development Framework
- Design, configuration, integration, testing, calibration, …
**User Applications**
- Functional SW components („Apps“)

**SW Runtime Framework**
- AUTOSAR Basic Software
- System functions (diagnostic, SW update, …)
- Basic apps (data server, personalization, …)

**SW Development Framework**
- Design, configuration, integration, testing, calibration, …

**SW Platform Architecture** („one software for all projects“)
- Definition of platform components and variants
- 200% Product → 150% Program → 100% Project
- Strategy and technique for Reuse / Update / Merge

---

**Platforms, Frameworks, ...**

- **User Applications**
  - Functional SW components („Apps“)

- **SW Runtime Framework**
  - AUTOSAR Basic Software
  - System functions (diagnostic, SW update, …)
  - Basic apps (data server, personalization, …)

- **SW Development Framework**
  - Design, configuration, integration, testing, calibration, …

- **SW Platform Architecture** („one software for all projects“)
  - Definition of platform components and variants
  - 200% Product → 150% Program → 100% Project
  - Strategy and technique for Reuse / Update / Merge
**User Applications**
- Functional SW components ("Apps")

**SW Runtime Framework**
- AUTOSAR Basic Software
- System functions (diagnostic, SW update, …)
- Basic apps (data server, personalization, …)

**SW Development Framework**
- Design, configuration, integration, testing, calibration, …

**SW Platform Architecture** ("one software for all projects")
- Definition of platform components and variants
- 200% Product → 150% Program → 100% Project
- Strategy and technique for Reuse / Update / Merge

**SW Platform Development Framework**
- Support of different specification levels → engineering backbone
- Support of platform management and parallel working
User Applications
- Functional SW components („Apps“)

SW Runtime Framework
- AUTOSAR Basic Software
- System functions (diagnostic, SW update, …)
- Basic apps (data server, personalization, …)

SW Development Framework
- Design, configuration, integration, testing, calibration, …

SW Platform Architecture („one software for all projects“)
- Definition of platform components and variants
- 200% Product → 150% Program → 100% Project
- Strategy and technique for Reuse / Update / Merge

SW Platform Development Framework
- Support of different specification levels → engineering backbone
- Support of platform management and parallel working

Processes & Organization
- Agile development (SCRUM, …) & ASPIEC
- Safety and security processes (ISO 26262, ISO 21434)
Automotive EE has to adopt new technologies → We need to go IT!

### EE Trends

<table>
<thead>
<tr>
<th>ECU Topology</th>
<th>Domain oriented</th>
<th>Cloud &amp; HPC &amp; Zone orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECU HW</td>
<td>$\mu$Controller</td>
<td>$\mu$Processor, SoC</td>
</tr>
</tbody>
</table>

1. **SW Architecture (ECU)**
2. **SW Framework**
   - AUTOSAR Classic
   - Adaptive, Runtime & Dev Frameworks
3. **SW Platforms**
   - Tier1
   - Tier1 and OEM

- **SW Configuration**
- **SW Language**
- **Communication**
- **Work split**
- **Development pattern**
- **SW Lifecycle**
Automotive EE has to adopt new technologies → We need to go IT!

**EE Trends**

- **ECU Topology**: Domain oriented → Cloud & HPC & Zone orientation
- **ECU HW**: μController → μProcessor, SoC
- **SW Architecture (ECU)**
- **SW Framework**: AUTOSAR Classic → Adaptive, Runtime & Dev Frameworks
- **SW Platforms**: Tier1 → Tier1 and OEM
- **SW Configuration**
- **SW Language**
- **Communication**
- **Work split**: ECU oriented
- **Development pattern**
- **SW Lifecycle**
Development Workflow in an Adaptive Environment

OEMs, Tier1s and Tier2s contribute according their expertise

**OEM: E/E Architect**
- Map platform SW into E/E architecture

**Tier1: ECU Architect**
- Define high level UCM, watchdog, diagnostics
- HW not yet available

**Tier1: Platform SWCL Architect**
- Design new service interfaces for internal interaction

**Tier1/2: SWCL Architect**
- Platform SW mapping to OEM needs
- Transformation of service interfaces
- New apps / functionalities
- Create diagnostic mapping

**Tier1: Platform SWCL Architect**
- Design new service interfaces for internal interaction

**Tier1: SWCL Architect**
- Define platform available services
- Service Interfaces
- Service Id
- Instance Id

**OEM: Vehicle Integrator**
- Provide Vehicle Package

**Tier1/2: ECU Integrator**
- HW Setup
- OS Installation / Startup / Configuration
- SWCL bound to machine
- Business logic tested in evaluation board

