GENIVI®

TECHNICAL SUMMIT '19

Hypervisor Project

Gunnar Andersson, GENIVI Development Lead

GENIVI Tech Summit, Troy, MI | 12 November 2019

GENIVI®

Hypervisor Project

- A collaborative group addressing challenges in Virtualization
- A key activity since the start of GENIVI's Domain Interaction and Multi-OS integration strategy
- Bootstrapped with a number of workshops, All-Member Meetings, Tech-Summits
- Weekly call every Monday at 10am CET
  - US-friendly call time is possible (let us know your interest)
- Documented on GENIVI Public Wiki:
  - Minutes
  - Agenda announcement

Project Introduction

GENIVI®

Project Goal

Make Virtualization easier to deploy in automotive environments

Project Goals

Make Virtualization easier to deploy in automotive environments

- Standardization
  - Hypervisor APIs
  - Virtual device interfaces
  - Architectures
  - Guest integration
- Outreach and education
  - Designing with virtualization – Whitepaper
  - Why is a hypervisor necessary?

Areas of Standardization

- Key areas for standardizing interfaces
- Collaboration includes several open source and commercial hypervisors
- Interoperability
- Vendor Neutral Interfaces

Hypervisor API	Guest Boot Protocol
Power Management	Device Virtualization
Architectures	Hypervisor Requirements

Participants Since Project Start

- ADIT
- Alpine
- ARM
- Elektrobit
- Kernkonzept
- EPAM, and other Xen-Project representatives
- Green Hills Software
- OpenSynergy
- Perseus
- Sysgo / PikeOS
- Virtual Open Systems
- XVisor founder and developer

Effects of Standardization

GENIVI®

- Hypervisor API
- Guest Boot Protocol
- Power Management
- Device Virtualization
- Architectures
- Hypervisor Requirements

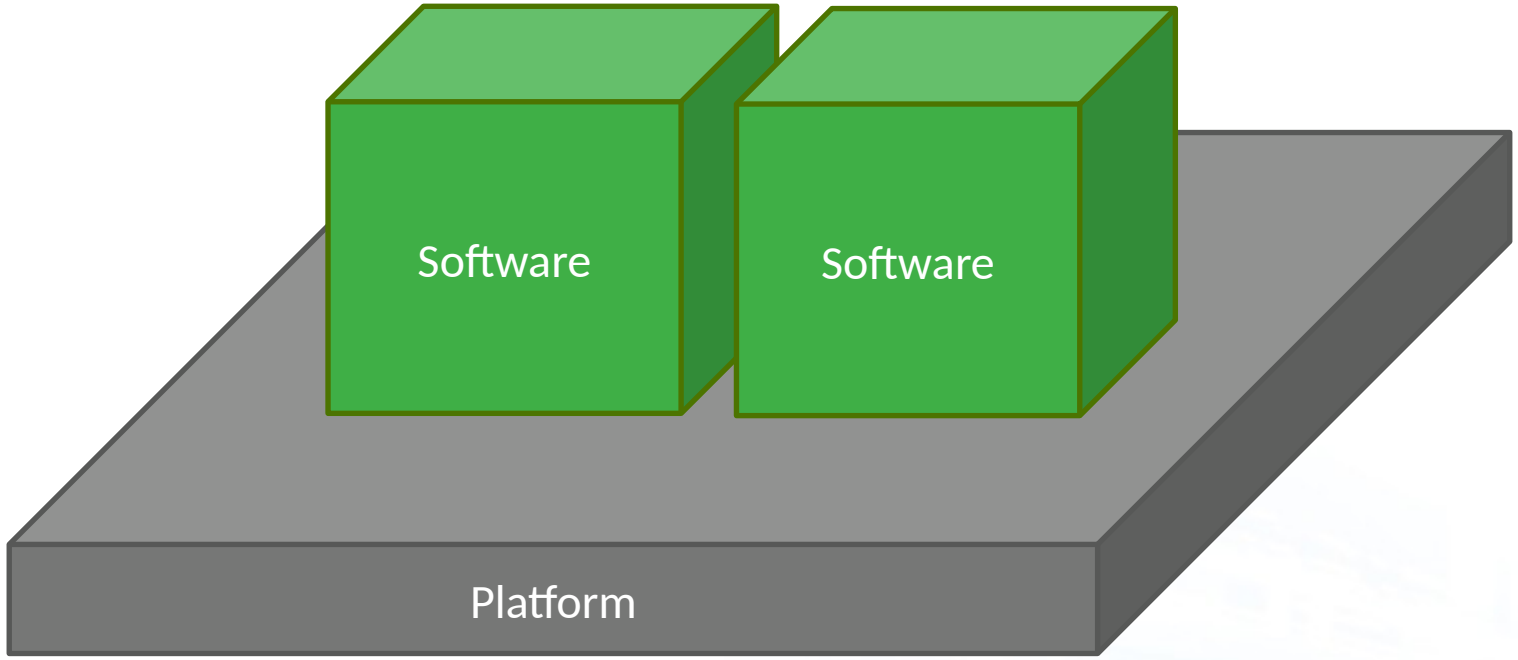
- Virtual machines can be ported across hypervisors
- Common code (drivers, etc)
- Validation tests
- Easier multi sourcing
- Standards can be referenced in OEM specifications
- Shortened development cycles
- Software re-use

