



THE LINUX FOUNDATION  
**OPEN SOURCE SUMMIT**

NORTH AMERICA

# How to accelerate Software Defined Vehicle(SDV) with OSS

Yuichi Kusakabe  
Chief Architect / OSPO Tech Lead  
Honda Motor Co., Ltd.

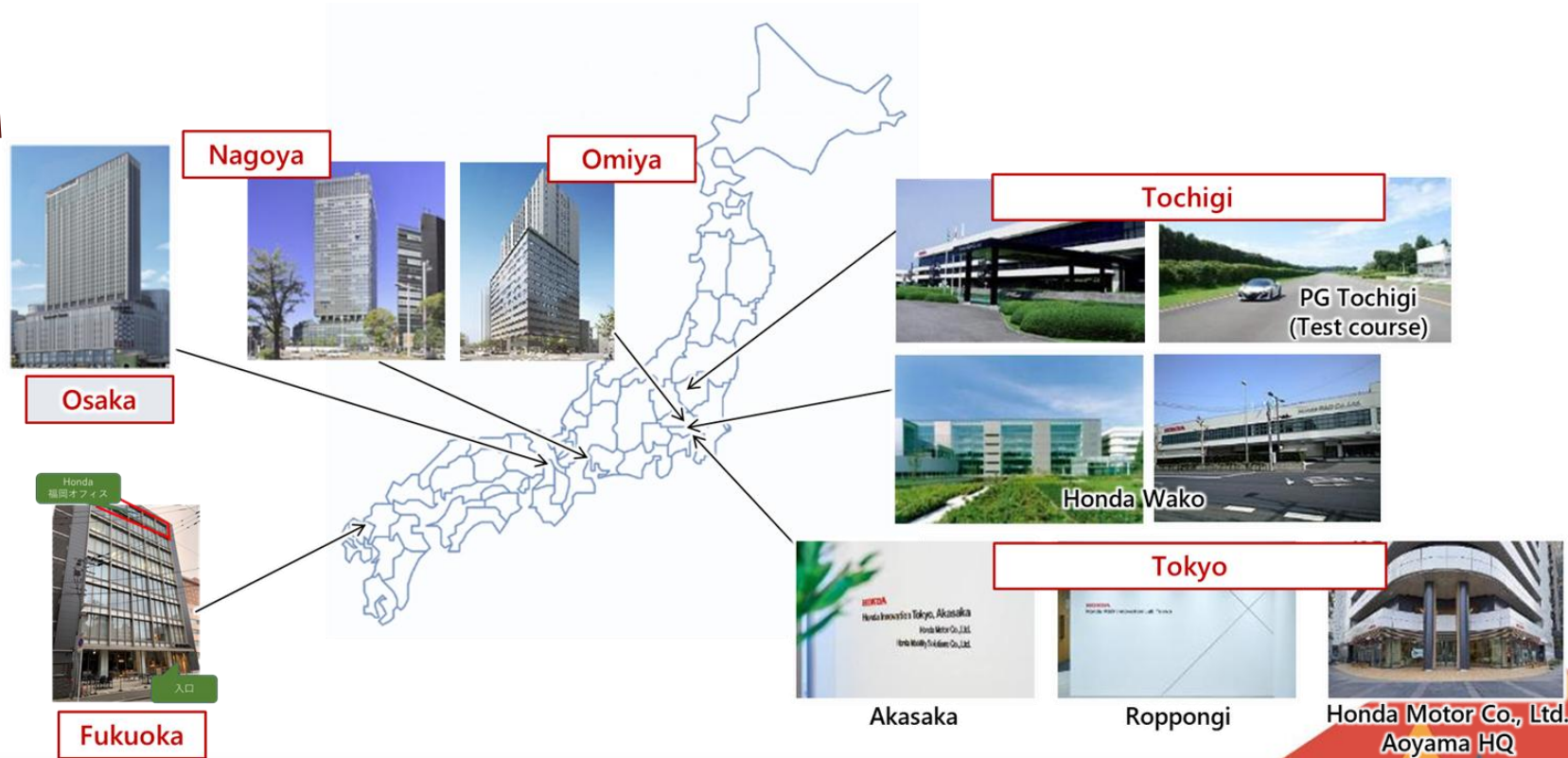


# About myself(Yuichi Kusakabe)

- 2005/4 Automotive Tier1 in Japan
  - Renesas H8S AVN power control(Non-OS)
  - Panasonic UniPhier USB-Audio(Fastest in the industry with HDD transfer function)
- 2009/12 Working together with in Japan OEM
  - Engaged in the R&D of SoC for next-generation IVI and multimedia OS
- 2011/1 Paradigm shift to use OSS
  - Renesas RMA1 Linux-PF Project Leader, AGL Board Member
  - Renesas R-Car M2/H3/M3 Fast Boot, Java, HTML5 browser
- 2020/6 Join Honda Motor in IVI software development team
  - Qualcomm SA8155P AAOS based IVI Lead Architect
- 2024/4 Launched Honda OSPO(Open Source Program Office)
  - SDV(Software Defined Vehicle) SW Chief Architect and OSPO Tech Lead



# Honda Software Studio in Japan



**HONDA**  
The Power of Dreams



# Agenda

- Background of In-House Software development
- Overview of the SDV
- Key points of the SDV
- Cooperation with the OSS community
- Conclusion
- Additional information





# Background of In-House Software development



# About Honda's DNA (Waigaya and A00)

## Waigaya

"Waigaya" is Honda's unique culture of open and frank discussions about "dreams" and "ideal work styles," regardless of age or position. Instead of being venues for compromise or coordination to reach consensus, Waigaya sessions are an opportunity to create new values and concepts by thoroughly exchanging opinions in a serious and honest manner. Many of Honda's innovations, such as industry firsts and world firsts, have been born from deepening substantive discussions through Waigaya.

## A00

The first thing discussed when a Honda project is launched is, "what kind of world is this work trying to realize?" One might call it a concept or guideline to ensure that we never waver until the very end. At Honda, we call this "A00." Whenever we hit a wall or have a disagreement, we always go back to it and use it as the basis for all our decisions.

# About Honda's DNA (120% products quality)

## | Aiming for 120% products quality

"We have to aim for 120% product quality. If 99% of the products we make are perfect, that would seem like a pretty good record. However, the customers who become the owners of the remaining 1% will surely consider their products 100% defective. It is unacceptable that even one customer in a thousand—even one customer in ten thousand—should receive a defective product. That's why we have to aim for 120%." When founder Soichiro Honda said this he defined the company's fundamental approach to quality: what it means to strive to be a company society wants to exist. Determined to meet or exceed the expectations of customers, Honda is taking new initiatives to reach ever-higher product quality standards. That is who we are.

To strengthen customer trust by offering products founded in safety and offering a new level of outstanding quality, Honda has created a quality cycle that continuously enhances quality at every stage: design, development, production, sales and after-sales service.



# Background of In-House IVI Software development

## ■ What is IVI (in-vehicle information) system ?

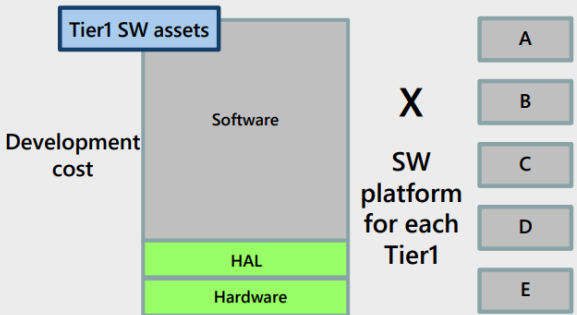
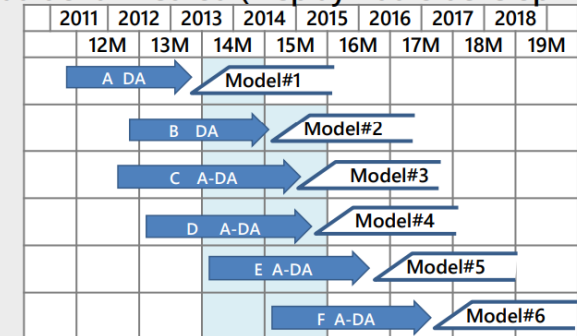
An infotainment system that connects to the outside of the vehicle, adding communication functionality



The IVI system connects various things and provides "information" and "entertainment"

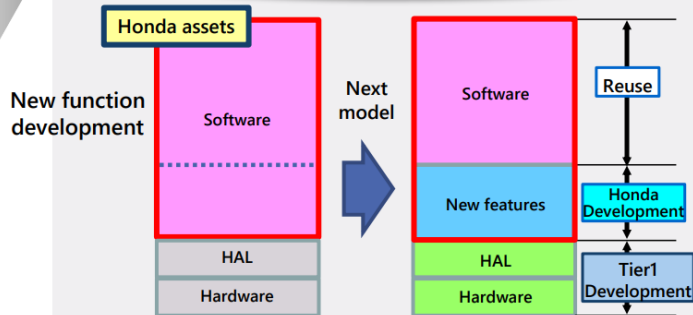
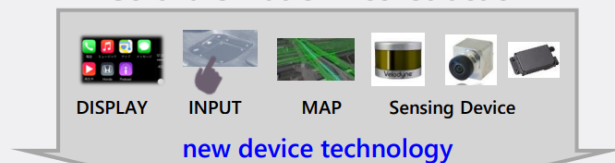
# Background of In-House IVI Software development

Traditional method (Display Audio development)



- Software development costs for each model
- Development resources
- Software quality issues

Software Platform construction

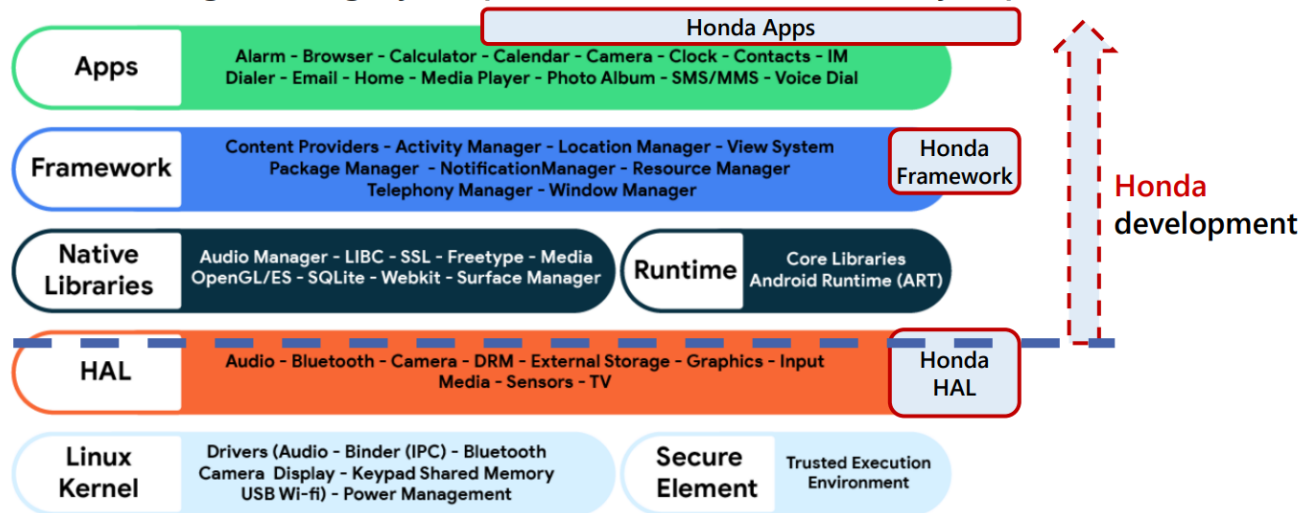


Increase SW reusability and minimize duplicate development

Aiming to improve the efficiency and quality of software development

# Background of In-House IVI Software development

- Honda develops software at a higher level than HAL
- Responsible for HAL and hardware at Tier 1
- Some Interfaces that are missing from the AOSP standard, Honda will design and add new Interfaces
  - Customization = Negative Legacy = Update failure, Minimize is very important



