AGENDA

E/E ARCHITECTURE OVERVIEW

SOA COMMUNICATION PROTOCOLS

TAKEAWAYS & PERSPECTIVE
E/E ARCHITECTURE OVERVIEW

SOA COMMUNICATION PROTOCOLS

TAKEAWAYS & PERSPECTIVE
VEHICLE DESIGN TREND

Hardware-defined
Off the lot = Depreciation

Software-defined
Update = Appreciation

Hardware-defined

Off the lot = Depreciation

Software-defined
Update = Appreciation

Service Oriented Architecture

Domain → Cross-Domain → Zonal architecture

SDK providing standard APIs

Software Services

Infrastructure Virtualization
WHAT ACCELERATES SOA

- Continuous improvement
  - Connectivity & OTA
- Self or cooperative driving
  - AD/ADAS & V2X
- Cross-domain needs
  - Domain overlapping between ADAS, HMI, energy, connectivity, etc.
- In-car Marketplace
  - Apps and features on demand
- New vehicle ownership
  - Carsharing, user profiling, etc.
- Scalable car platform
  - Seamless integration and software modularity
- Software complexity
  - Decoupling, integration, versioning, variant management, etc.
- Time to market
  - Reduce Complexity and facilitate integration
TOWARDS SOA – TARGET

Autosar  RTOS  Unix-like

ECU  ECU  ECU

IT Infrastructure
TOWARDS SOA – TARGET

HW abstraction
Resource optimization
Reconfigurability
Simplify:
- Resource management interfaces
- Service allocation

Virtualized E/E Architecture
- System (HW & OS)
- Network

E/E Architecture

SOA requires a service-oriented infrastructure

Virtualization

SOA communication middleware

Data-centric
Modularity
Reusability
Simplify:
- New use case adding
- Data interfaces

From signal to service
Service discovery
HW/SW decoupling

STELLANTIS

2021/11/03
Stellantis — IEEE SA Ethernet & IP @ Automotive Technology Day
Key points:

- Heterogeneity
- Coexistence
- Boundary
- Standardization
AGENDA

E/E ARCHITECTURE OVERVIEW

SOA COMMUNICATION PROTOCOLS

TAKEAWAYS & PERSPECTIVE
Ethernet & IP enable the deployment of centralized architecture

Standardization of protocol deployment:
- Data model definition
- Data version evolution
- Software stack evolution
- Non-functional variant management

SOA architecture style seems to be the solution for reducing the actual complexity

A service is:
- A software service
- Self-contained
- A black box to service consumers
- May be composed of other services
- Independently deployable
- Independently updatable
SOA = Software + Network + Business process

- **Modularity**
  - Application Abstraction
  - Granularity
  - Service Composition
  - Reusability
  - Infrastructure Virtualization
  - Scalability

- **Loose coupling**
  - Fault Tolerance
  - Flexibility
  - Orchestration

- **Open Standards**
  - Standardization
  - Interoperability

- **Simplicity**
  - Redundant Complexity

**Service Bus**

- **Virtualized E/E Infrastructure**

**Service Interface definition**

- **Client Implementation**
  - Stub/proxy
- **Service Implementation**
  - Skeleton

**Control services**

- Auto-scaling
- Load balancing
- Self-Healing

**Interactions**

- Data logging
- Network monitoring
- Orchestration
- Statistics
- Error handling
- Service scheduling
- Service allocation
- API
- Interface evolution
## SOA FOR CAR PLATFORM EVOLUTION

<table>
<thead>
<tr>
<th>Time</th>
<th>EXISTING</th>
<th>NOW</th>
<th>FUTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Space</strong></td>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Offboard Infrastructure</td>
<td><img src="image4.png" alt="Diagram" /></td>
<td><img src="image5.png" alt="Diagram" /></td>
<td><img src="image6.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Onboard Infrastructure</td>
<td><img src="image7.png" alt="Diagram" /></td>
<td><img src="image8.png" alt="Diagram" /></td>
<td><img src="image9.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Automotive Software</td>
<td><img src="image10.png" alt="Diagram" /></td>
<td><img src="image11.png" alt="Diagram" /></td>
<td><img src="image12.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**EXISTING**
- Offboard Infrastructure
- Onboard Infrastructure
- Automotive Software