Standardized Automotive Virtual Platform

GENIVI®

What is a “Platform”

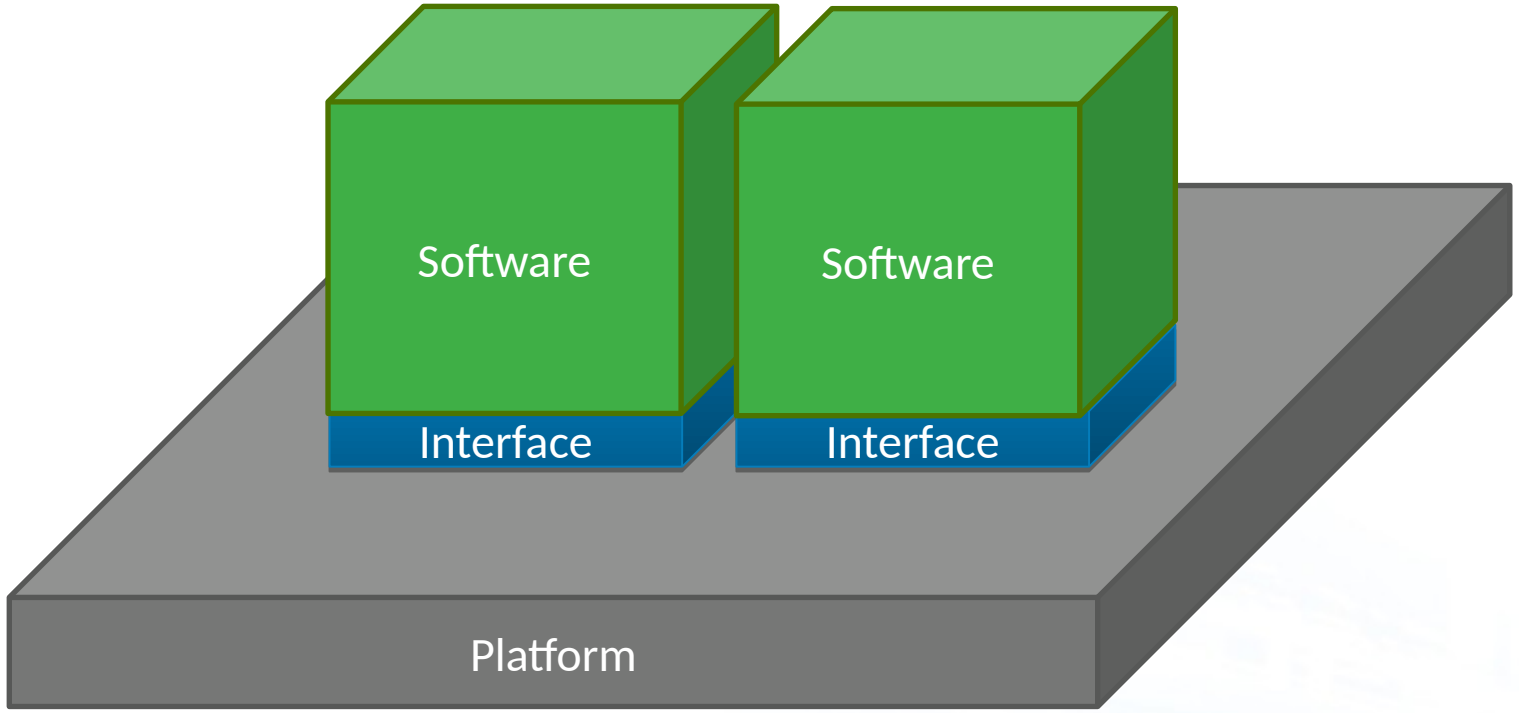
A computing platform is an **environment** where **software** can be run



What is a “Platform”