※出所 <https://source.android.com/docs/setup/about?authuser=2&%3Bhl=ja&hl=ja>

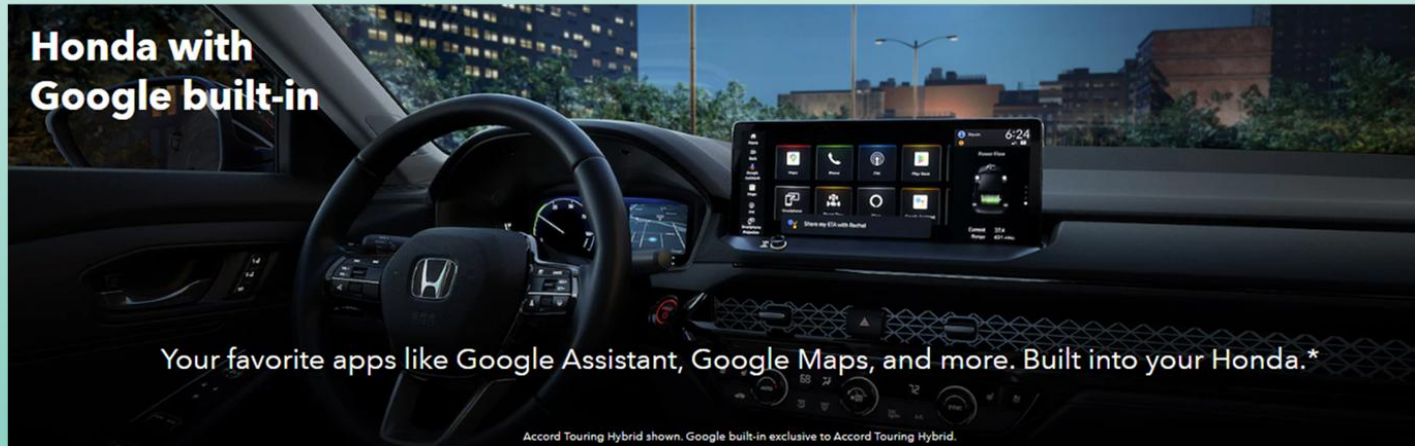
Optimize division of labor to focus on application development  
Succeeded in developing a based on AAOS that in-house development of IVI software platform

# 23M/Y US ACCORD IVI system

## ■ Achievement

The First-Ever Honda with Google built-in.  
The 2023 Accord Touring Hybrid with Google built-in also offers an electrifying hybrid powertrain, 12-speaker Bose premium sound system, Head-Up Display, and more.

[2023 Honda Accord Google Services - YouTube](#)



# Overview of the SDV



# What is SDV ?

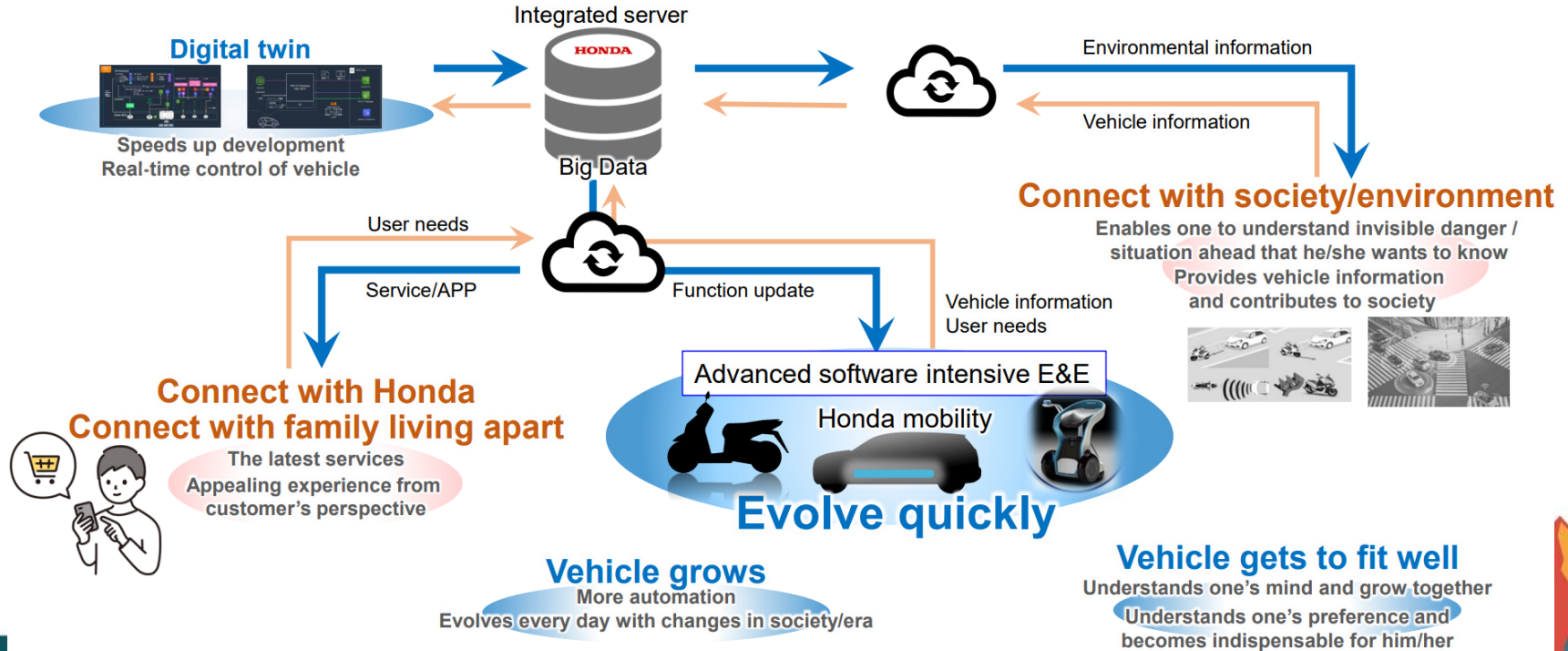
## Software Defined Vehicle





# Example of SDV(Software Defined Vehicle) system

Realize each individual's dream and deliver the joy of mobility  
by outside vehicle connection and quick evolution



