FTF 2016
TECHNOLOGY FORUM

LINUX FOR AUTONOMOUS DRIVE

FTF-AUT-N1789

STEFAN MALINICI
NXP AUTOMOTIVE OPERATING SYSTEMS TEAM MGR
FTF-AUT-N1789
MAY 18, 2016

AGENDA

- Autonomous Drive Trends and Software Needs
- Why Linux in Autonomous Drive
- Challenges of the Open Source Software
- How Linux is Validated
- Safety Aspects
- Technical Solution
- Demo
- Legal Aspects
- Conclusion

AUTONOMOUS DRIVE TRENDS AND SOFTWARE NEEDS

Autonomous Cars

An **autonomous car** is a **vehicle** that is capable of **sensing** its environment and navigating without human input.

Source: Wikipedia

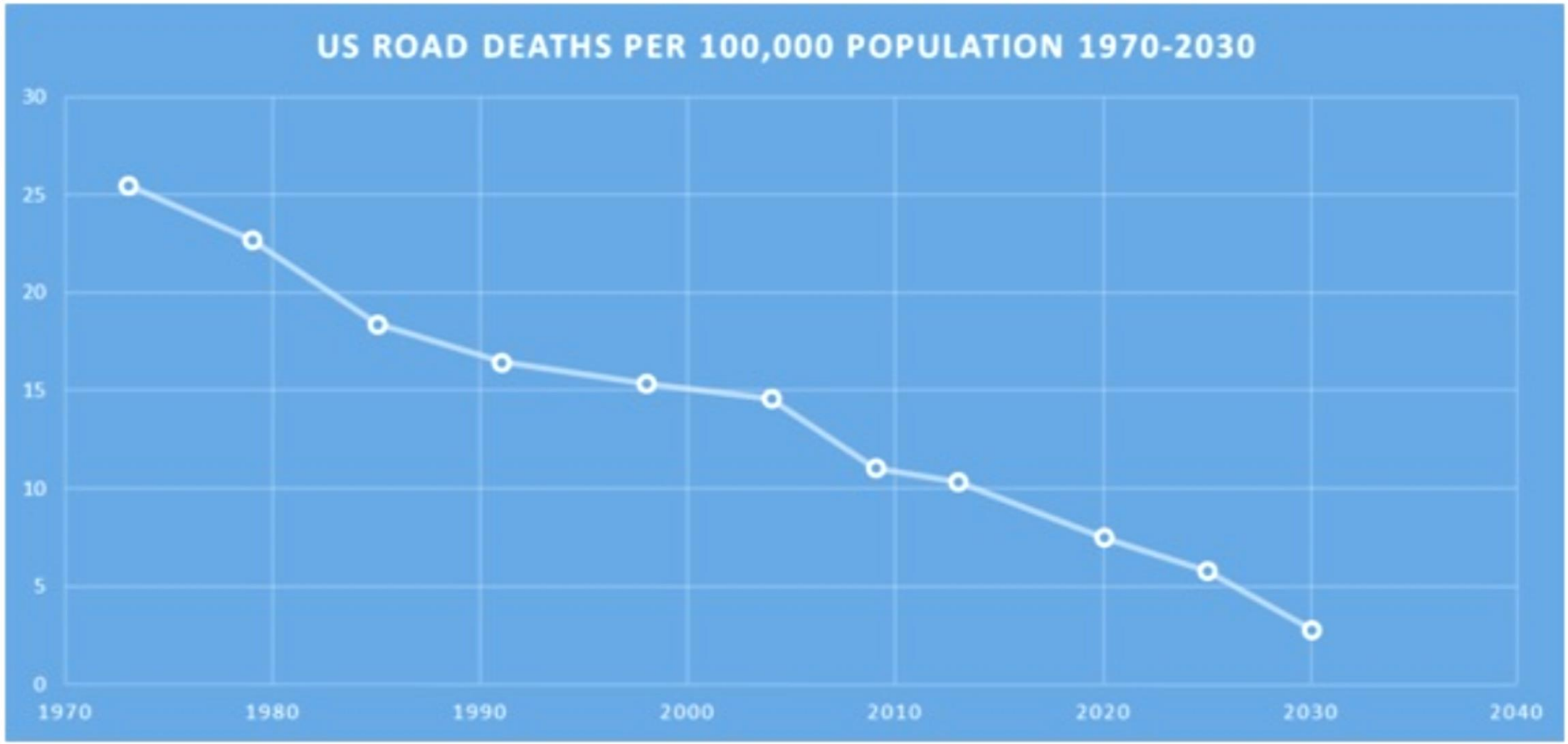
IEEE predicts up to 75% of vehicles will be autonomous in 2040