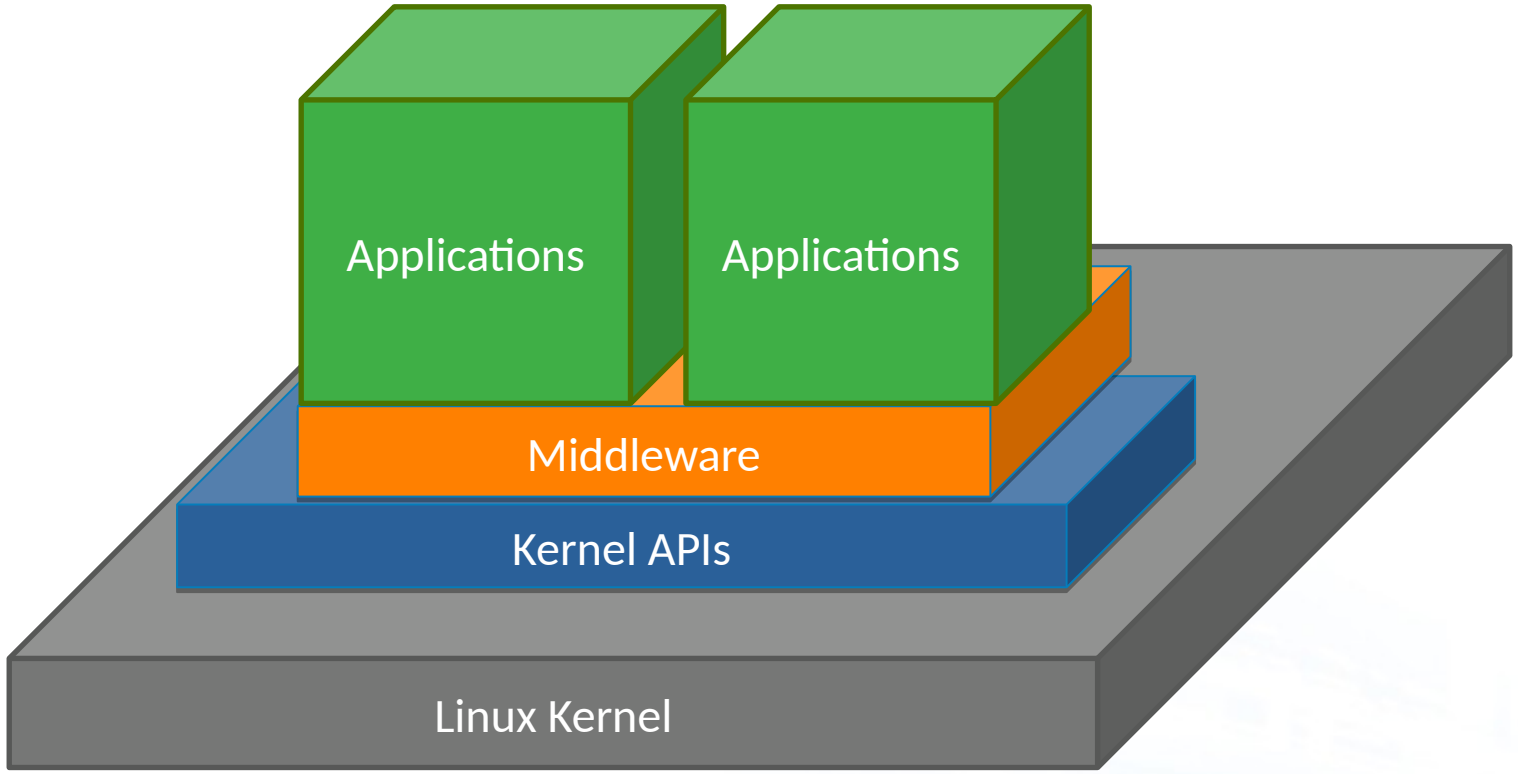
A computing platform is an **environment** where **software** can be run