# Example of SDV(Software Defined Vehicle) system



# Example of SDV(Software Defined Vehicle) system

Virtual, digital twin development

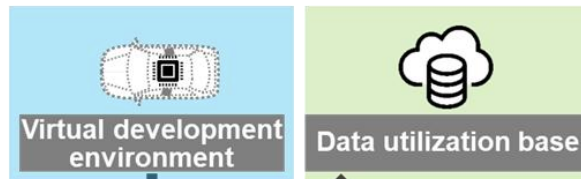


  
Utilize standard architecture/  
technology/tool

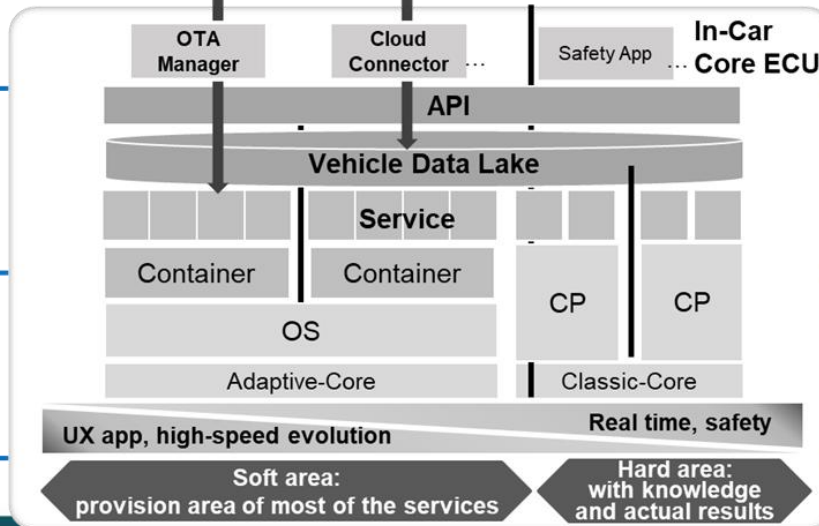
  
Hardware-less development/  
verification environment

  
End To End environment  
including other ECUs

Out-Car



Data driven development



  
Enhance development  
efficiency by automation

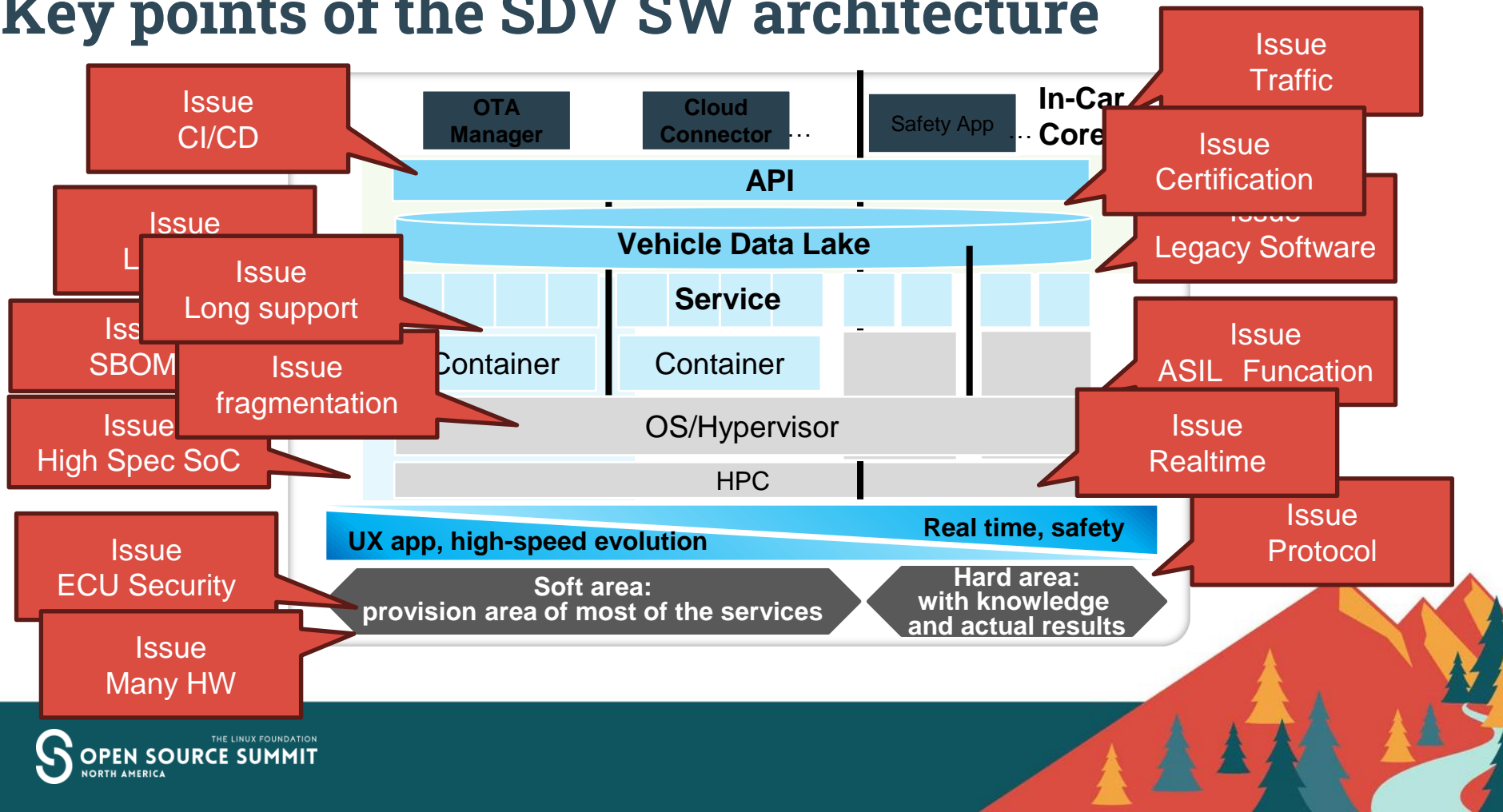
  
Data driven development/  
analysis environment

  
Cross-generation SW  
development  
(shift to one-branch)

# Key points of the SDV



# Key points of the SDV SW architecture





# Issue 1: Use of high-performance HW(SoC)

- An example of a high-performance SoC -

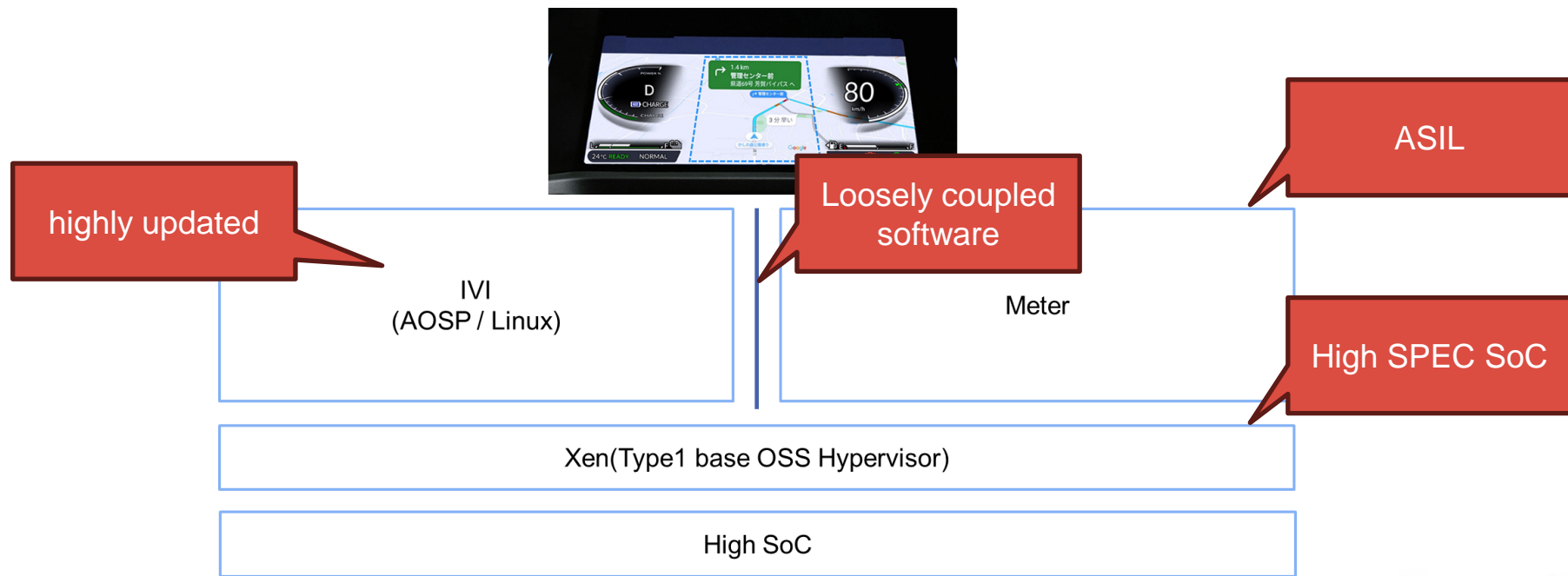
- CPU cores: 32 Arm® Cortex®-A720AE CPU cores delivering over 1,000k DMIPS performance
- Real-time processing: 6 Cortex-R52 CPU cores (lockstep) delivering >60k DMIPS performance and achieving ASIL D
- AI accelerators: up to 400 TOPS of AI accelerators
- GPUs: up to 4 TFLOPS GPUs on board to provide superior graphics processing performance
- Chiplet technology: AI performance and graphics processing performance can be extended by applying chiplet technology.

**SW/architecture needed to bring out HW SPECs comparable to those of high-performance PCs**





## Issue 2: Achieving different application requirements in one SoC



Can applications with different characteristics be updated appropriately while maintaining loose coupling?

# Issue 3: Asset diversion of RTOS-based software

## - Linux Kernel scheduling policy -

### SCHED\_OTHER

the standard round-robin time-sharing policy;

### SCHED\_BATCH

for "batch" style execution of processes; and

### SCHED\_IDLE

for running *very* low priority background jobs.

Various "real-time" policies are also supported, for special time-critical applications that need precise control over the way in which runnable threads are selected for execution. For the rules governing when a process may use these policies, see [sched\(7\)](#). The real-time policies that may be specified in *policy* are:

### SCHED\_FIFO

a first-in, first-out policy; and

### SCHED\_RR

a round-robin policy.

Linux also provides the following policy:

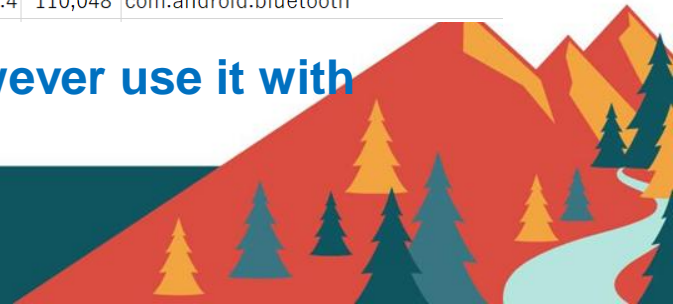
### SCHED\_DEADLINE

a deadline scheduling policy; see [sched\(7\)](#) for details.

## - List of AOSP RT threads -

RTPRI	%CP	%MEM	RSS	NAME
5	0	0	0	[rot_doneq_0_1]
3	0.6	0.3	24,168	android.hardware.audio@2.0-servic
3	4.2	0.3	24,168	android.hardware.audio@2.0-servic
3	0.3	0.3	28,916	audioserver
3	0	0.3	28,916	audioserver
3	3.3	0.3	28,916	audioserver
2	2.2	0.2	20,324	android.hardware.graphics.compose
2	0.2	0.5	39,316	surfaceflinger
2	0.1	0.5	39,316	surfaceflinger
2	0.1	0.5	39,316	surfaceflinger
2	0	0.5	39,316	surfaceflinger
2	0.3	0.2	20,324	android.hardware.graphics.compose
2	0.3	0	0	[kgsl_worker_thr]
2	0	0.2	20,324	android.hardware.graphics.compose
1	0	1.4	110,048	com.android.bluetooth
1	4.5	0.5	39,316	surfaceflinger
1	0	1.4	110,048	com.android.bluetooth

**The Linux Kernel also supports real-time thread, However use it with caution!**



# Issue 4: Consideration of different life cycles

## - EOL for Linux Kernel (6 to 2 years) -

### Longterm release kernels

Version	Maintainer	Released	Projected EOL
6.12	Greg Kroah-Hartman & Sasha Levin	2024-11-17	Dec, 2026
6.6	Greg Kroah-Hartman & Sasha Levin	2023-10-29	Dec, 2026
6.1	Greg Kroah-Hartman & Sasha Levin	2022-12-11	Dec, 2027
5.15	Greg Kroah-Hartman & Sasha Levin	2021-10-31	Dec, 2026
5.10	Greg Kroah-Hartman & Sasha Levin	2020-12-13	Dec, 2026
5.4	Greg Kroah-Hartman & Sasha Levin	2019-11-24	Dec, 2025

## - EOL for ACK(Android Common Kernel) (4 years) -

ACK branch	Launch date	Support lifetime (years)	EOL
android11-5.4	2019-11-24	6	2026-01-01
android12-5.4	2019-11-24	6	2026-01-01
android12-5.10	2020-12-13	6	2027-07-01
android13-5.10	2020-12-13	6	2027-07-01
android13-5.15	2021-10-31	6	2028-07-01
android14-5.15	2021-10-31	6	2028-07-01
android14-6.1	2022-12-11	6	2029-07-01
android15-6.6	2023-10-29	4	2028-07-01
android16-6.12	2024-11-17	4	2029-07-01

Even the SW and version used has a different lifecycle



# Issue 5: Lead time for development environment

## - The long road to SW development -

- NDA
- SoW
- AAA contracts
- BBB contract
- CCC contract

## - An example of development using OSS -

- Clone master

```
mkdir -p AGL/master ; cd AGL/master
repo init -b master -u https://gerrit.automotivelinux.org/gerrit/AGL/AGL-repo
repo sync
```
- Configure for aws-ec2-x86-64 or aws-ec2-arm64

```
source meta-agl/scripts/aglsetup.sh -m aws-ec2-x86-64 agl-demo agl-devel agl-rdp
#or for graviton:
source meta-agl/scripts/aglsetup.sh -m aws-ec2-arm64 agl-demo agl-devel agl-rdp
```
- bitbake e.g. the flutter image

```
bitbake agl-ivi-demo-platform-flutter
```

The lead time of the development environment also affects the speed of SW development.



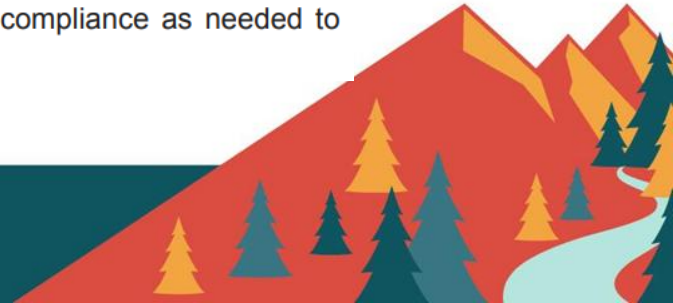
# Issue 6: Automotive Safety Integrity Level(ISO26262)

Regulations on automotive products require compliance with safety standards such as ISO 26262 (“Road Vehicles - Functional Safety”). ISO 26262 defines four Automotive Safety Integrity Levels (ASIL): ASIL A (the least restrictive), ASIL B, ASIL C and ASIL D (the most restrictive). The entire system, its hardware and its software from the virtualization solution up to the applications, needs to be certified to be integrated in production vehicles.

