ANDROID AUTOMOTIVE EMBEDDED OS

INTRODUCTION BY P3

AUTHOR

PETER GESSLER
Android Automotive Architect
Peter.Gessler@p3-group.com

ABSTRACT

Google's operating system Android Automotive Embedded OS for connected in-vehicle infotainment (IVI) systems is already disrupting the traditional automotive infotainment landscape. In this technical white paper, we will give an overview of Android Automotive Embedded OS features and architecture to support the decision-making process for Original Equipment Manufacturers' (OEMs) and Tier 1 suppliers' concerning their future infotainment strategies.

KEYWORDS

Android Automotive, Operating System (OS), Google Automotive Services (GAS), Human-Machine Interface (HMI), In-Vehicle Infotainment (IVI), User Experience (UX)

I. INTRODUCTION

Today's users demand from cars' IVIs and connected services the same intuitive and exciting experience they are used to from their favorite consumer electronic devices, apps and cloud services. Furthermore, they expect their personal application eco-system to be integrates into the vehicle. All of this can now be achieved much easier with Google's Android Automotive OS.

Worldwide, vehicle manufacturers are today evaluating the benefits of Android Automotive Embedded OS carefully. Some have already chosen to enter a formal partnership with Google to co-create their next generation IVI incl. Google Automotive Services (GAS). Others are using the open source project AOSP incl. the car extensions to build an Android Automotive System independently while a third fraction is still hesitating mostly due to concerns regarding dependencies and data ownership.

For those currently at the decision-making crossroad we want to shed light on some of the more technical aspects of Google's Android Automotive OS. The paper is however not trying to be exhaustive in tackling all technical aspects but rather aims at giving an overview. For a more deep-dive discussion we recommend our Android Automotive Base Training.

II. FEATURE OVERVIEW

In order to understand the individual components and the added value of the operating system, we want to give a brief overview of its structure.