**NOW**
- ECU
- RTE
- OS
- IT

**FUTURE**
- ECU
- RTE
- Car OS
- IT
- Virtualization
- Sensor
- Actuator
- IPC (inter-process communication)
- Communication protocols
AUTOMOTIVE SOA COMMUNICATION ARCHITECTURE

- Classic Autosar
- Classic Autosar
- Classic Autosar
- Adaptive Autosar
- Unix-like
- Unix-like

Z-ECU Z-ECU C-ECU T-Box

Cloud Infrastructure

Urban Infrastructure

Factory Automation Infrastructure

SOME/IP (-SD)

DDS

OPC UA

DDS

MQTT

HTTP/Rest

WebSocket

CoAP

SoAd / TCP or UDP based

AMQP
## SOA Protocol Comparison

<table>
<thead>
<tr>
<th>SOA Criteria</th>
<th>AMQP</th>
<th>MQTT</th>
<th>DDS</th>
<th>SOME/IP</th>
<th>OPC UA</th>
<th>CoAP</th>
<th>HTTP/REST</th>
<th>WebSocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication Paradigm</td>
<td>P/S</td>
<td>P/S</td>
<td>P/S</td>
<td>P/S</td>
<td>R/R</td>
<td>P/S</td>
<td>R/R</td>
<td>R/R</td>
</tr>
<tr>
<td>Topology</td>
<td>N-1-N or 1-1</td>
<td>N-1-N</td>
<td>N-N</td>
<td>1-N</td>
<td>N-1-N or N-N</td>
<td>N-1-N</td>
<td>N-1-N</td>
<td>N-1-N</td>
</tr>
<tr>
<td>IDL</td>
<td>-</td>
<td>Franca*</td>
<td>OMG</td>
<td>Franca</td>
<td>OMG</td>
<td>-</td>
<td>REST, Franca*</td>
<td>WEB, Franca*</td>
</tr>
<tr>
<td>Discovery</td>
<td>✘</td>
<td>✘</td>
<td>Data</td>
<td>Service</td>
<td>Service</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>Asynchronous available</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Broker-based</td>
<td>✓</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>QoS Awareness</td>
<td>Availability</td>
<td>Reliability</td>
<td>✓</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>TLS / DTLS</td>
<td>TLS</td>
<td>TLS</td>
<td>TLS/DTLS</td>
<td>TLS/DTLS</td>
<td>TLS/DTLS</td>
<td>DTLS</td>
<td>TLS</td>
<td>TLS</td>
</tr>
<tr>
<td>Extra Security in the protocol</td>
<td>Session authentication</td>
<td>Session authentication</td>
<td>Session authentication</td>
<td>Data authentication</td>
<td>Data confidentiality</td>
<td>Session authentication</td>
<td>Data confidentiality</td>
<td>Operation access control</td>
</tr>
<tr>
<td>Application</td>
<td>Cloud</td>
<td>Car-Cloud &amp; In-Vehicle</td>
<td>In-Vehicle</td>
<td>In-Vehicle</td>
<td>Car-Cloud</td>
<td>Car-Cloud</td>
<td>Car-Cloud &amp; In-Vehicle</td>
<td>Car-Cloud &amp; In-Vehicle</td>
</tr>
</tbody>
</table>

R/R = Request/Response; P/S = Publish/Subscribe. L = Linux; CA= Classic Autosar; AA= Adaptive Autosar; A= Android. * Require extra convertor to use FrancaIDL.
SOA PROTOCOLS

DDS

Data-centric

Data addressing
Non-legacy ECU communication
App abstraction thru named network data
Data discovery / Location transparence

SOMES/IP

Automotive SOA

Service interface
Legacy ECU communication
Service discovery

OPC UA

Interoperability

Data modeling
Non-legacy ECU communication
Protocol abstraction thru common data models
Server discovery (host/session discovery)