Among the automotive functions presented in Section 3, IVI typically requires ASIL A or no certification, Instrument Cluster and Telematics typically require ASIL B and more advanced functions such as ADAS and digital mirrors require ASIL C or D.

ISO 26262 suggests a classic systems engineering approach to software development and provides regulations and recommendations throughout the product development process from conceptual development through decommissioning. Automotive device makers must document and follow their ISO 26262 certified development process. This process provides checkpoints to ensure correctness of design through comprehensive verification and validation. During the development process artifacts are created at every stage to document product design and verification, track change history of work items, show full traceability of testing based on product requirements, provide proof of process control and report process compliance as needed to certification authorities.

## Complexity of mixed SW from QM to ASIL-D



# Issue 7: Software Bill Of Materials

[android / platform / manifest / refs/heads/android14-qpr3-release / . / default.xml](#)

blob: 13b083f073083156df7c0b23109027b56d491e7d [file] [log] [blame] [edit]

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <manifest>
3
4   <remote name="aosp"
5         fetch=".."
6         review="https://android-review.googlesource.com/" />
7   <default revision="android14-qpr3-release"
8         remote="aosp"
9         sync-j="4" />
10
11   <superproject name="platform/superproject" remote="aosp" revision="android14-qpr3-release"/>
12   <contactinfo bugurl="go/repo-bug" />
13   <project path="build/make" name="platform/build" groups="pdk,sysui-studio" >
14     <linkfile src="CleanSpec.mk" dest="build/CleanSpec.mk" />
15     <linkfile src="buildspec.mk.default" dest="build/buildspec.mk.default" />
16     <linkfile src="core" dest="build/core" />
17     <linkfile src="envsetup.sh" dest="build/envsetup.sh" />
18     <linkfile src="target" dest="build/target" />
19     <linkfile src="tools" dest="build/tools" />
20   </project>
21   <project path="build/orchestrator" name="platform/build/orchestrator" groups="pdk" />
22   <project path="build/bazel" name="platform/build/bazel" groups="pdk" >
23     <linkfile src="bazel.WORKSPACE" dest="WORKSPACE" />
24     <linkfile src="bazel.BUILD" dest="BUILD" />
25   </project>
26   <project path="build/bazel-common-rules" name="platform/build/bazel-common-rules" groups="pdk" />
```

**OSS Repository**  
**58.9%(799/1,357)**

[platform/external/aac](#)  
[platform/external/abi-compliance-checker](#)  
[platform/external/abi-dumper](#)  
[platform/external/abseil-cpp](#)  
[platform/external/accessibility-test-framework](#)  
[platform/external/accompanist](#)  
[platform/external/actionbarsherlock](#)  
[platform/external/adeb](#)  
[platform/external/adhd](#)  
[platform/external/adt-infra](#)  
[platform/external/aeht](#)  
[platform/external/aes](#)  
[platform/external/AFLplusplus](#)  
[platform/external/alac](#)

## 38 Creating a Software Bill of Materials

Once you are able to build an image for your project, once the licenses for each software component are all identified (see "Working With Licenses") and once vulnerability fixes are applied (see "Checking for Vulnerabilities"), the OpenEmbedded build system can generate a description of all the components you used, their licenses, their dependencies, their sources, the changes that were applied to them and the known vulnerabilities that were fixed.

This description is generated in the form of a **Software Bill of Materials** (SBOM), using the [SPDX](#) standard.

When you release software, this is the most standard way to provide information about the Software Supply Chain of your software image and SDK. The **SBOM** tooling is often used to ensure open source license compliance by providing the license texts used in the product which legal departments and end users can read in standardized format.

SBOM information is also critical to performing vulnerability exposure assessments, as all the components used in the Software Supply Chain are listed.

The OpenEmbedded build system doesn't generate such information by default. To make this happen, you must inherit the [create-spdx](#) class from a configuration file:

```
INHERIT += "create-spdx"
```

Upon building an image, you will then get:

- **SPDX** output in JSON format as an `IMAGE-MACHINE.spdx.json` file in `tmp/deploy/images/MACHINE/` inside the **Build Directory**.
- This toplevel file is accompanied by an `IMAGE-MACHINE.spdx.index.json` containing an index of JSON **SPDX** files for individual recipes.
- The compressed archive `IMAGE-MACHINE.spdx.tar.zst` contains the index and the files for the single recipes.

## NOASSERTION: Package Download Location

Package Name	SPDX Identifier	Package Version	Package FileName	Package Supplier	Package Originator	Home Page	Package Download Location
can-utils	SPDXRef-Package-can-utils	2023.03		Organization: OpenEmbedded ()			NOASSERTION



# Cooperation with the OSS community



# Lots of OSS communities involved in SDV



Automotive Grade  
Linux (AGL)



**ELISA**  
Enabling **Linux** in  
**Safety** Applications



**Zephyr**


# What is Automotive Grade Linux?

Automotive Grade Linux is a collaborative, open source project that brings together automakers, suppliers, and technology companies for the purpose of building Linux-based, open source software platforms for automotive applications that can serve as de facto industry standards.

AGL address all software in the vehicle: infotainment, instrument cluster, heads-up-display (HUD), telematics, connected car, advanced driver assistance systems (ADAS), functional safety, and autonomous driving.

Adopting a shared platform across the industry reduces fragmentation and allows automakers and suppliers to reuse the same code base, which leads to rapid innovation and faster time-to-market for new products.

AGL is a Linux Foundation project and its goals are as follows:

- 
- Build a single platform for the entire industry
  - Develop 70 to 80% of the starting point for a production project
  - Reduce fragmentation by combining the best of open source
  - Develop an ecosystem of developers, suppliers, and expertise that all use a single platform

You can find additional overview information on the "[About Automotive Grade Linux](#)" page. You can find information on the AGL Unified Code Base on the "[Unified Code Base](#)" page.

- VirtIO available in UCB for hypervisor use cases
- Non-hypervisor (loopback) use cases complete for Quillback
- Unified HMI – Virtual display used by different ECUs based on VirtIO (Panasonic)
- ***SDV reference PF under development***

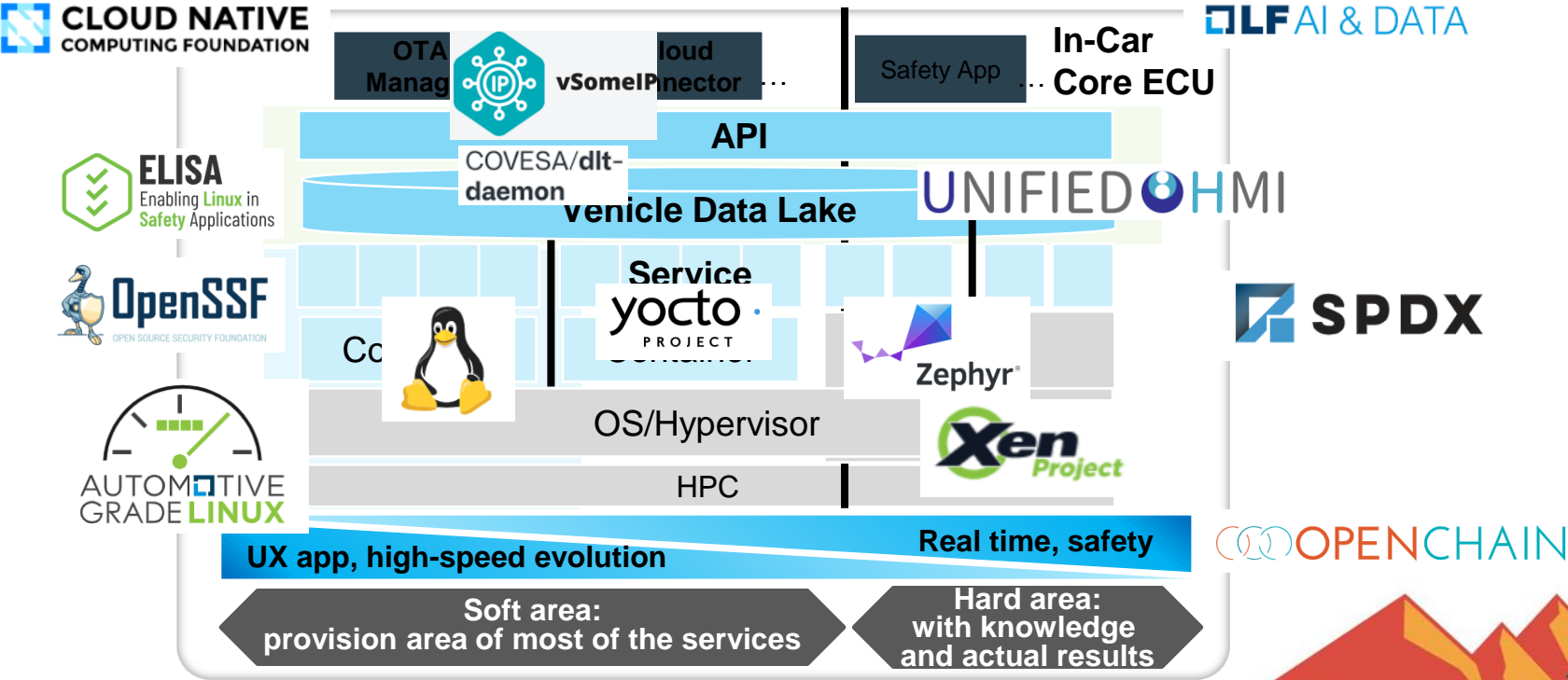
## Charter

The AGL Software Defined Vehicle (SDV) Expert Group (SDV-EG) (formerly known as Virtualization and Containers Expert Group (EG-VIRT)) is responsible to design and implement virtualization solutions for AGL. Containers, Hypervisors (both based on Virtualization Extensions and TrustZone) and any other virtualization solution for x86/ARM are considered to be of interest for this Expert Group. The AGL Unified Code Base (UCB) supports the KVM hypervisor on the Renesas RCar M3 platform.

The SDV-VIRT expert group has been focused on defining the Virtualization platform architecture of AGL since 2018. One result of the work is this [white paper](https://lf-automotivelinux.atlassian.net/wiki/spaces/VE/overview)

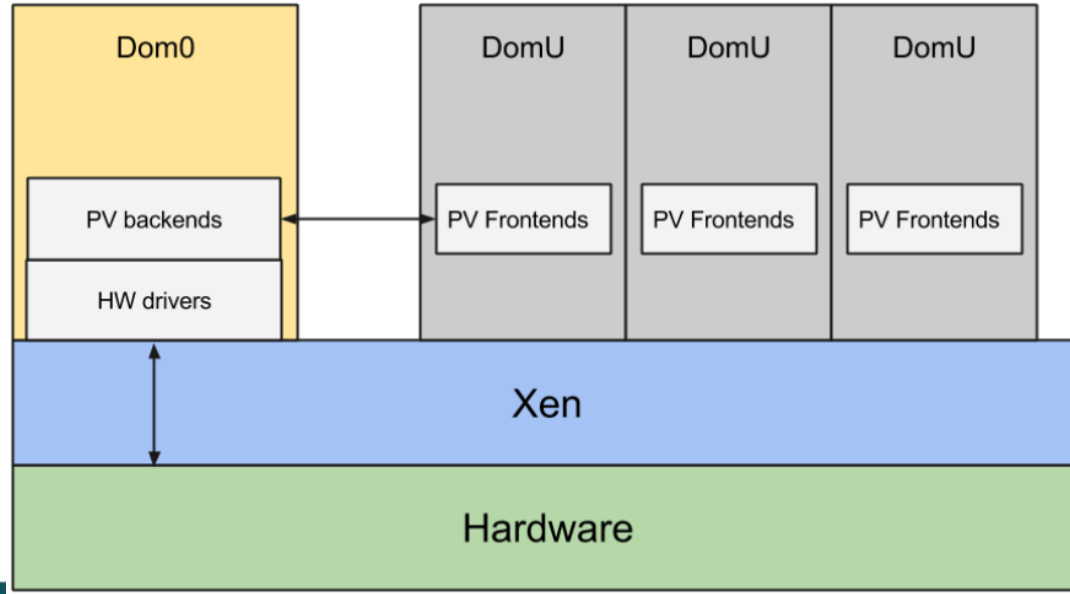


# Mapping of SDV architecture and OSS



# Hypervisor: Xen

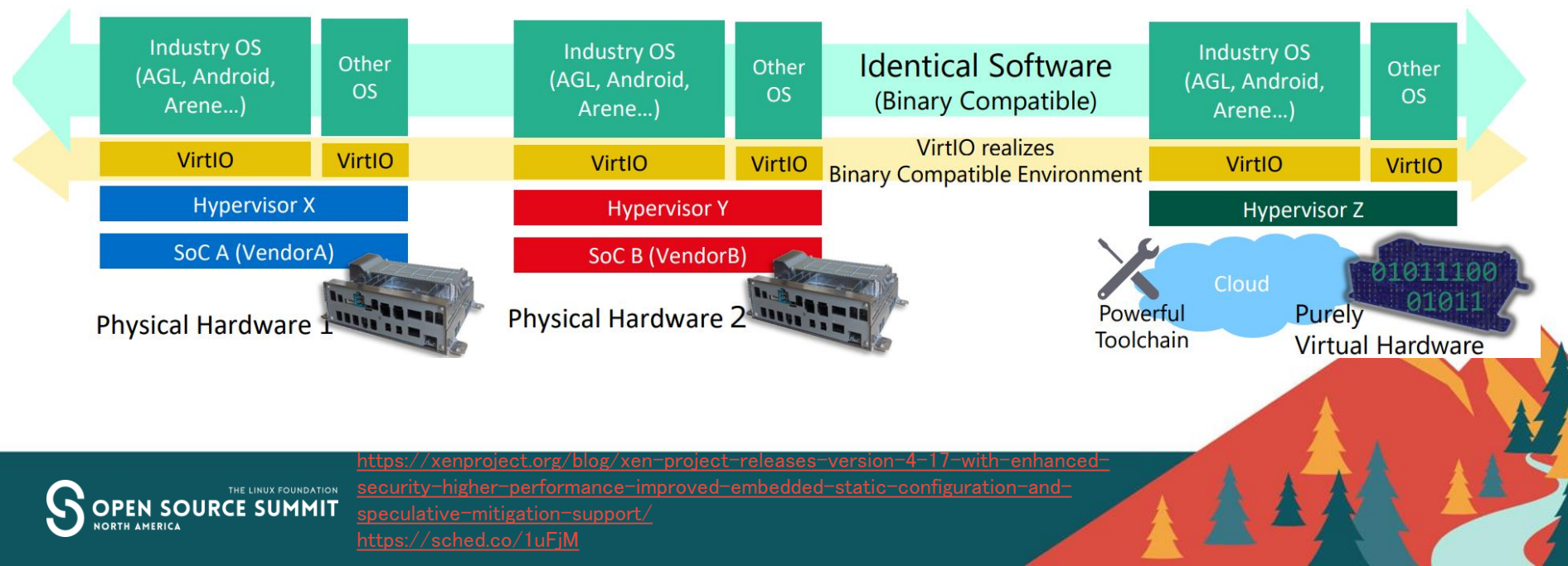
Xen is type-1 hypervisor: it runs directly on the hardware, everything else in the system is running as a virtual machine on top of Xen, including Dom0, the first virtual machine. Dom0 is created by Xen, is privileged and drives the devices on the platform. Xen virtualizes CPU, memory, interrupts and timers, providing virtual machines with one or more virtual CPUs, a fraction of the memory of the system, a virtual interrupt controller and a virtual timer





# VirtIO support for Xen

ARM: Add “tech preview” implementation for VirtIO. Xen now includes full support for VirtIO on embedded systems, on ARM, for the virtio-mmio transport, allowing a wide range of VirtIO devices to be supported. This includes front-end support in Linux, toolstack (libxl/xl) and dom0less support, and a userspace backend. Currently, the following stand-alone backends are available and have been tested: virtio-disk, virtio-net, i2c, and gpio.



# Xen ASIL(ISO26262)

## XEN HYPERVISOR SAFETY CERTIFICATION PLAN

Safety Manager: Senthil Rajagopal

### Xen Hypervisor software safety certification

ISO 26262:2018, Element level ASIL D  
IEC 61508, Edition 2 Systematic Capability 3 (SIL 3)

#### Phase I

Safety  
Concept  
Review



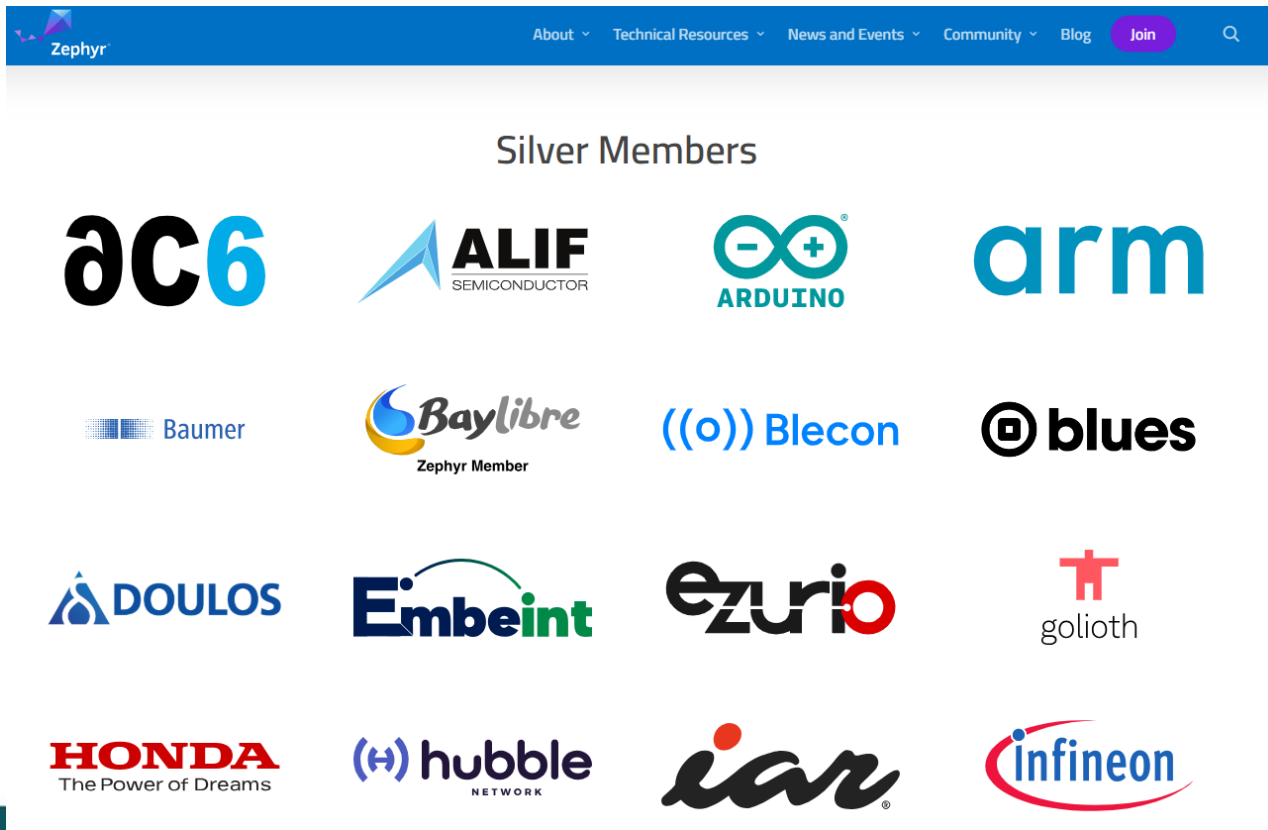
#### Phase II

Final  
Assessment

Completed November 2024!



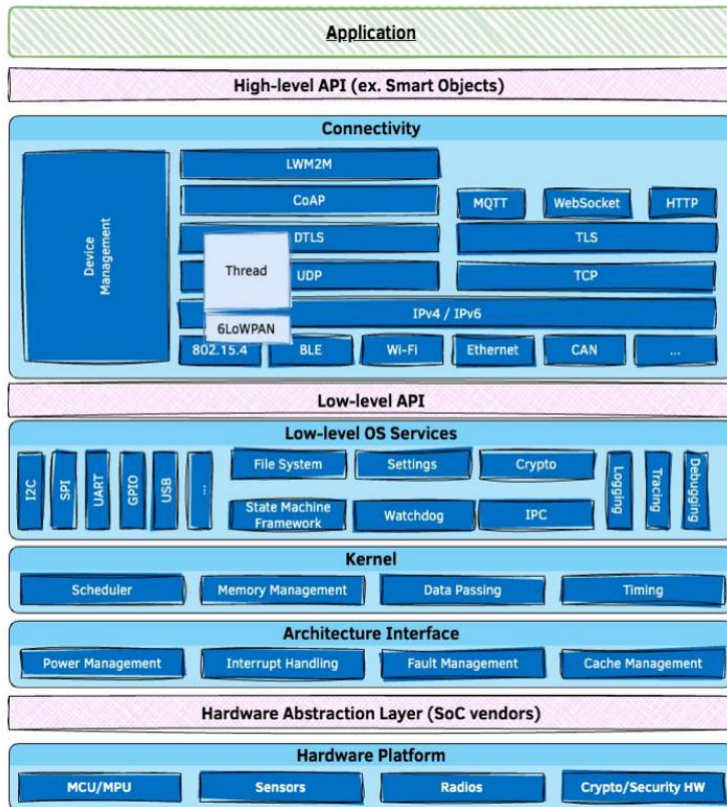
# RTOS: Zephyr



# RTOS: Zephyr

## Architecture

- Lightweight kernel & supporting drivers and services
- Portable, secure, power-efficient
- Highly connected
  - Bluetooth 5.0 & BLE
  - Wi-Fi, Ethernet, CANbus, ...
  - IoT protocols: CoAP, LwM2M, MQTT, OpenThread, ...
  - USB & USB-C



# Zephyr ASIL(ISO26262)



## Zephyr: Journey to Safety Certification

Kate Stewart, Zephyr Project Director, The Linux Foundation  