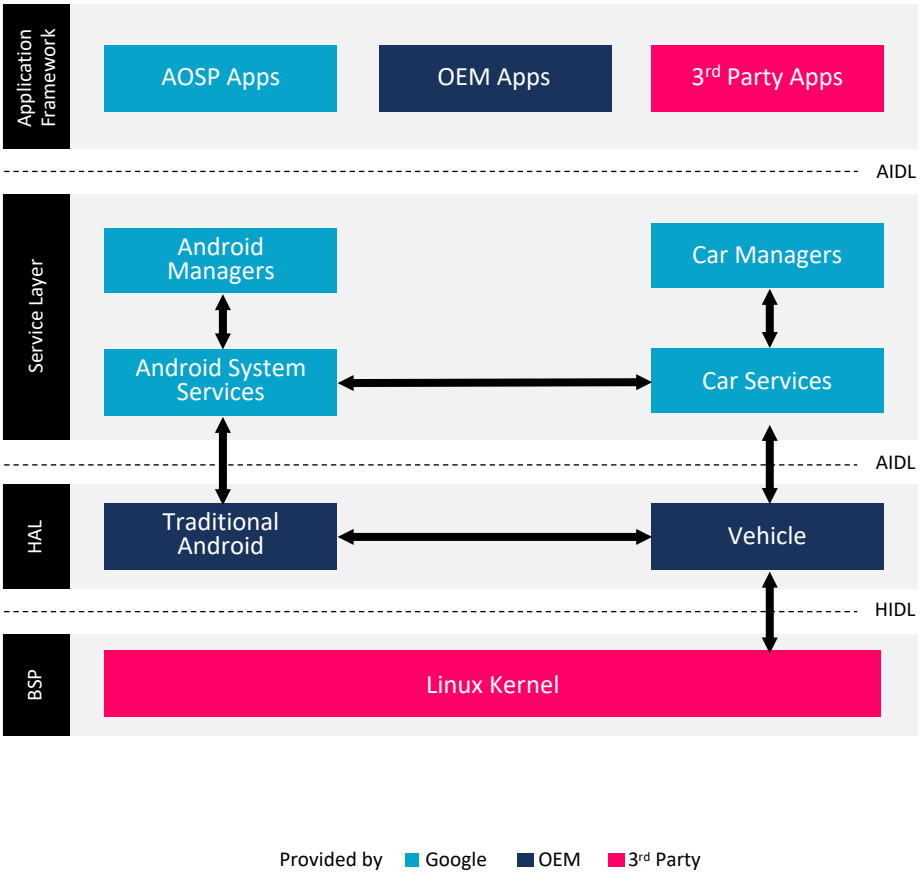

Figure 1: Android Automotive architecture component view. Based on [1].

II. FEATURE OVERVIEW

Figure 1 shows the abstract layer architecture of Android Automotive with the division into four layers. In this section, we focus on GAS and the built-in applications.

Google Automotive Services (GAS).

GAS describes a set of customer-specific and technical services that are precompiled by Google and provided through a licensing model. The most important services are

- **Google Maps & Navigation**: For navigation from point A to point B with intelligent address, route, petrol station and charging station search.
- **Google Assistant**: Voice personal assistant for controlling various vehicle functionalities (can be extended) or give additional information to the user.
- **Google Playstore**: Provision and management of 3rd party applications that are tailored to be used in the vehicle.
- **SetupWizard**: Creation of vehicle user profile accounts and connectivity setup.
- **Automotive Keyboard**: A keyboard adapted for the automotive industry to operate the touchscreen and support various languages.

The OEM receives access to GAS through an associated partnership with Google. This provides access to close communication & support, extended technical documentation as well as the quarter pre-release (QPR) versions with new updates and upgrades.

Non-GAS describes a platform version that does not require the integration of GAS. The OEM simply downloads the freely available AOSP source code with car extensions and integrates its own applications and services. You would choose this variant for example in case of a planned launch in China, due to non-availability of Google services in this market, or if you are a Tier 1 supplier without OEM contract as Google currently only partners directly with OEMs.

Hero applications. Besides GAS, Google is developing applications such as

- **Media Center**: Skeleton for the integration of media sources such as the LocalMediaPlayer. The skeleton is fully integrated and interacts seamlessly with the Notification center and the Dialer.
- **Dialer**: The central telephone application, which allows the contacts of the connected smartphone to be managed and calls to be made.
- **Car Settings**: Management of various system settings such as Time & Languages, User Management and Connectivity.
- **Notification Center**: Brief system- notifications for the user and interactions to start applications.

These applications are available at android.googlesource.com. In addition to these vehicle-specific applications, numerous other applications are available at *... / package / apps /...*

applications and implement general rules and restrictions that apply to all system and user applications.

UI Frameworks The SystemUI / CarSystemUI manage the general structure of the central screen. The user can customize these if necessary and change the individual fragments of the bars and their content (e.g. StatusBar at the top of the screen, global NavigationBar at the bottom as well as main fragment and HVAC bar). Furthermore, the OEM/Tier 1 can manage the theming (use of colors, fonts and styles) and the display of pop-ups via the SystemUI.

Google defines the SystemUI as "...a persistent process that provides UI for the system but outside of the system_server process" [2]. The SystemUIApplication extends the SystemUI with a defined set of services, for example the SystemBars, PowerUI or self-designed services that work in an isolated way, which are a major part of the system user interface and starts with the boot process.

One of the most important extensions to Android Automotive is the DrivingUxRestrictions framework. This is already integrated into the applications provided by Google. The framework uses the configuration file specified by the OEM to prevent touch interaction by the end customer in certain driving situations so that the user is not distracted. The OEM can extend and customize the existing framework therein.

Car-lib: In addition to the functions described for the HMI, there are countless others on the other layers that are provided by Google. We want to point out three special services [2] which reduced a lot of work for us.

CarInfoManager: Depending on the development strategy, the OEM may want to manage multiple vehicle variants with one platform version. The CarInfoManager can be used to dynamically adapt the HMI. As a proxy component, this provides the static information regarding the vehicle model, variant and other relevant vehicle properties.