...with a well known interface



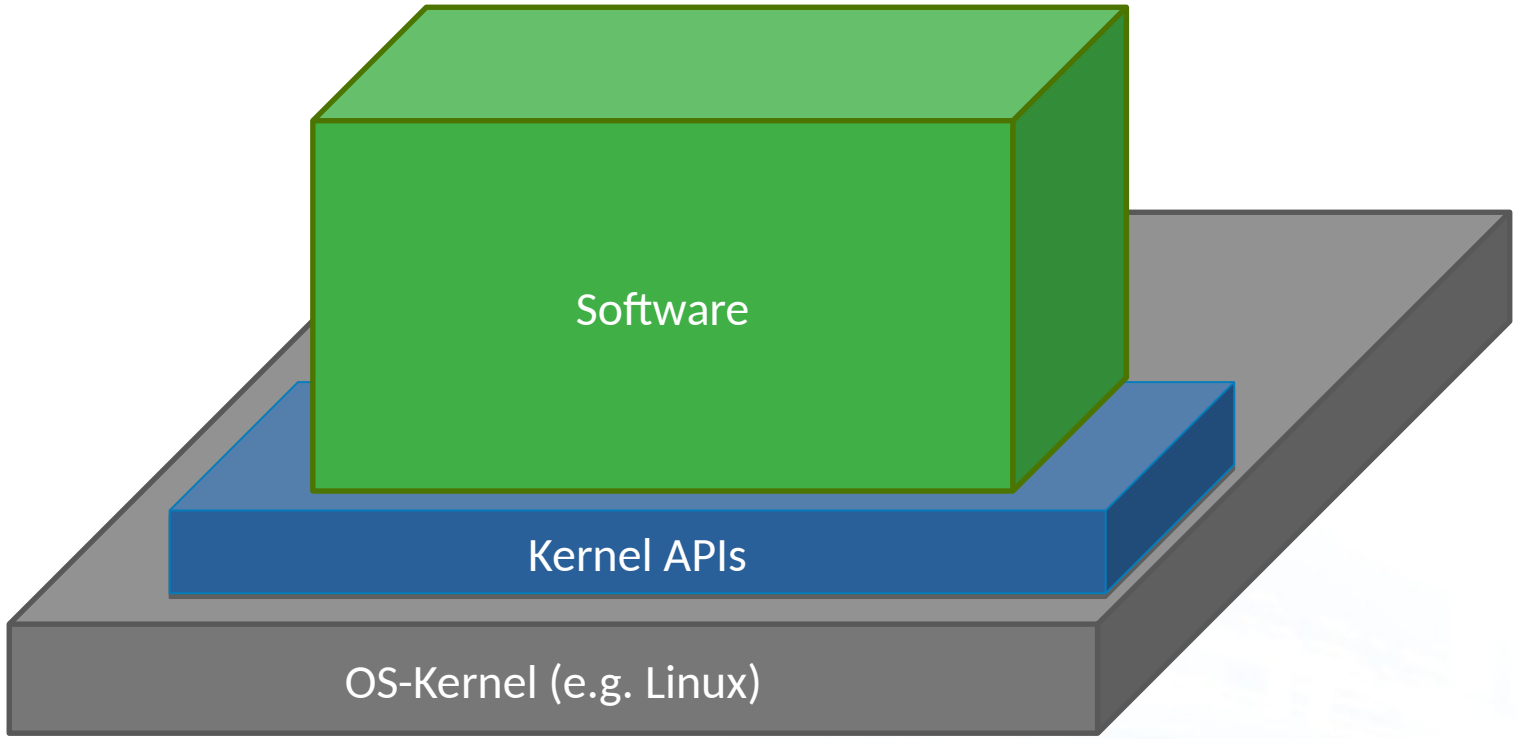
Standard APIs

GENIVI has a long history of driving standard APIs



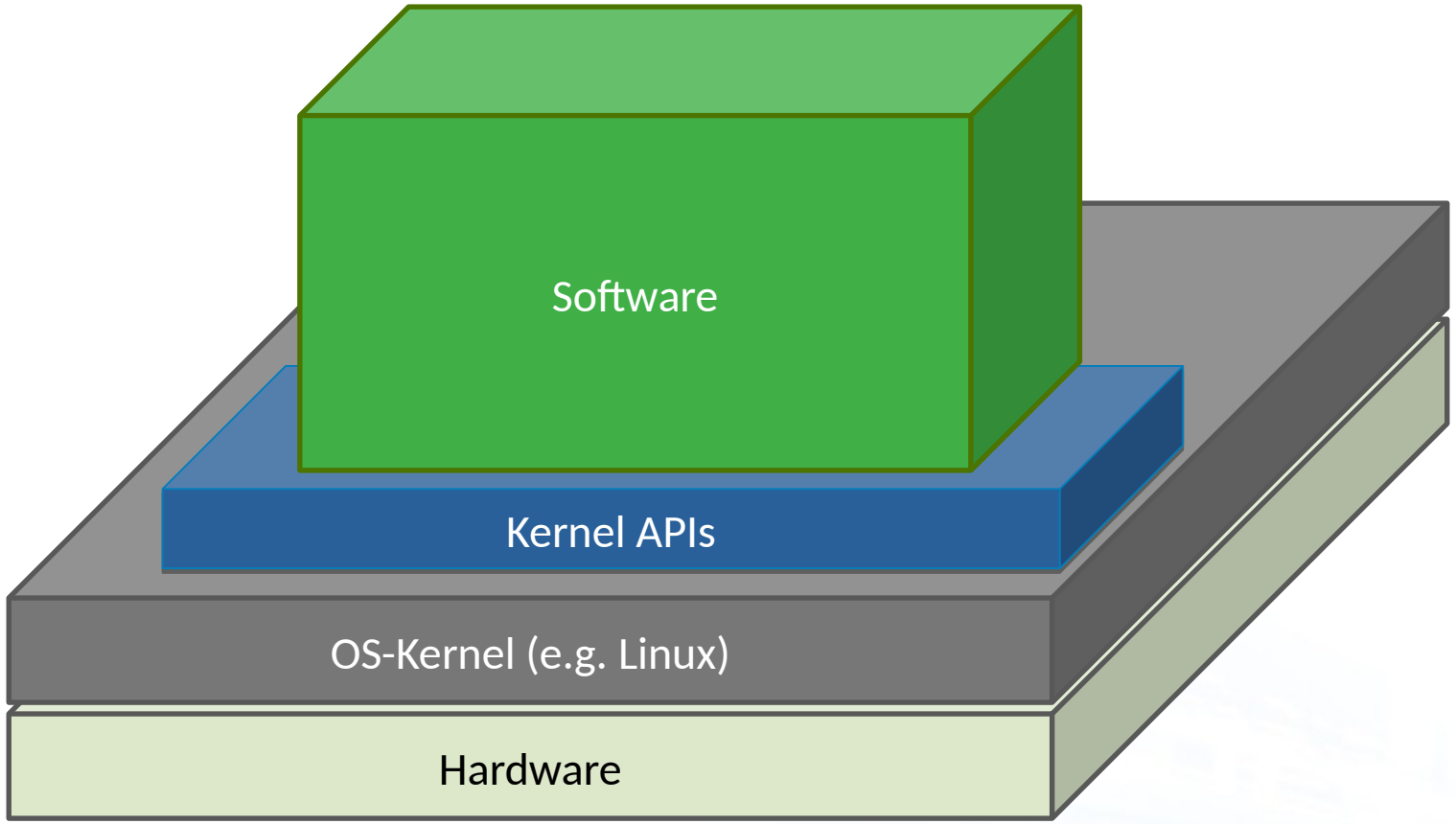
Single kernel systems