Nicole Pappler, Zephyr Functional Safety Manager, AlektoMetis

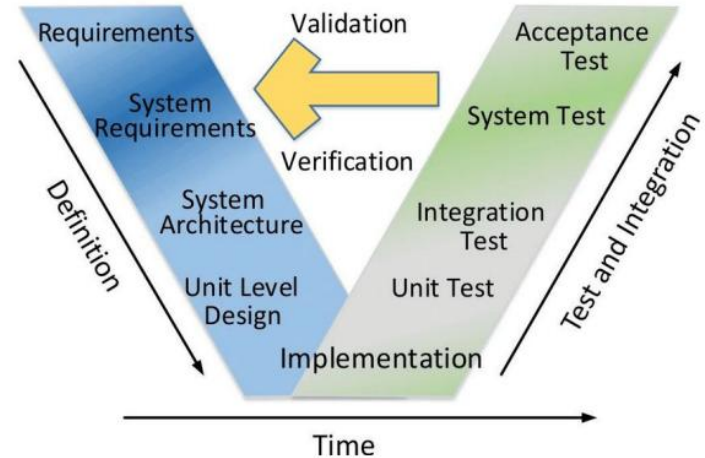


# Zephyr ASIL(ISO26262)

## Safety Engineering 101: the "V-Model"



- The V-model for functional safety development originated from systems engineering practices to structure the process of designing and validating complex systems.
- It was later adapted and widely adopted in the automotive industry and other safety-critical sectors as a framework for ensuring the systematic integration of safety requirements and processes throughout each stage of the development lifecycle.
  - **ISO 26262** in the automotive industry
  - **IEC 61508** for industrial systems
  - **DO-178C** in aerospace



Source Image image provided under CC-4.0  
[https://www.researchgate.net/figure/The-functional-safety-development-via-V-model-14\\_fig4\\_362572593](https://www.researchgate.net/figure/The-functional-safety-development-via-V-model-14_fig4_362572593)



# Zephyr ASIL(ISO26262)

## Coding Guidelines

### Main rules

The coding guideline rules are based on MISRA-C 2012 and are a **subset** of MISRA-C. The subset is listed in the table below with a summary of the rules, its MISRA-C severity and the equivalent rules from other standards for reference.

The severity and other references in the table below are for informational purposes only. The listed rules are all required for Zephyr and all new code should comply with the rules listed below.

Main rules				
Zephyr rule	Description	MISRA-C 2012 rule	MISRA-C severity	CERT C reference
1	Any implementation-defined behaviour on which the output of the program depends shall be documented and understood	Dir 1.1 <a href="#">↗</a>	Required	MSC09-C <a href="#">↗</a>
2	All source files shall compile without any compilation errors	Dir 2.1 <a href="#">↗</a>	Required	N/A
3	All code shall be traceable to documented requirements	Dir 3.1 <a href="#">↗</a>	Required	N/A
4	Run-time failures shall be minimized	Dir 4.1 <a href="#">↗</a>	Required	N/A
5	All usage of assembly language should be documented	Dir 4.2 <a href="#">↗</a>	Advisory	N/A

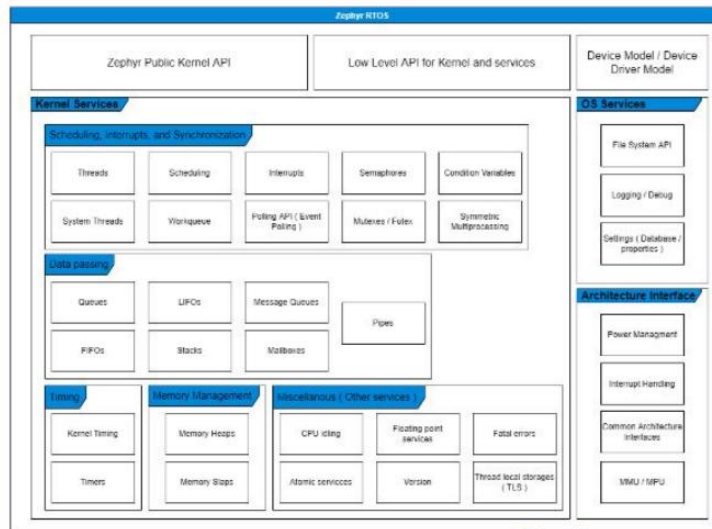


# Zephyr ASIL(ISO26262)

## Zephyr Initial Certification Focus



- Start with a limited scope of kernel and interfaces
- Initial target is IEC 61508 SIL 3 / SC 3 (IEC 61508-3, 7.4.2.12, Route 3s)
- Option for 26262 certification has been included in contract with certification authority should there be sufficient member interest



Starting scope

**Scope** can be **extended** to include **additional components** with associated **requirements** and **traceability** as determined by the safety committee

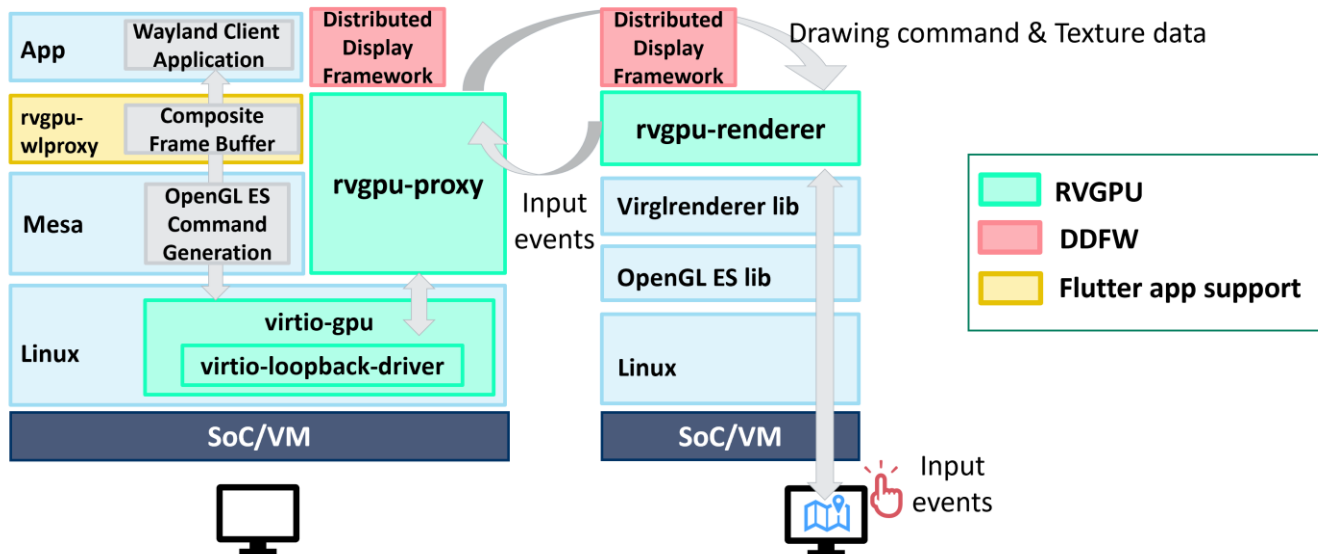
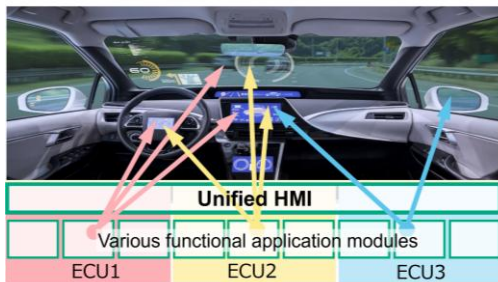
# Unified HMI

"Unified HMI" is a Software-Defined display virtualization platform based on VirtIO GPU technology. Unified HMI allows for flexible development of the entire cockpit and cabin UI/UX, across multiple displays, independent of hardware and OS configuration.

Without Unified HMI



With Unified HMI



# VSOMEIP



## vSomeIP

### Overview:

The vSomeIP stack implements the <http://some-ip.com/> (Scalable Service-Oriented Middleware over IP (SOME/IP)) protocol. The stack consists of:

- a shared library for SOME/IP (`libvsomeip3.so`)
- a shared library for SOME/IP's configuration module (`libvsomeip3-cfg.so`)
- a shared library for SOME/IP's service discovery (`libvsomeip3-sd.so`)
- a shared library for SOME/IP's E2E protection module (`libvsomeip3-e2e.so`)

### Optional:

- a shared library for compatibility with vsomeip v2 (`libvsomeip.so`)

### Lead:

Gonalo Almeida (Critical Techworks), Diogo Pedrozza (Critical Techworks)

### Resources:

- [Project page](#)

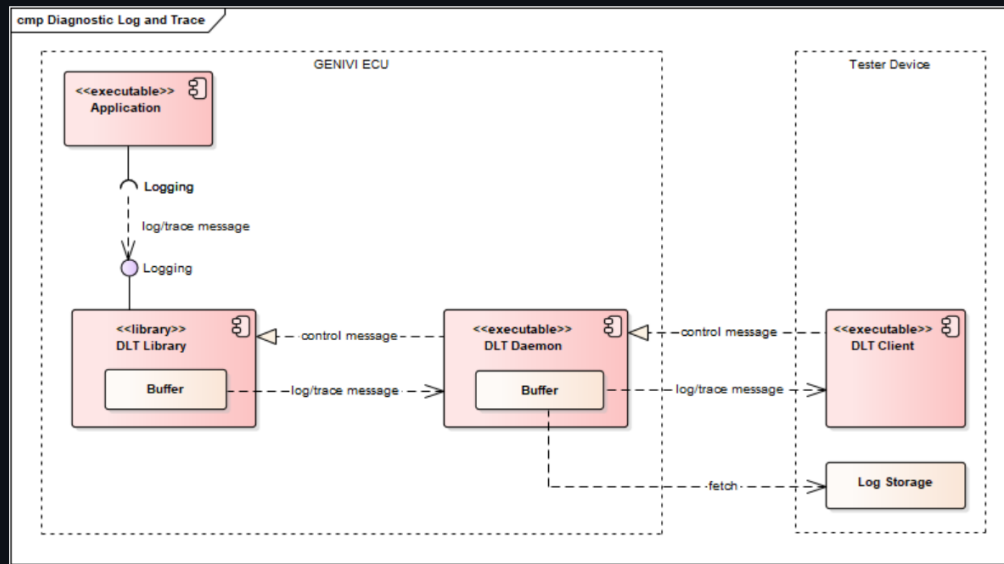


# DLT(Diagnostic Log and Trace)

## Overview

COVESA DLT provides a log and trace interface, based on the standardised protocol specified in the [AUTOSAR Classic Platform R19-11 DLT](#). It is used by other COVESA components but can serve as logging framework for other applications without relation to COVESA.

The most important terms and parts are depicted in the following figure. Please refer to [Glossary](#) for a full overview over DLT-specific terms.



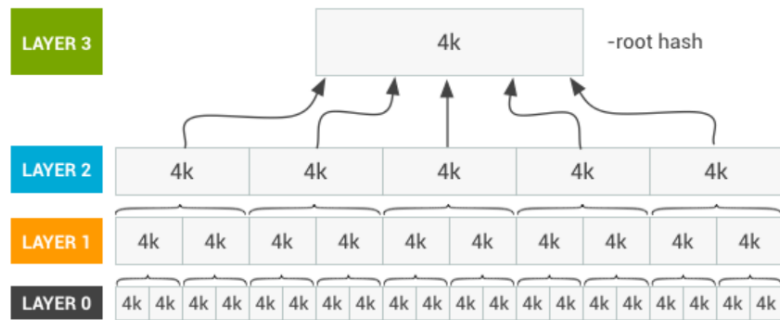
# Verified Boot

Android 4.4 and higher supports Verified Boot through the optional device-mapper-verity (dm-verity) kernel feature, which provides transparent integrity checking of block devices. dm-verity helps prevent persistent rootkits that can hold onto root privileges and compromise devices. This feature helps Android users be sure when booting a device it is in the same state as when it was last used.

Potentially Harmful Applications (PHAs) with root privileges can hide from detection programs and otherwise mask themselves. The rooting software can do this because it is often more privileged than the detectors, enabling the software to "lie" to the detection programs.