CarPowerManager: The behavior of the infotainment system and its applications largely depends on the system state of the vehicle. These communicate via the CarPowerManager with the Vehicle HAL and the Vehicle Microcontroller Unit (VMCU) based on a generic state machine, which is displayed in Figure 2.

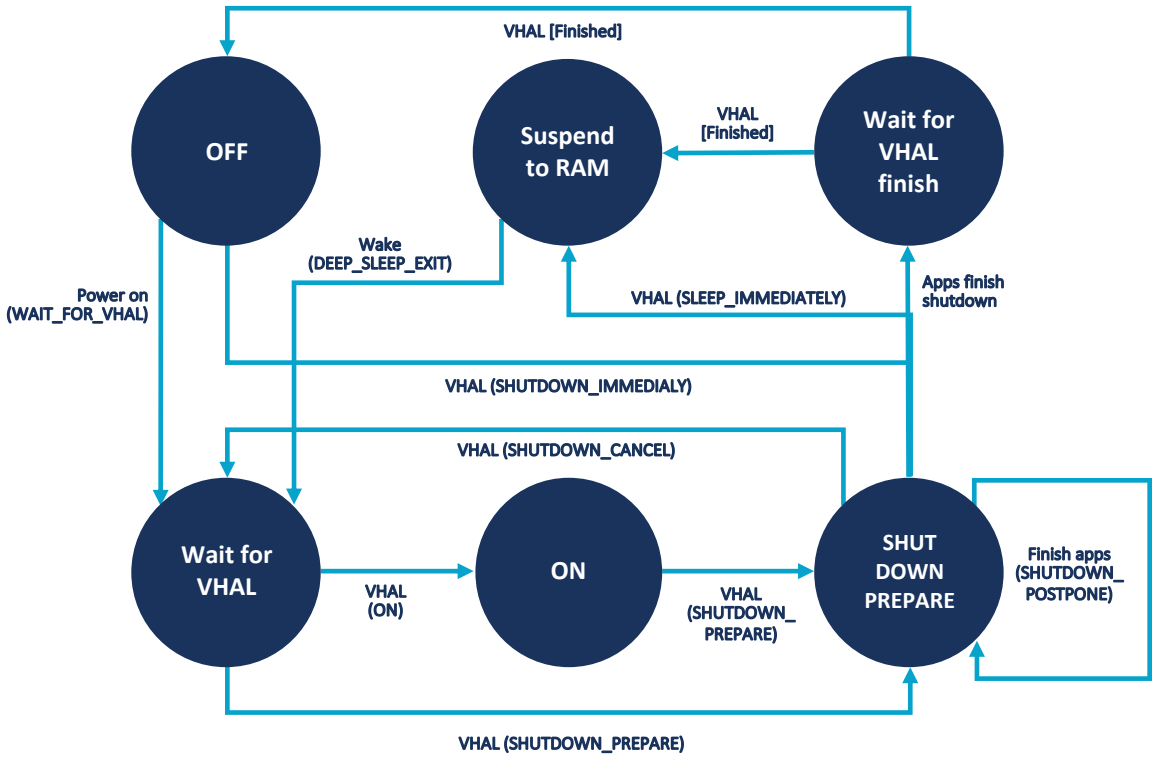

Figure 2: Googles car power state machine. Based on [3].

The applications can, therefore, perform an individual action in the event of a specific state or a state-change. This is necessary, for example, when switching on/off services such as Bluetooth or Wi-Fi.

CarProjectionManager: The efficient integration and handling of different projection technologies is a key requirement for today's infotainment systems. The user should be free to choose between Android Auto, Apple CarPlay or other mirroring technologies. With CarProjectionManager, Google enables the development of an application that guarantees the same system behavior when establishing a connection, managing smartphones and closing the connection.