Ford CEO expects fully autonomous cars by 2020

Next generation Audi A8 capable of fully autonomous driving in 2017

Uber fleet to be driverless by 2030

Why Autonomous Drive



Levels of Autonomous Drive

LEVEL 1
Driver Assistance

- **Adaptive cruise control (ACC)**
- **Automatic braking**
- **Lane keeping**

LEVEL 2
Partial Automation

- **Partial automated parking**
- **Traffic jam assistance**
- **Emergency brake with steer**

LEVEL 3
Conditional Automation

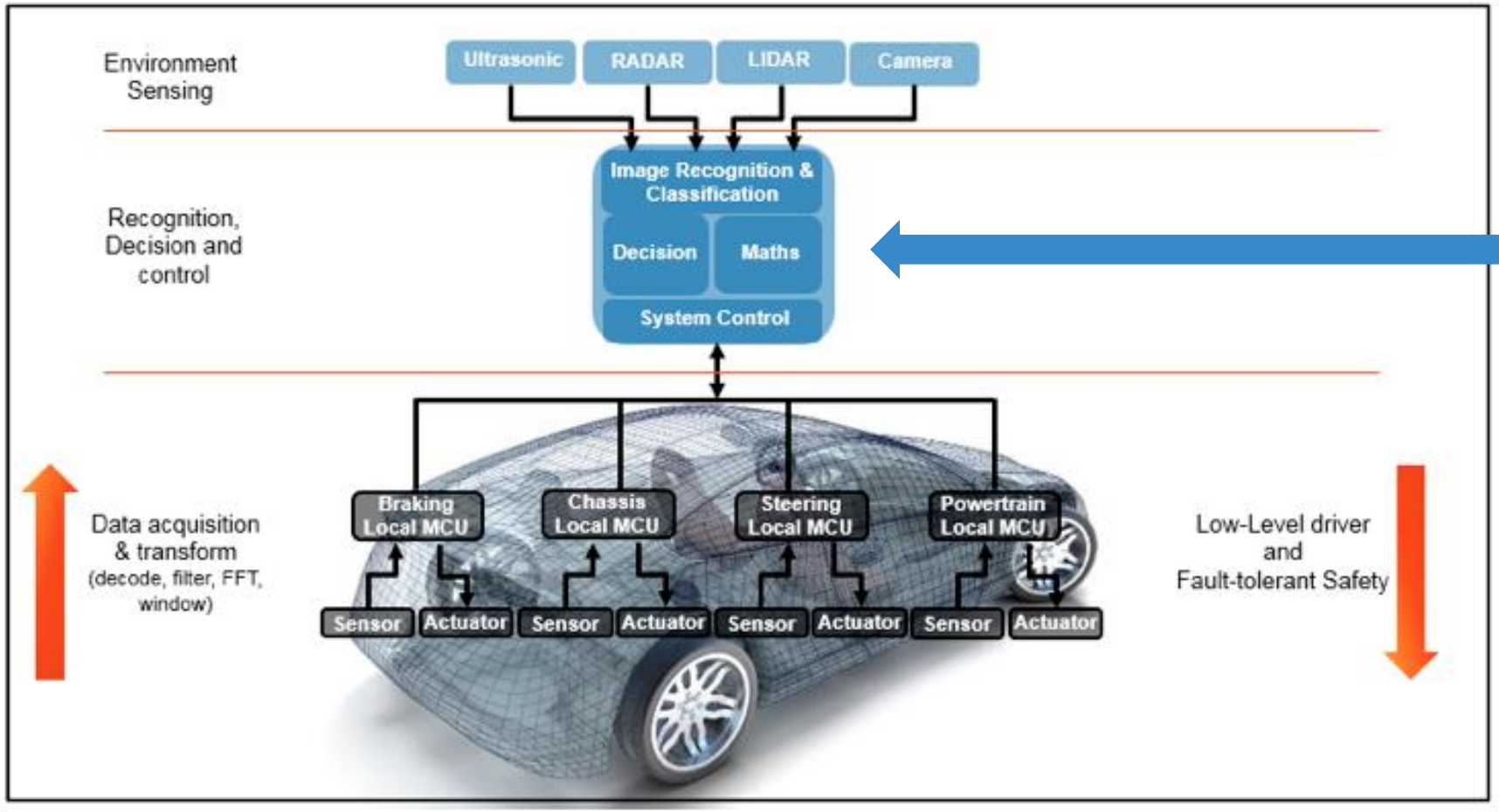
- **Semi autonomous:**
  - **Highway chauffeur**
  - **Self parking**