**Tier1/2: ECU Integrator**
- HW Setup
- OS Installation / Startup / Configuration
- SWCL bound to machine
- Business logic tested in evaluation board

**Tier1/2: SWCL Integrator**
- Create a complete SWCL

**Application Developer**
- Prepare for OEM backend
- Install BSW (core)
- Create SW package
- Collect executables, manifests

**executable source code**
- Write App
- Add port prototypes
- Add PHM, DIAG, PER
- TSYN, ports
- Write business logic

**Mapped service instance to machine**
- Machine design
- Extract
- SWCL design
- Global service catalogue
- Define globally available Services
- Service Interfaces
- Service Id
- Instance Id

**Installable SWCL (core)**
- Installable SWCL (App)
- Create SW package
- Collect executables, manifests

**Export global service catalogue**
- Define globally available Services
- Service Interfaces
- Service Id
- Instance Id

**Installable SWCL (App)**
- Installable SWCL (core)
- Export global service catalogue
- Collect executables, manifests

**Deploy ARA platform services**
- Rights/access management
- PHM (SWCL internal)
- PER
- Local IPC communication

**Create Machine design**
- Extract
- SWCL service catalogue
- Machine design
- Process design
- Map machine design to machine

**Write business logic**
- Write App
- Add PHM, DIAG, PER
- TSYN, ports
- Write business logic

**Define SWCL internal services**
- Process design
- Map process design to machine
- Define provide ports
  (ara::com, TSYN,...)
- PHM Daisy Chaining
- DEXT (enriched)
- Map diagnostic object to process

**Define Machine Deployment**
- Processor
- Processor Cores
- Function Groups
- Module Instantiation

**Define platform available services**
- Service Interfaces
- Service Id
- Instance Id

**Define SWCL Requirements**
- DEXT (initial)
- Target address
- Map service interface to SWCL

**Define SWCL internal services**
- Process design
- Map service interface to SWCL

**Create SWCL design**
- Map service instance to SWCL design

**Create Machine design**
- SWCL design
- Global service catalogue
- SWCL requirements
- DEXT (initial)
- Service interfaces
- Service Id
- Instance Id
EE Trends

Automotive EE has to adopt new technologies → We need to go IT!

<table>
<thead>
<tr>
<th>1. ECU Topology</th>
<th>Domain oriented</th>
<th>Cloud &amp; HPC &amp; Zone orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECU HW</td>
<td>μController</td>
<td>μProcessor, SoC</td>
</tr>
</tbody>
</table>

2. SW Architecture (ECU)

<table>
<thead>
<tr>
<th>SW Framework</th>
<th>AUTOSAR Classic</th>
<th>Adaptive, Runtime &amp; Dev Frameworks</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW Platforms</td>
<td>Tier 1</td>
<td>Tier 1 and OEM</td>
</tr>
</tbody>
</table>

3. SW Configuration

<table>
<thead>
<tr>
<th>Communication</th>
<th>Work split</th>
<th>Development pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ECU oriented</td>
<td>Discipline oriented (Appl, BSW, HW)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SW Lifecycle</td>
</tr>
</tbody>
</table>
Automotive EE has to adopt new technologies → We need to go IT!

### EE Trends

1. **ECU Topology**
   - Domain oriented
   - Cloud & HPC & Zone orientation

2. **ECU HW**
   - μController
   - Processor, SoC

3. **SW Architecture (ECU)**
   - Monolithic (ECU)
   - Modular (Cluster – Service)

4. **SW Framework**
   - AUTOSAR Classic
   - Adaptive, Runtime & Dev Frameworks

5. **SW Platforms**
   - Tier 1
   - Tier 1 and OEM

6. **SW Configuration**
   - Static
   - Dynamic (using manifests)

7. **SW Language**
   - C
   - C++, GO, ...

8. **Communication**
   - Signals, CAN, LIN
   - Services, Ethernet

9. **Work split**
   - ECU oriented
   - Discipline oriented (Appl, BSW, HW)

10. **Development pattern**
    - Static
    - Agile, CI based

11. **SW Lifecycle**
    - Frozen at SOP
    - Update and Upgrade over the Air
Automotive EE has to adopt new technologies → We need to go IT!

To cope with future challenges we need to adapt in many aspects - let’s find the best way to complement each other.

And we need to capitalize our old strengths.
For more information about Vector and our products please visit

www.vector.com

Author:

Dr. Günther Heling