Application code depends on a stable well-defined kernel API



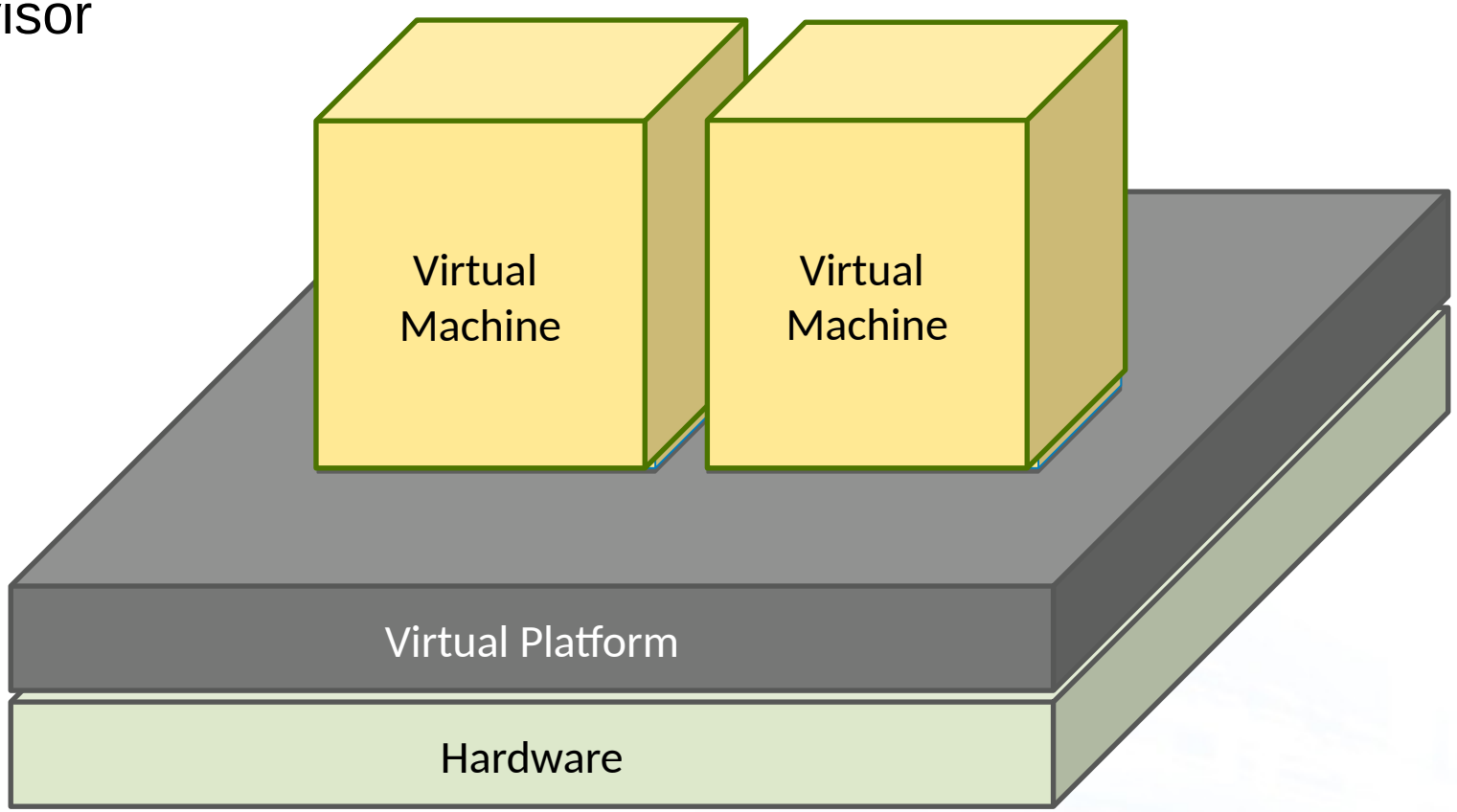
Single kernel systems

Application code depends on a stable well-defined kernel API
OS Kernel controls hardware



What is the “Virtual Platform”