Vendor Specific Extensions

Companion Information Models
Base, DA, AC, HA, Programs

OPC UA Meta Model
Basic rules for exposing information with OPC UA

Built-In Information Models
Message Model
External Configurations, Statements, Message Readers and Writers

Built-In Security
Publication/Subscribe

(Source: omg.org/spec/DDS)
(Source: autosar.org)
(Source: opcfoundation.org)
## AUTOMOTIVE SOA CHALLENGES

| **Heterogeneity** | • Ensure the solution is compatible with the multiple ecosystems existing in the car (Classic and Adaptive Autosar, Android, Linux, etc)  
• Define and comply with standards and the global deployment pipelines |
| **Dynamincity** | • Deploy dynamic services from stateful to stateless in terms of security certifications, capacities, etc.  
• Reduce service complexity (towards fine-grained services)  
• Respect the latency limits required by actual and new applications |
| **Development Process** | • Develop new interface and network standards and all the control layer services  
• Evolve the development and testing process of new software requirements  
• Transform or repackage the legacy applications for new SW contexts |
| **Resource Consumption** | • Maintain lightweight consumption for the control-plane services and SOA related mechanisms  
• Handle simultaneously different types of communication protocols (pub/sub, req/resp…) and different QoS levels |
| **RAMS & Cybersecurity** | • Handle compatibility vulnerabilities of onboard/offboard heterogeneous SW contexts.  
• Rethink and adapt the current failover mechanisms to meet safety requirements  
• Deploy security mechanisms to prohibit the unauthorized access to the local/remote attack surface |
AGENDA

E/E ARCHITECTURE OVERVIEW

SOA COMMUNICATION PROTOCOLS

TAKEAWAYS & PERSPECTIVE
New services in SOA-compatible EE Architecture
Some services may stay in control plane

Orchestration in control plane
- Task Assignment
- Network Supervision
- Error Handling
- Dynamic Reconfiguration

The EE infrastructure is transparent for services thanks to the control plane

Control plane:
- SOTA/FOTA/AOTA update
- Vehicle state management
- Lifecycle management
- QoS management
- Dynamic flow mapping
- Data optimization
- Network diagnosis
- Load balancing
- Network access control
- Network virtualization
- Service monitoring
SOA requires the communication management and the service-oriented infrastructure

Standardization is the key to reduce the actual complexity

Different protocols aim at solving different problems, i.e., legacy vs interoperability vs data-centric

Complicated to unify communication protocols, but easier to unify data/information models

Data abstraction interface is essential for SOA deployment, i.e., API

Both data interface and software components are updatable and require branching and merging strategies

Explicit service design policy for data and control planes, respectively

HW/SW decoupling thanks to control plane
While looking at the evolution of...

From TCP/IP to information-centric network

From 5G to 6G

The trend is towards Data-centric

Data modeling
Data addressing, i.e., topic or named data

Neither data producer nor data consumer is permanent. Apps are short-lived, constantly added/updated/removed.

Few programming effort
Scalable software architecture
## Evolution of Software Architecture

<table>
<thead>
<tr>
<th>1990’s</th>
<th>2000’s</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spaghetti code</strong></td>
<td><strong>Lasagna code</strong></td>
</tr>
<tr>
<td>Unstructured and hard-to-maintain code caused by lack of style rules or volatile requirements. This architecture resembles a tangled pile of spaghetti in a bowl.</td>
<td>Source code with overlapping layers, like the stacked design of lasagna. This code structure makes it difficult to change one layer without affecting others.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2010’s</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ravioli code</strong></td>
<td><strong>Pizza code</strong></td>
</tr>
<tr>
<td>Isolated bits of code that resemble ravioli. These are easy to understand individually but—taken as a group—add to the app’s call stack and complexity.</td>
<td>A codebase with interconnected classes or functions with unclear roles or responsibilities. These choices result in a flat architecture, like toppings on a pizza.</td>
</tr>
</tbody>
</table>

(Source: techtarget.com)