AGL Software Defined Connected Car Architecture

Summary

This summary is based on the revision of the document “The Automotive Grade Linux Software Defined Architecture” version 1.2 (19.04.2018) Some sections are copied to this document for reference and comment.

The purpose of this analysis is to give the GENIVI Hypervisor Project input for further discussion, to promote reuse of already performed work, and to understand what changes this group would propose in case of disagreement (or simply improvement opportunity). The work may result in reporting improvements to the AGL paper (upstream principle), and in reuse of its content, either verbatim or modified, in our publications and project results.

Summary of paper, as of May 2018

Section 1 consists of a short introduction that motivates the the AGL project as well as defining its structure and goals. This section also provides the definition of the term virtualization. Important to note here is that the definition, when referring to embedded systems, does not define hypervisors as the only mean by which virtualization can be done, but includes the possibility to use system partitioner and containers as means to achieve separation between virtual environments. The different technical terms are defined more precisely in Section 6 of the document.

Section 2 outlines some challenges and benefits that virtualization can bring in the context of automotive.

Section 3 presents some use cases and requirements extracted from the latter. Generally, it is stated that virtualization provides means to create execution environments, which then execute different automotive functions. To achieve this, certain requirements are imposed on the ECU which would enable virtualization itself:

> **Hardware virtualization** - support for CPU, Memory and interrupts
>
> **Multicore processor** - that would allow to allocate one or multiple cores to each execution environment
>
> **Trusted Computing Module** to isolate safety-security critical applications and assets
>
> Optionally: **IO virtualization** to support GPU and connectivity sharing

The automotive functions could be of the following: Instrument cluster, IVI System, Telematics, Safety Critical Functions.

From these automotive functions, the paper then extracts requirements regarding the Computing, Peripherals sharing, Security, and Performance. Please refer to Table 1 as seen in [1].

	Automotive functions requirements for virtualized ECUs	
	Computing	• Static resource partitioning and flexible on-demand resource allocation (CPU, RAM, GPU and IO). • Memory/IO bus bandwidth allocation and rebalancing.
	Peripherals sharing	• GPU and displays shall be shared between execution environments supporting both fixed (each one talks to its own display or to a specified area on a single display) and flexible configurations (shape, z-order, position and assignment of surfaces from different execution environments may change at run time). • Inputs shall be routed to one or multiple execution environments depending on current mode, display configuration (for touch screens), active application (for jog dials & buttons), etc. • Audio shall be shared between execution environments. Sound complex mixing policies for multiple audio streams and routing of dynamic source/sink devices (BT profiles, USB speakers/ microphones, etc.) shall be supported. • Network shall be shared between execution environments. Virtual networks with different security characteristics shall be supported (e.g., traffic filtering and security mechanisms). • Storage shall support static or shared allocation, together with routing of dynamic storage devices (USB mass storage).
	Security	• Root of Trust and Secure boot shall be supported for all execution environments. • Trusted Computing (discrete TPM, Arm TrustZone or similar) shall be available and configurable for all execution environments. • Hardware isolation shall be supported (cache, interrupts, IOMMUs, firewalls, etc.).
	Performance and Power consumption	• Virtualization performance overhead shall be minimal: 1-2% on CPU/memory benchmarks, up to 5% on GPU benchmarks. • Predictability shall be guaranteed. Minimal performance requirements shall be met in any condition (unexpected events, system overload, etc.). • Execution environments fast boot: Less than 2 seconds for safety critical applications, less than 5 seconds for Instrument Cluster, and 10 seconds for IVI. Hibernate and Suspend to RAM shall be supported. • Execution environments startup order shall be predictable. • Advanced power management shall be implemented with flexible policies for each execution environment.

	Safety	• System monitoring shall be supported to attest and verify that the system is correctly running. • Restart shall be possible for each execution environment in case of failure. • Redundancy shall be supported for the highest level of fault tolerance with fallback solutions available to react in case of failure. • Real time support shall be guaranteed together with predictive reaction time.

Table 1: Automotive functions requirements for virtualized ECUs

Section 4 details the AGL approach towards virtualization and has a discussion on how a business model can be tailored to achieve some of the requirements mentioned in the previous section. The three key pillars of the AGL approach as defined in [1] are:

Modularity: hypervisors, virtual machines, AGL Profiles and automotive functions are seen by the AGL architecture as interchangeable modules that can be plugged in at compilation time (and where possible at runtime). The combination of the modules makes the difference. We can instantiate different modules. Modules can communicate with each other. To achieve modularity, interoperability will be required, especially between open and proprietary components.

Openness: There is no restriction in the way the AGL virtualization platform can be used, deployed and extended. The AGL virtualization architecture supports multiple hypervisors, CPU architectures, software licenses and can be executed as a host and guest.

Mixed Criticality: Applications with different level of criticality are targeted to coexist and run in a virtualized manner. As a consequence, AGL virtualization approach targets to consolidate applications different certification requirements.

Section 5 defines the AGL Virtualization architecture. Key definitions here are:

Execution Environments - these are silos that run concurrently on the system and enable the execution of different applications. Different types of EE are supported: certified, non certified, trusted, non trusted, open source, proprietary. Some of them are classified as Critical Execution Environments (CEEs), because their applications have an impact of the safety and security of the system. The others are considered Non Critical Execution Environments (NCEEs). They could be implemented as binary applications, combination of a set of libraries with the related application, or full featured operating systems, etc.

Communication buses - Consolidated EEs (and their applications) need to interact and communicate with each other. Two types of communication bus are supported by the architecture:

Critical communication bus: it is restricted to Critical Execution Environments (CEEs) only. The communication here has to address important requirements of safety and security.

Non critical communication bus: it is open to all the EEs available in the system. It has to address high performance and security requirements.

Virtualization Platform - This module leverages on system software and hardware functions to create the silos for the execution of different EEs. Depending on the type and the combination of the functions

used to build this module, the virtualization platform can be implemented using technologies like hypervisors, system partitioners, containers, etc. EEs can communicate with the Virtualization Platform (EE configuration, power management, etc) through specific channels.

In these definitions, it is again stressed that the Virtualization itself can be achieved by means other than hypervisors.

Section 6 Provides an extensive definition the different types of mechanisms for virtualization as well as an overview of open source and commercial solutions that offer the functionality.

Section 7 is a conclusion of the paper.

References:

[1] The AGL software defined connected car architecture:
https://docs.google.com/document/d/1HpYzClh0nDEocsUHb17X0DxiehsAbCgyWE-P2Wk_RNU/edit