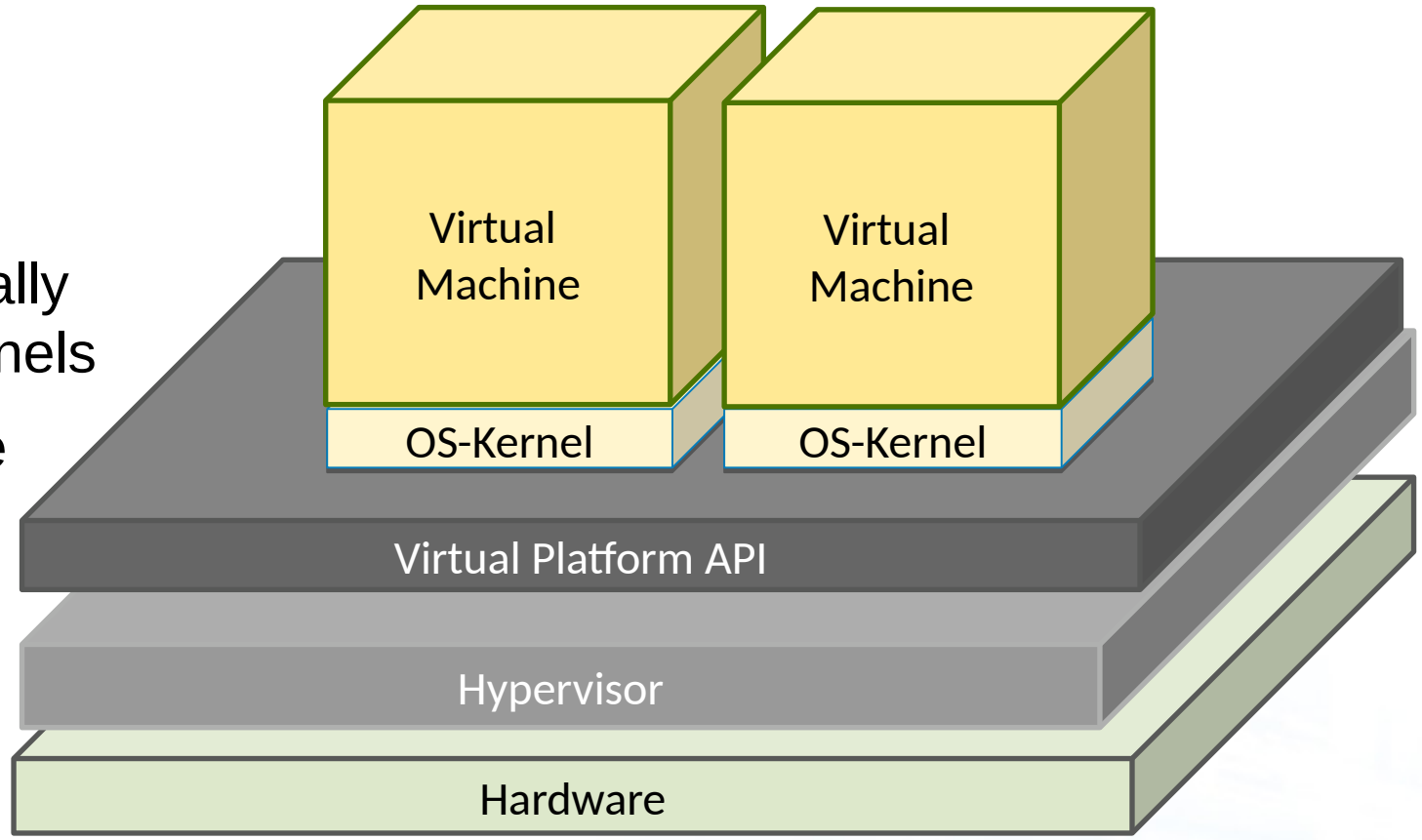
The virtual platform is a **hypervisor environment** where **Virtual Machines** can be run on top of a hypervisor



What is the “Virtual Platform”