LEVEL 3
Fully Automated

- **Fully autonomous**

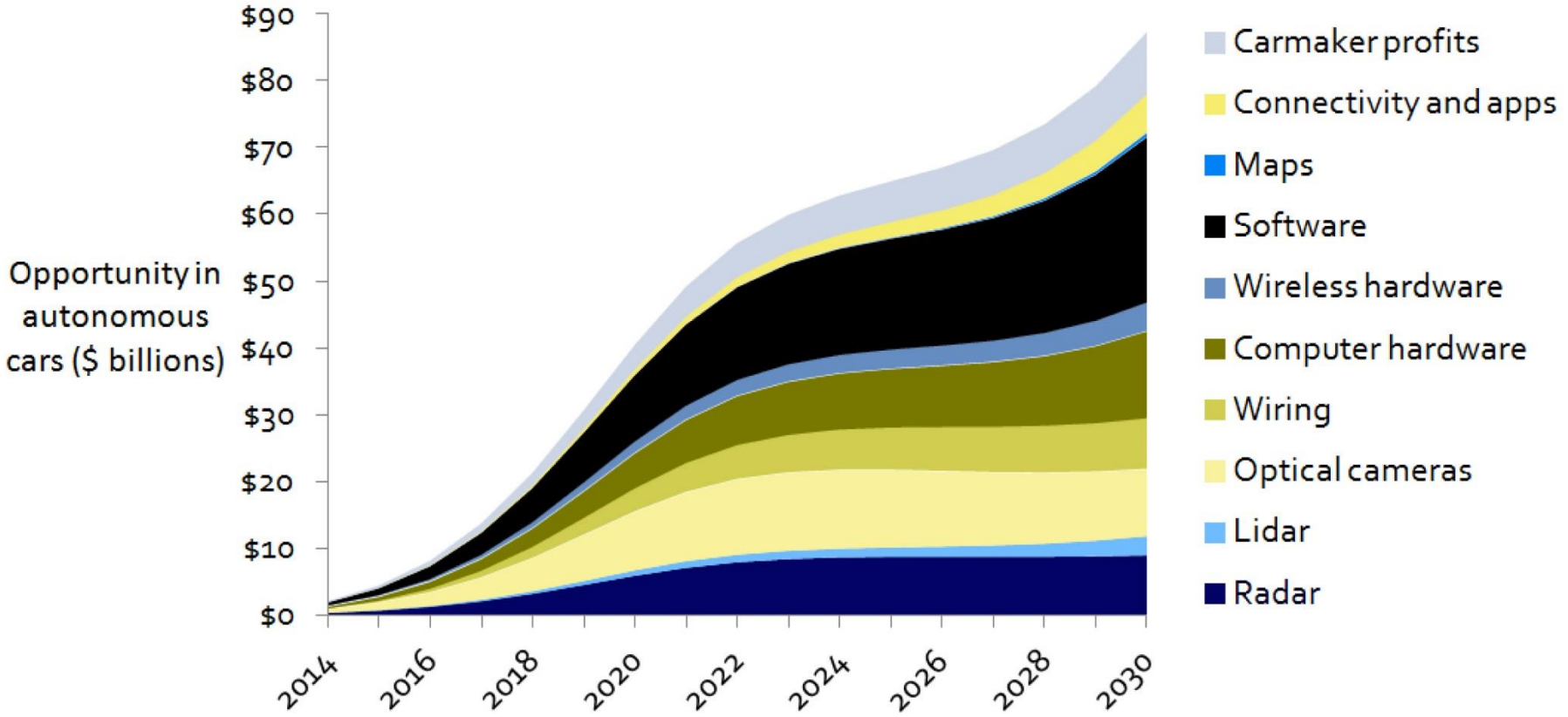
Present → 2020 → 2025 → 2030

Autonomous Drive Applications



Software importance is huge

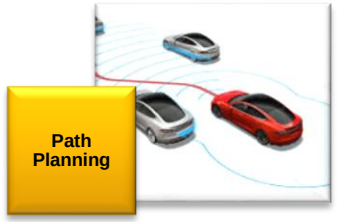
Software Drives Autonomous Drive

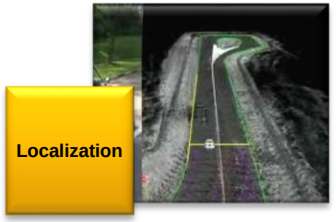


Source: Lux Research, Inc.
www.luxresearchinc.com

Software will be competitive differentiator. The software opportunity in autonomous cars will grow rapidly from $0.5 billion today to $10 billion in 2020 and $25 billion in 2030.

Software Components for Autonomous Drive





Apps

Vision | Codecs | Communication Stacks | Algorithms | Tools

Middleware

Camera | Sensors | Communication Drivers | Accelerators

Drivers

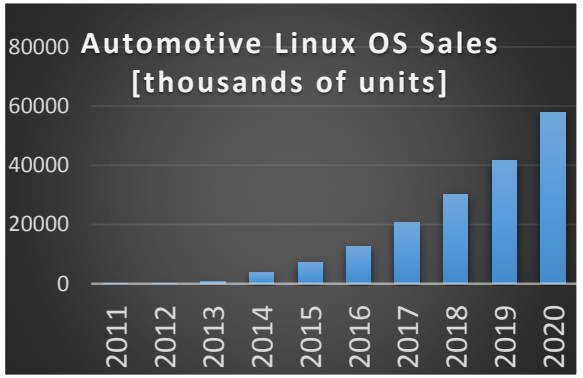
Memory Management | Scheduling | Real Time Scheduling | Filesystems | Hypervisor

OS

Safety and Security

WHY LINUX IN AUTONOMOUS DRIVE?

History of Open Source & Automotive



Huge increase of Linux presence in the car

AUTOMOTIVE GRADE LINUX

- Part of Open Source Foundation
- Targets Infotainment, telematics and Instrument Cluster
- All major OEMs and Tier1s are members and also NXP

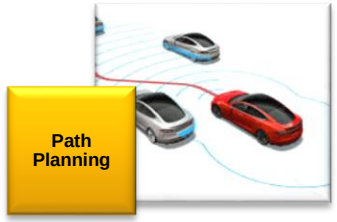
GENIVI®

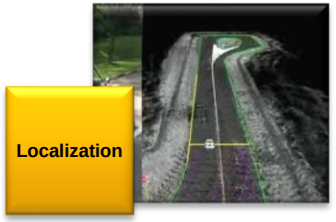
- V-model development approach
- They provide an infotainment and diagnostics framework (mostly middleware)
- NXP has platforms compliant with GENIVI

Advantages of using Open Source in Automotive

- The Open Source Community offers a huge variety of **free software**
- Unrestricted **freedom of use**
- Open means it software can be **easily reused**
- Linux is **constantly evolving**
- Software applications developed on desktop PCs can be easily ported on embedded platforms
- Flexibility and modularity – allows **separation from proprietary code**