The dm-verity feature lets you look at a block device, the underlying storage layer of the file system, and determine if it matches its expected configuration. It does this using a cryptographic hash tree. For every block (typically 4k), there is a SHA256 hash.

Because the hash values are stored in a tree of pages, only the top-level "root" hash must be trusted to verify the rest of the tree. The ability to modify any of the blocks would be equivalent to breaking the cryptographic hash. See the following diagram for a depiction of this structure.





# MACSec

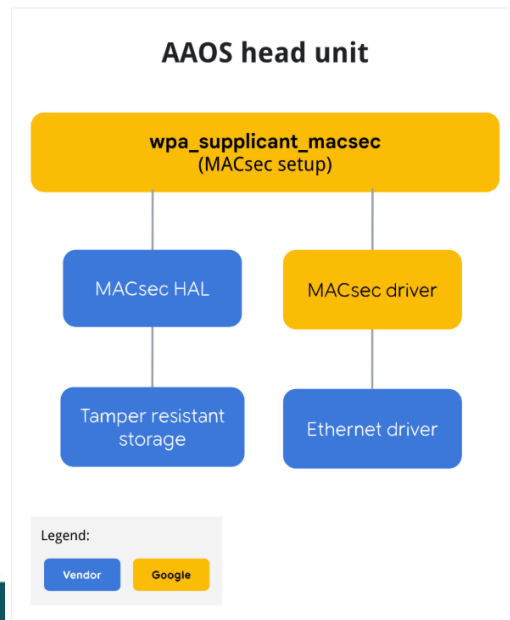
## Enable MACsec for Ethernet features

This page explains how to enable MACsec for Ethernet features.

Use MACsec to authenticate and encrypt the Ethernet communication used by in-vehicle infotainment (IVI) for different ECU units, protecting the data from tampering, replay, or information disclosure. Do this by enabling MACsec IEEE 802.11AE for the Ethernet network.

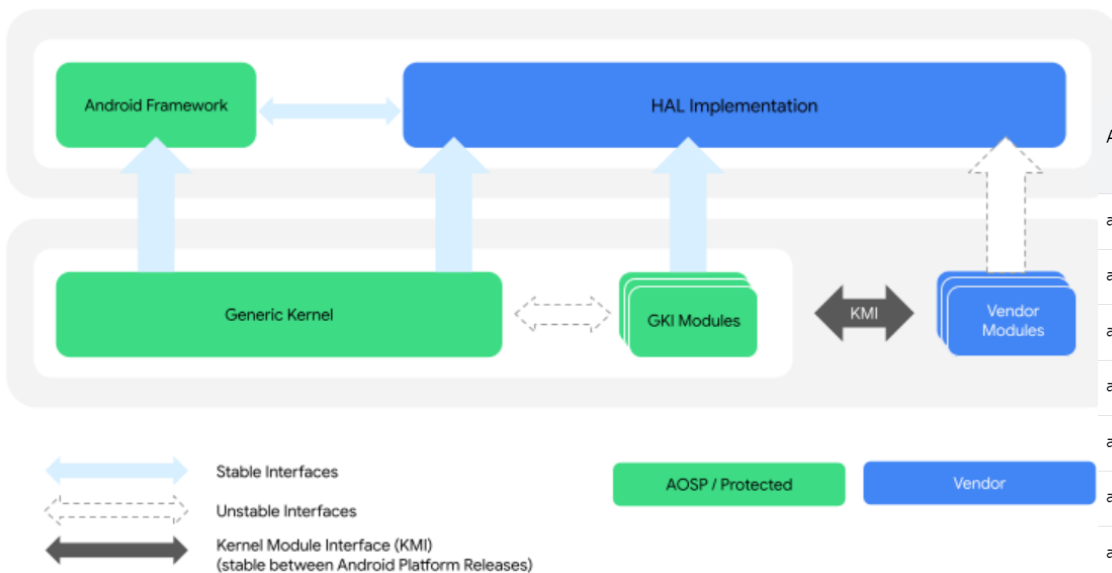
### Overview

To enable MACsec, `wpa_supplicant` is used as the daemon for handling the MACsec Key Agreement (MKA) handshake. A MACsec HAL is defined for storing the MACsec pre-shared-key, called the connectivity association key (CAK) securely. The MACsec HAL only supports CAK. This vendor-specific MACsec HAL stores the CAK securely in a tamper resistant storage. Provisioning the key depends on the vendor implementation.



# GKI(Generic Kernel Image)

The GKI kernel interacts with hardware-specific *vendor modules* containing system on a chip (SoC) and board-specific code. The interaction between the GKI kernel and vendor modules is enabled by the *Kernel/Module Interface (KMI)* consisting of symbol lists identifying the functions and global data required by vendor modules. Figure 1 shows the GKI kernel and vendor module architecture:



ACK branch	Launch date	Support lifetime (years)	EOL
android11-5.4	2019-11-24	6	2026-01-01
android12-5.4	2019-11-24	6	2026-01-01
android12-5.10	2020-12-13	6	2027-07-01
android13-5.10	2020-12-13	6	2027-07-01
android13-5.15	2021-10-31	6	2028-07-01
android14-5.15	2021-10-31	6	2028-07-01
android14-6.1	2022-12-11	6	2029-07-01
android15-6.6	2023-10-29	4	2028-07-01
android16-6.12	2024-11-17	4	2029-07-01

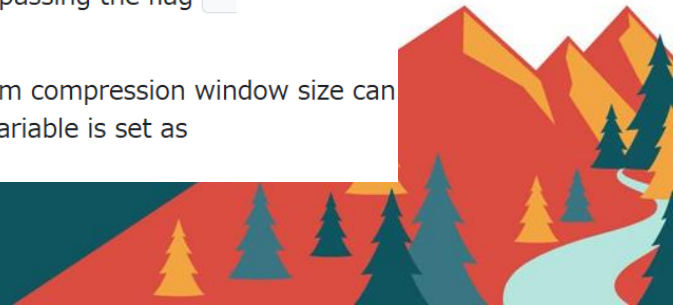
# A/B Software update

Android OTA updates occasionally contain changed files that don't correspond to code changes. They're actually build system artifacts. This can occur when the same code, built at different times, from different directories, or on different machines produces a large number of changed files. Such excess files increase the size of an OTA patch, and make it difficult to determine which code changed.

To make the contents of an OTA more transparent, AOSP includes build system changes designed to reduce the size of OTA patches. Unnecessary file changes between builds have been eliminated, and only patch-related files are contained in OTA updates. AOSP also includes a [build diff tool](#), which filters out common build-related file changes to provide a cleaner build file diff, and a [block mapping tool](#), which helps you keep block allocation consistent.

A build system can create unnecessarily large patches in several ways. To mitigate this, in Android 8.0 and higher, new features were implemented to reduce the patch size for each file diff. Improvements that reduced OTA-update package sizes include the following:

- Usage of **ZSTD**, a generic-purpose, lossless-compression algorithm for full images on non-A/B device updates. **ZSTD** can be customized for higher compression ratios by increasing compression level. Compression level is set during OTA generation time and can be set by passing the flag `--vabc_compression_param=zstd,$COMPRESSION_LEVEL`
- Increasing the compression window size used during OTA. The maximum compression window size can be set by customizing the build parameter in a device's `.mk` file. This variable is set as `PRODUCT_VIRTUAL_AB_COMPRESSION_FACTOR := 262144`



# SBOM(SPDX and SPDX Tools)



**Data: Challenges**

**Format**

SPDX 3.0

{JSON}

**Elements**

**SPDX LITE**

In consideration with

- NTIA minimum elements
- Auto-ISAC's SBOM report
- etc.

**Dependencies**

**Semi-Automation**

Tool Generates SBOM → Policy-based Evaluation → Review and Modified

**Procedure**

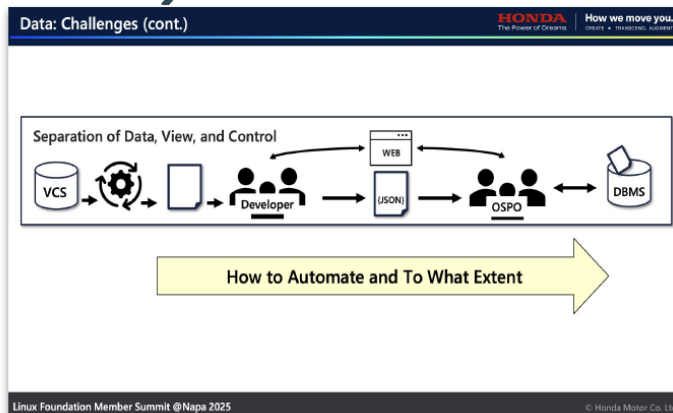
PLAN → PREPARE → DEVELOP → BUILD → TEST → RELEASE → RETIRE → CONFIGURE → MAINTAIN → MONITOR (OBSERVE)

Open source SBOM, Source SBOM, Build SBOM, Deployed SBOM, Runtime SBOM

To catch the dependencies of the deliverables

Linux Foundation Member Summit @Napa 2025

© Honda Motor Co. Ltd.



**Governance challenges: Risk management & efficiency**

- Implementing in other development teams (e.g. ECUs and NPUs)
- Aligning process and toolchain with standards (e.g.: UNR155/156, UNECE WP.29, ISO/SAE 21432, ISO/IEC 26262)
- Automated policy-based workflow

ISO/IEC 5230	ISO/IEC 19974	Functional Safety	SUMS & OTA	Automate Software Asset Management, License Compliance, and Security Assurance
Program foundation		Design Phase Test Phase	Cloud icon Car icon	VCS → [ ] → Developer → [JSON] → OSPO → DBMS