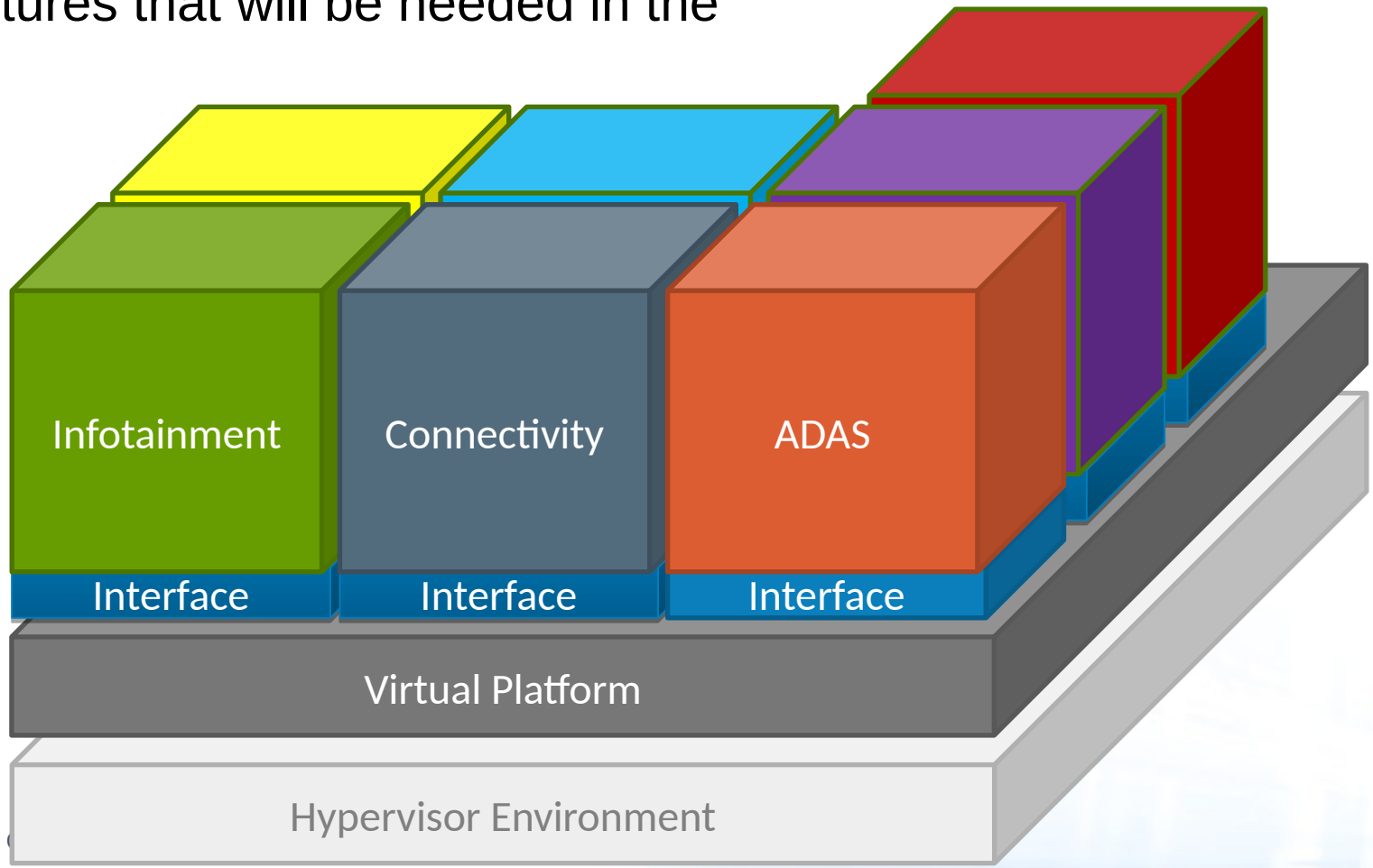
The virtual platform is a **hypervisor environment** where **Virtual Machines** can be run on top of a hypervisor

- Multiple VMs
- Running potentially different OS Kernels
- Virtual Hardware



Automotive Virtual Platform

A shared automotive virtual platform will allow building the **software defined** vehicle system architectures that will be needed in the next decade.



Challenges in implementation

GENIVI®

Challenges in implementing virtualization

- Porting existing full systems from “real hardware” to virtual machine
  - Kernel to (virtual) hardware boundary is now different
  - This boundary is different on different HVs
  - → Varying feature support
  - → Integration problems, uncertainties
  - Behavioral changes?
    - Performance?
    - Scheduling of processes (e.g. Real-time or otherwise critical)
  - Isolation guarantees, popularly referred to as handling *mixed criticality*
    - Safety critical functions and Cybersecurity concerns
  - Vendor Lock-in

Addressing the Challenges with GENIVI

- HV differences in interface → *interface/platform standards!*
- HV differences in supported features → *shared requirements!*
- Integration issues → *reduced...*
- Vendor lock-in → *eliminated... ?*
- OS Kernel to (virtual) hardware boundary → *better defined, shared*
- Isolation guarantees → *analysis, design recommendations*

Remaining challenges

- Performance → ??
- The challenges of process/VM scheduling → ??
- Remaining behavioral differences

- Analysis, shared understanding, shared code can chip away at these

- **Solve what we can in collaboration**
  = time to address remaining challenges!

Automotive Virtual Platform Specification

- Primarily a Interface description document
- Plus some additional requirements
- Vendor/Hypervisor neutral
- Encourages many interoperable independent implementations
- Based on existing standards and references those
  - VIRTIO
  - ARM platform standards
    - Selected small specifications that might be implemented on other hardware
- Selects, clarifies, and augments requirements to match the Automotive needs

Areas of Standardization

- Standard hypercall interfaces
  - Architecture calling conventions
- Scheduler control
- Guest state management

Find commonality between hypervisors, avoid interface overlap

Hypervisor API	Guest Boot Protocol
Power Management	Device Virtualization
Architectures	Hypervisor Requirements

Areas of Standardization

- Boot protocols allow hypervisors to load and boot virtual machines
- Often project or hypervisor specific
- Many protocols exist
  - Classic “Master Boot Record”
  - Linux aarch64 boot protocol
  - Android aboot
  - Standard UEFI
  - ARM Embedded Base Boot Requirements (EBBR)

Allow booting any guest on any hypervisor

Hypervisor API	**Guest Boot Protocol**
Power Management	Device Virtualization
Architectures	Hypervisor Requirements

Areas of Standardization

- Power Management is critical for automotive
  - Thermal constraints
  - Standby endurance
- Many existing protocols
  - ACPI (Advanced Configuration and Power Interface)
  - ARM PSCI (Power State Control Interface)
  - ARM SCMI (System Controller Management Interface)