IV.ANDROID AUTOMOTIVE EMBEDDED OS ARCHITECTURE

The Android platform (AOSP) can be generically divided into the components displayed in figure 3. Those are,

- Application framework and applications
- Android Automotive system service and Binder IPC
- Hardware Abstraction Layer
- Linux Kernel

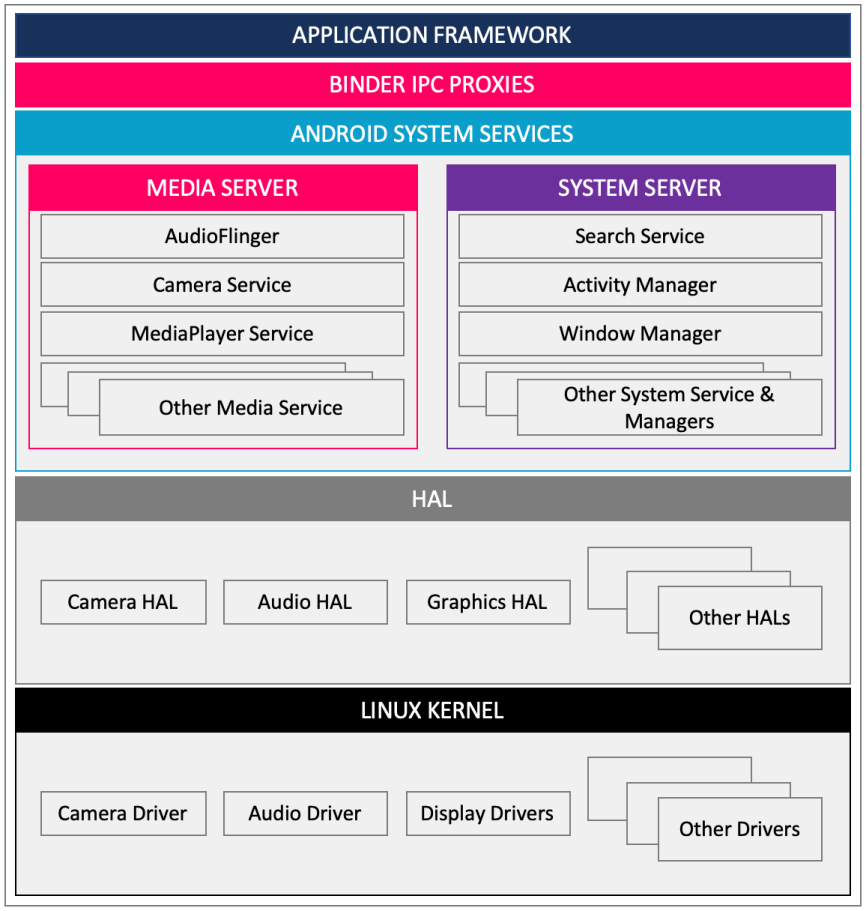

Figure 3: Android system architecture [4].

IV. ANDROID AUTOMOTIVE EMBEDDED OS ARCHITECTURE

Google extended its AOSP system with

- Car system applications
- Car APIs
- Car Services
- Vehicle Hardware Abstraction Layer

to provide a fully functional vehicle-agnostic in-vehicle infotainment operating system (refer to figure 1). The source code distribution of the IVI generally consists of

OEM and 3rd party applications as a set of Android applications including the HMI and application background services in the /product partition.

Android Open Source Project (AOSP): Include all the GIT-tree packages from the generic system applications, the application framework, system services through the HAL interfaces and should be in the /system partition.

Board Support Package (BSP): Includes the Linux kernel image with the HAL implementation for given hardware. The BSP is System on the Chip (SoC) dependent and part of the /vendor partition.

The OEM can extend the existing source code with self-developed automotive or non-automotive applications and system services, e.g. head-up display (HUD) management, tire pressure monitoring, charge program management, and others to extend the functionality of its infotainment system.

Due to the architecture change carried out in Project Treble and the expansion of the available partitions, not only the HMI layer but also the Android framework or the BSP and the hardware can be replaced in the future (see Figure 4).