Relevant tasks defined and supported				
Open source: current review and approval				
Compliance artifact creation and delivery				
Understanding: open source community engagement				
Adherence to the specification requirements				

Linux Foundation Member Summit @Napa 2025

© Honda Motor Co. Ltd.

**SBOM Quality**

Completeness × Accuracy → Consistency in Supply chain

- Dependency Depth
- Component Info
- Relationships
- NO "No-Assertion"
- Canonical or Normalized values
- License and Copyright Info
- Relationships

Case Examples require additional effort

1. Incomplete and/or Inconsistent value
  - Likely when different tools are used in suppliers
2. License and copyright Info
  - Updates or something may require source code analysis and tracking changes
3. Transitive dependencies
  - Some tools may lack transitive dependencies

Linux Foundation Member Summit @Napa 2025

© Honda Motor Co. Ltd.

?

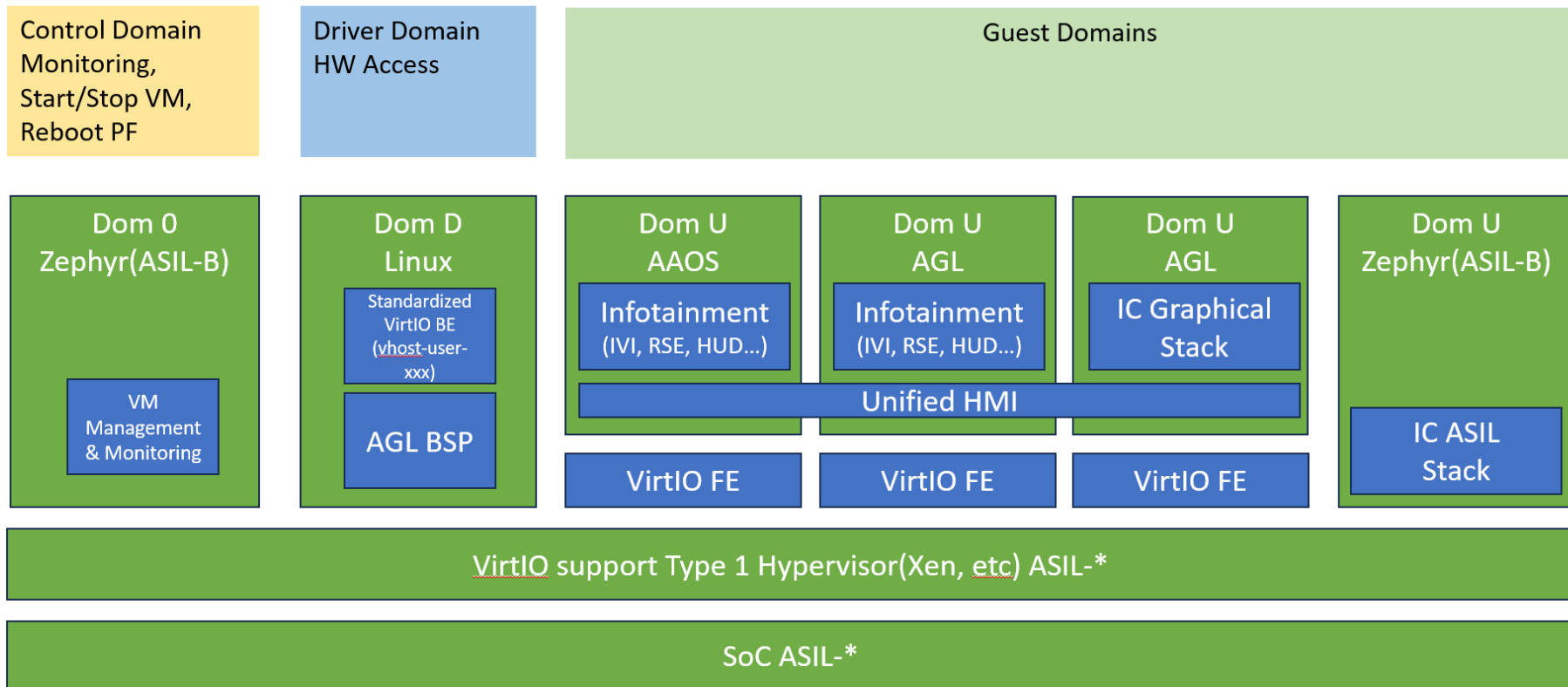


# AGL announced a SDV Reference PF named SoDeV





# SoDeV Architecture Overview(SoC ASIL-A/B)



# SDV Reference PF PoC



Cluster App and Backend on AGL

HW in-depend

AGL Cluster

Guest Linux



AGL UCB

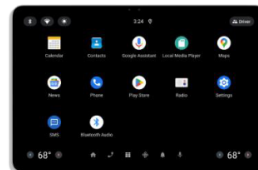
App

VIRTIO Backend

Guest AOSP

Android System

App



Trout based AAOS

HW in-depend

AAOS 15

Full virtual(CPU, MMU, etc.), Para-virtual support by Xen virtualized system

HW depend

Xen

Guest Linux

(Control Domain)

BSP drivers

Linux kernel (BSP part)

VirtIO Frontend

Linux kernel (GKI part)

Virtual Machine

CPU	Timer
memory	GIC

Virtual Machine

CPU	Timer
memory	GIC

Virtual Machine

CPU	Timer
memory	GIC

Xen hypervisor

Full Virtualized

Scheduler

Timer

MMU

interrupt

Cortex®-A57 (x4) + Cortex®-A53 (x4)

R-Car H3 Starter Kit (2 display output by extension)



# Conclusion



# Conclusion

- There are many many challenges to achieving SDV
- It is impossible to achieve it alone
- However, the OSS community already has many solutions
- We need to move forward together with the OSS community

**SoDeV in the AGL SDV Reference PF is one of the keys**



# Additional information



# ELISA WS Volvo Cars@Lund



Title	Presenter(s)
Welcome & Introductions	Philipp Ahmann, ETAS GmbH; Kate Stewart, Linux Foundation; Robert Fekete, Volvo Cars
Ask Me Anything about ELISA or Use of OSS in Safety Critical Applications	Philipp Ahmann, ETAS GmbH; Gabriele Paoloni, Red Hat
Arduino Portenta X8 as a community reference hardware for safe systems	David Cuartielles, Arduino

Example System within ELISA as Cross Community Effort with AGL, Eclipse S-Core, KernelCI, Xen, Zephyr, and more

Philipp Ahmann, ETAS GmbH; Yuichi Kusakabe, Honda Motors

Interaction between ELISA and Adjacent Communities such as Eclipse, Linaro, Rust, SPDX, Yocto, and more

Kate Stewart, Linux Foundation; Philipp Ahmann, ETAS GmbH

Safety Linux vs Safe(ty) Linux

Philipp Ahmann, ETAS GmbH; Paul Albertella, Codethink

How far do we go at the hardware level? An analysis of current state of kernel and integration

Olivier Charrier, Wind River; Alessandro Carminati, Red Hat

PX4Space

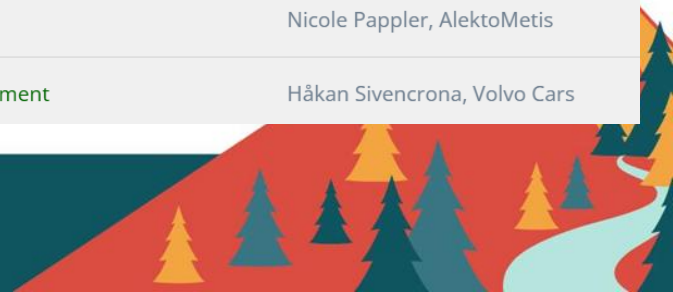
Pedro Roque, KTH Royal Institute of Technology

SPDX Safety Profile

Nicole Pappler, AlektoMetis

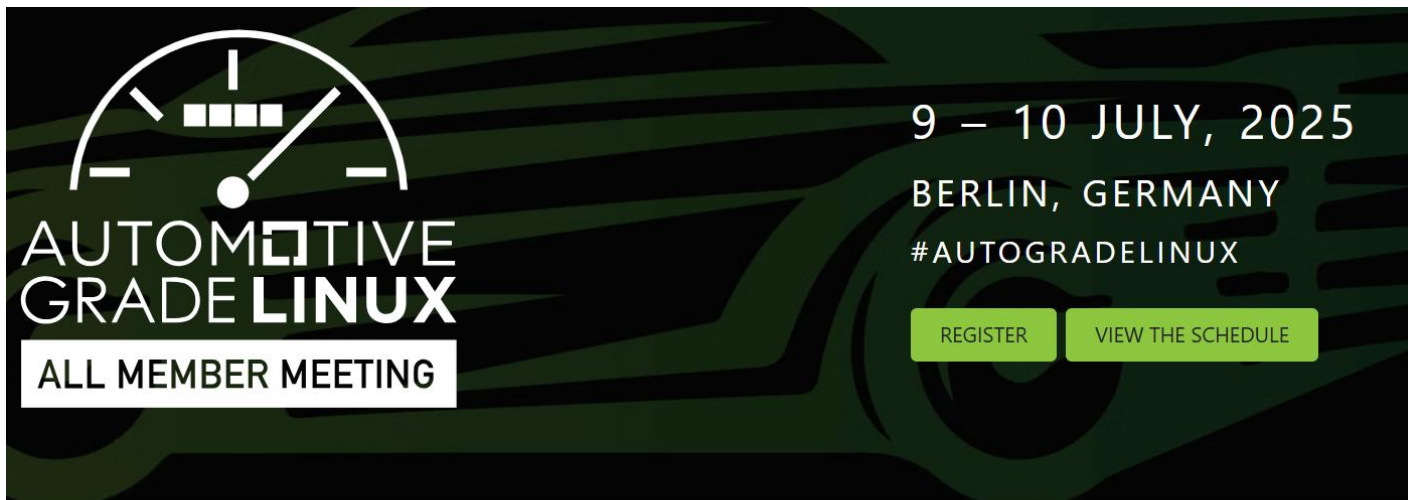
Safe Continuous Deployment

Håkan Sivencrona, Volvo Cars





# AGL All Member Meeting Summer@Berlin



A banner for the AGL All Member Meeting Summer@Berlin. The background is dark green with a stylized car wheel and tire pattern. On the left, there is a white speedometer icon with a needle pointing to the right. Below the speedometer, the text 'AUTOMOTIVE GRADE LINUX' is written in white, with 'AUTOMOTIVE' and 'GRADE' in a sans-serif font and 'LINUX' in a bold, stylized font. Below this, a white rectangular box contains the text 'ALL MEMBER MEETING' in black. On the right side of the banner, the dates '9 – 10 JULY, 2025' are written in white, followed by 'BERLIN, GERMANY' and the hashtag '#AUTOGRADELINUX'. At the bottom right, there are two green buttons: 'REGISTER' and 'VIEW THE SCHEDULE'.

9 – 10 JULY, 2025  
BERLIN, GERMANY  
#AUTOGRADELINUX

REGISTER VIEW THE SCHEDULE



Weeks



Days



Hours



Minutes



Seconds

The Automotive Grade Linux (AGL) All Member Meeting takes place bi-annually and brings the AGL community together to learn about the latest developments, share best practices and collaborate to drive rapid innovation across the industry.



# Xen Summit@ San Jose

## Xen Summit

Xen Summit is our annual event where the community connects with experts, discovers innovations, and shapes the future of virtualization.



# Open Source Summit Japan



**2025 DECEMBER 8-10**  
**TOKYO, JAPAN**  
**#OSSummit**

*The Crossroads of Code, Community, and Corporate  
Open Source*

REGISTER

SPONSOR

SUBMIT TO SPEAK

## ZEPHYR



The Zephyr track is for developers using or considering Zephyr in embedded products. Sessions will explore project advancements, security, tooling, and real-world applications across industries.

**Thank you very much**

**Let's build the future together**