Find common and suitable interfaces for automotive, define minimum requirements

Hypervisor API	Guest Boot Protocol
Power Management	Device Virtualization
Architectures	Hypervisor Requirements

Areas of Standardization

- I/O Virtualization is key when hardware resources must be shared
- Approaches differ but cross-platform and cross-architecture standards exist
  - Device emulation
  - SoC vendor specific solutions
  - VIRTIO

Define vendor neutral interfaces, leverage from enterprise and cloud infrastructure

Hypervisor API	Guest Boot Protocol
Power Management	Device Virtualization
Architectures	Hypervisor Requirements

Areas of Standardization

- Philosophies differ between hypervisors but general concepts often the same
  - Find a common language when talking about virtualization
  - Avoid naming confusion
  - Inform hypervisor users about their options
- Evaluate software architectures for virtualization

Make sure we can talk about the same thing and understand each other

Hypervisor API	Guest Boot Protocol
Power Management	Device Virtualization
Architectures	Hypervisor Requirements

Areas of Standardization

- OEMs and Tier 1 have broad requirement spaces
  - OEM standards
  - Different system architectures need different requirements
  - Commonality is huge but emphasis differ

Find minimum requirements that allow an even playing field for all hypervisor vendors

Hypervisor API	Guest Boot Protocol
Power Management	Device Virtualization
Architectures	Hypervisor Requirements

Automotive Virtual Platform Specification

- "Virtual Platform definition for Automotive Hypervisor Environments"
  - Describes essential hypervisor environment interfaces
  - Allows interoperability between different hypervisors

Hypervisor API
- SMCC
- …

Guest Boot Protocol
- UEFI
- EBBR

Power Management
- PSCI
- SCMI
- …

Device Virtualization
- VIRTIO
- Storage, Network, GPU, USB, Sensors, Audio, Media Codec, Serial, …

Architectures
- Glossary
- Definitions
- …

Hypervisor Requirements
- Use cases
- Minimum requirements

I/O Virtualization with VIRTIO

- Reference OASIS VIRTIO standard
  - Extend where needed
  - Enhance where possible
  - Collaborate with upstream
- Leverage as much as possible from existing standard devices (storage, network, GPU, etc)
- Add automotive domain specific devices to upstream standard

Many hypervisors used in automotive already use the VIRTIO standard today.

The Automotive Virtual Platform Specification

- Currently: No one else is doing this for automotive. This is *the* initiative.
- Always open to collaboration, merging, adaption
- We recognize previous work in this area, build on top of it
- Prepare for others to contribute, extend, build on top of our results
- Spreading the message that there should only be one *automotive* VP standard
- Outreach to other consortia, to GENIVI's industry collaboration partners
- Actively reusing "upstream" work by reference, and aiming to minimize differences
- If you agree – participate, review, write, express support, <u>be public about that support</u>, and take part in outreach and dissemination.

Status

- Draft document about to be finished
- Covers mainly I/O virtualization topics
- Deferring some areas that are in flux
- Discussion section analyze the background
- Requirement section has formal requirements, some mandatory, some conditional
- Reuses/refers to “upstream” (VIRTIO), with clarifications for automotive
- See introduction and analysis page here
- See current draft document here

Effects of Standardization (repeat)

Hypervisor API	Guest Boot Protocol
Power Management	Device Virtualization
Architectures	Hypervisor Requirements

- Virtual machines can be ported across hypervisors
- Common code (drivers, etc)
- Validation tests
- Easier multi sourcing
- Standards can be referenced in OEM specifications
- Shortened development cycles
- Software re-use

Whitepaper

GENIVI®

Designing with Virtualization Whitepaper

- The whitepaper is under development
- Highlight and analyze the *need* for Hypervisor technology
  - *Depending* on system requirements, of course
- VMs are separated by spacial and temporal isolation
- Space deals with usage of memory and other limited resources
- Time deals with allotting usage time *exactly* to VMs, not only for CPU cores but other constrained resources
- Modern CPUs/SoCs have multiple levels of caches and memory buses, some of which are shared between CPU cores and shared between VMs. VMs make use of DMA channels and other execution time or bandwidth constrained features.
- Kernels *may* not fully handle isolation of critical / non-critical tasks
- Multiple kernels *cannot (*cooperatively) handle it
- Diverse operating system needs make a single kernel choice insufficient

Outlook

GENIVI®

Outlook

- About to release draft of automotive virtual platform specification
  - Good status of certain areas
  - Omitting a few sections that are in flux
  - Architecture section to be added

- Find more hypervisor *users* and vendors who want to participate in the group

- Work more with system architectures
  - Providing reference architectures
  - Blueprints on how to effectively use hypervisor technology

JOIN OUR EFFORTS!

Thank you!

Visit GENIVI:

http://www.genivi.org

http://projects.genivi.org

Contact us:

help@genivi.org

GENIVI®

BACKUP SLIDES

GENIVI®

Scheduling and isolation challenges on multi-core SoCs