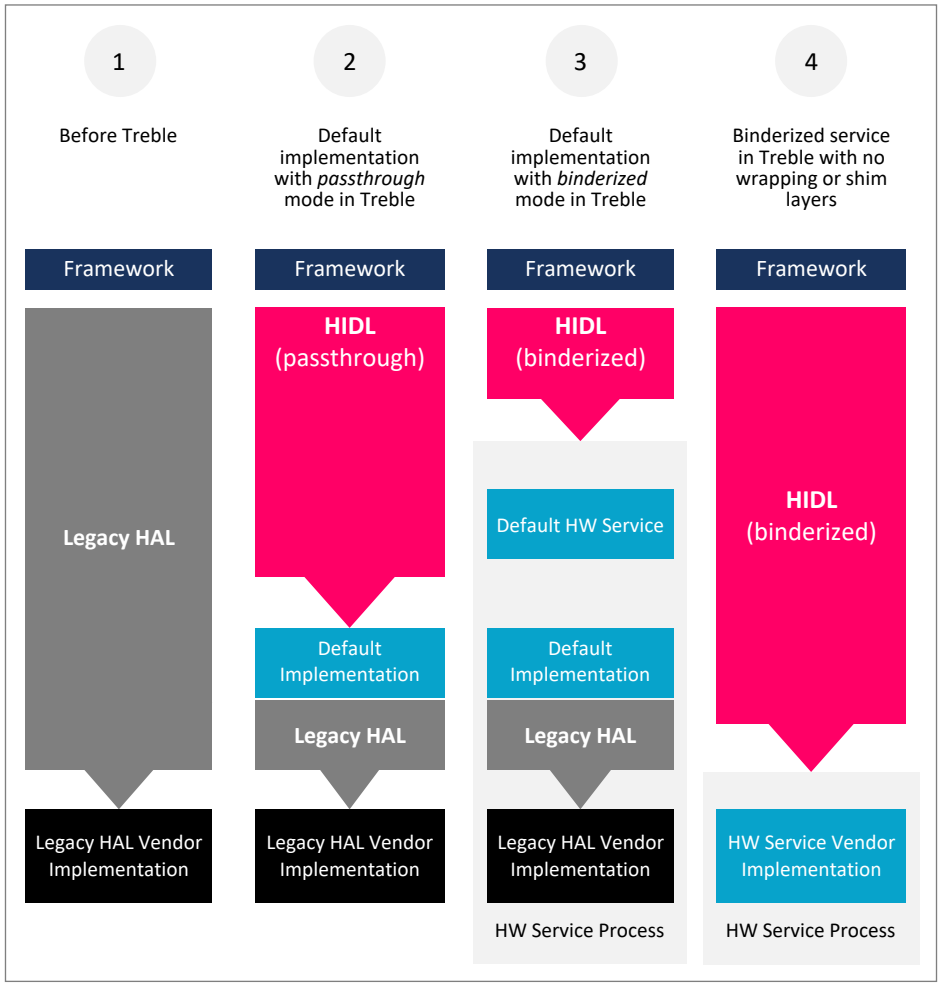

Figure 4: Platform-based operating system architecture [5].

The following section provides an overview of the responsibilities and tasks of respective system layers:

Application Framework: Commonly called the "HMI Layer", the Application Framework contains the system and user applications. Our recommendation is to design the applications in such way that they are only responsible for the visualization incl. small calculations to not block the MainUI thread and more the core business logic to the System Services in the Service Layer. Furthermore, applications manage their own translation labels and notifications using background services. This design allows for an easy update in the future and multiple HMI designs, e.g. for different car brands.

Service Layer. System services are included in the Service Layer and started by the SystemServer. They run as a System process which gives them additional privileges which normal Android Services do not have. This approach provides an opportunity for OEMs to develop other applications, that can use the service without source code duplication. Furthermore, OEMs can use the services as an additional layer for security reasons to avoid direct communication between the applications and the Hardware Abstraction Layer.
Vehicle HAL. The role of the Vehicle HAL is to expose car-specific interfaces to the system services, in an extendable, vehicle-agnostic manner. These interfaces include

- Access to signals to / from the ECUs in the vehicle
- Access to signals generated from the vehicle microcontroller unit to the IVI OS
- Access to service-oriented functions available on the vehicle network (e.g.: SOME-IP)

The described layers are the core elements of the platform and responsible for the data exchange between the applications and the vehicle ECUs. A detailed architecture is displayed in figure 5.