What Open Source Provides





Apps

Vision | Codecs | Communication Stacks | Algorithms | Tools

Middleware

Camera | Sensors | Communication Drivers | Accelerators

Drivers

Memory Management | Scheduling | Real Time Scheduling | Filesystems | Hypervisor

OS

Safety and Security

Covered
Partially Covered
Not Covered

Linux Real Time Patch

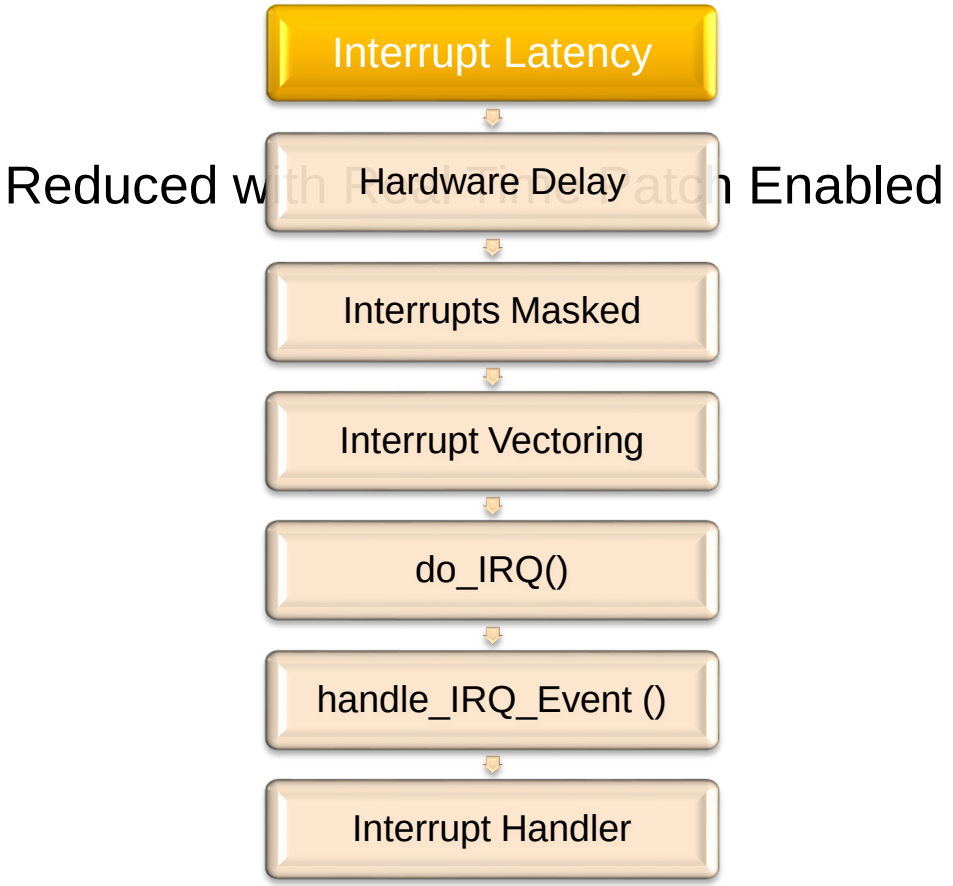
- This option further reduces the scheduling latency of the kernel by replacing almost every spinlock used by the kernel with pre-emptible mutexes and thus making all but the most critical kernel code involuntarily pre-emptible.

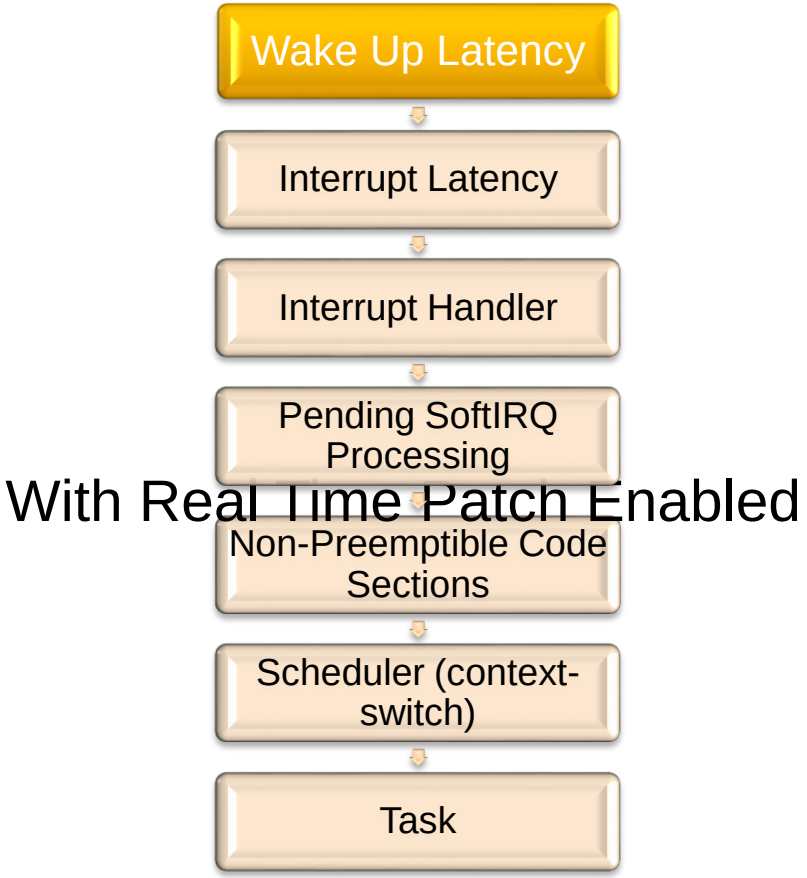
- The remaining handful of low level non-preemptible code paths are short and have a deterministic latency of a couple of tens of microseconds (depending on the hardware).