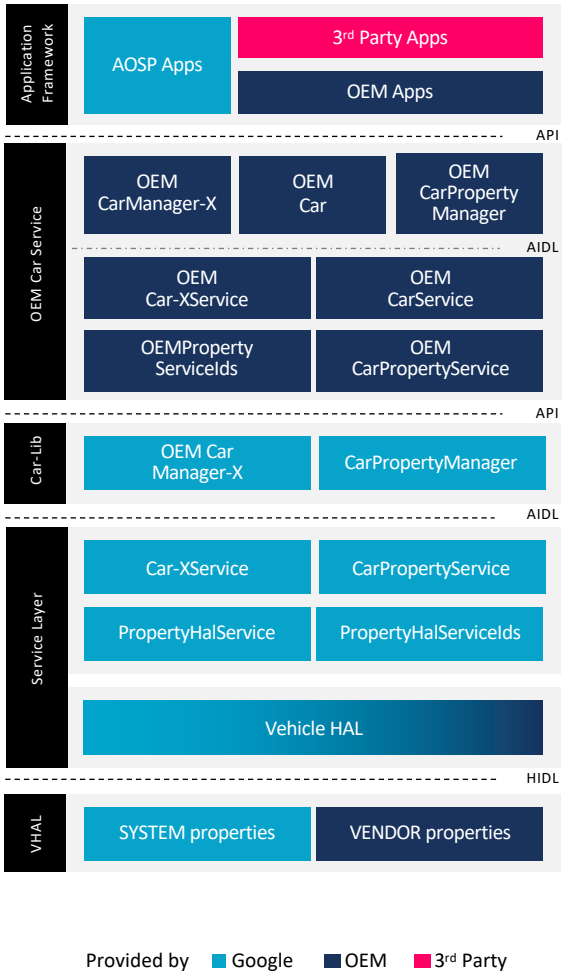

Figure 5: Detailed software component architecture view with extensions.
The processes will run top-down and bottom-up between the different components and layers.

CONCLUSION

In this technical white paper, we have provided some insights into the Android Automotive Embedded operating system, which is continuously being developed by Google and is available publicly on android.google-source.com. In addition to the basic features, frameworks and libraries we have explained the layered architecture and described how the system can be expanded by the OEM.

We consider Android Automotive an effective platform that includes all necessary core features. It requires lower development-, integration- and maintenance-cost for connected infotainment systems. The system can be fully customized, however any deviation from the original source code increases the OEM's development and maintenance effort. Another benefit is that Google will release regular patches and annual major upgrades with added features, extended functionalities and other improvements.

In our experience, the IVI development time can be cut short by 2 years compared to the usual 4-year development cycle. In this case Android Automotive Embedded OS was deployed incl. GAS and a fully customized HMI was developed. The implementation of a non-GAS system will require additional to for development and integration.

For more technical information on Android Automotive Embedded OS, we recommend our Android Automotive Training to dive deeper into the technical details.

CONTACT

PETER GESSLER
Android Automotive Architect
Peter.Gessler@p3-group.com

TINO MÜLLER
Mobility Solutions
Tino.Mueller@p3-group.com

MARIUS MAILAT
CTO
Marius.Mailat@p3-group.com

V. REFERENCES

1. https://source.android.com/devices/automotive
2. https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/packages/SystemUI/
3. https://source.android.com/devices/automotive/power/power
4. https://source.android.com/devices/architecture
5. https://events19.linuxfoundation.org/wp-content/uploads/2017/11/Project-Treble.-What-Makes-Android-8-Different_-Fedor-Tcymbal-Mera-Software-Services.pdf