- This also allows applications to run more 'smoothly' even when the system is under load, at the cost of lower throughput and runtime overhead to kernel code.

Linux Real Time Patch

- Introduce preemption points on long kernel paths

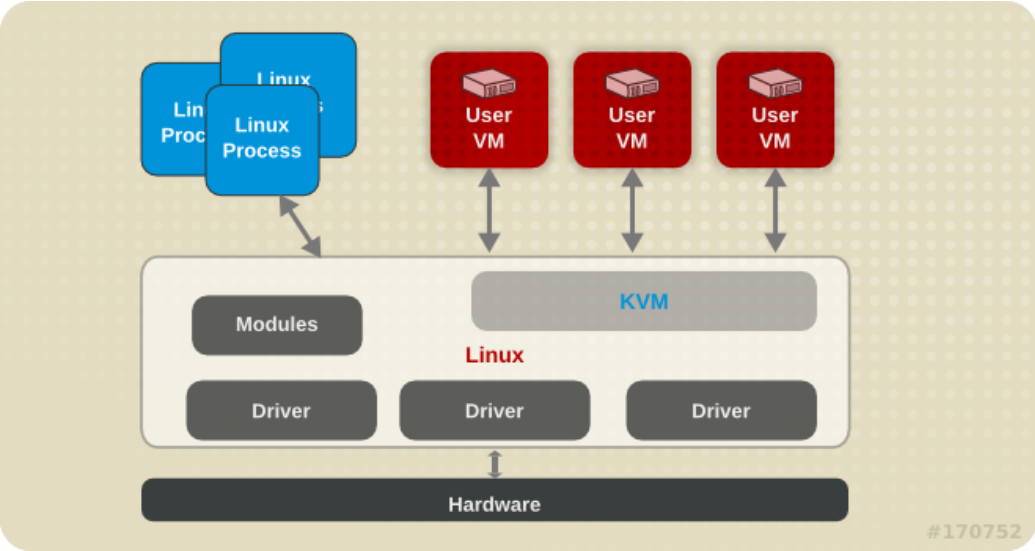


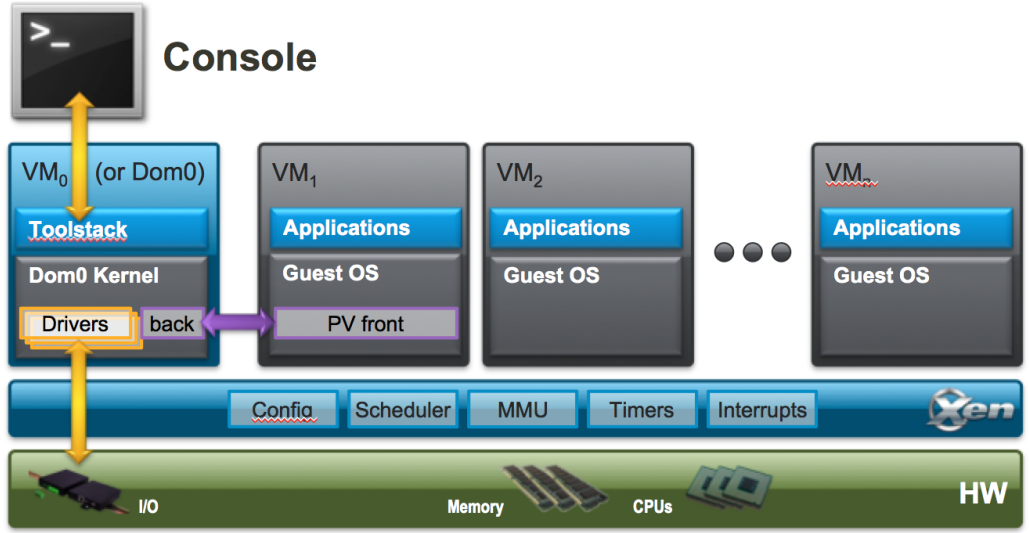


Source: Timesys

Hypervisors

Hypervisors help on separation between trusted and untrusted domains

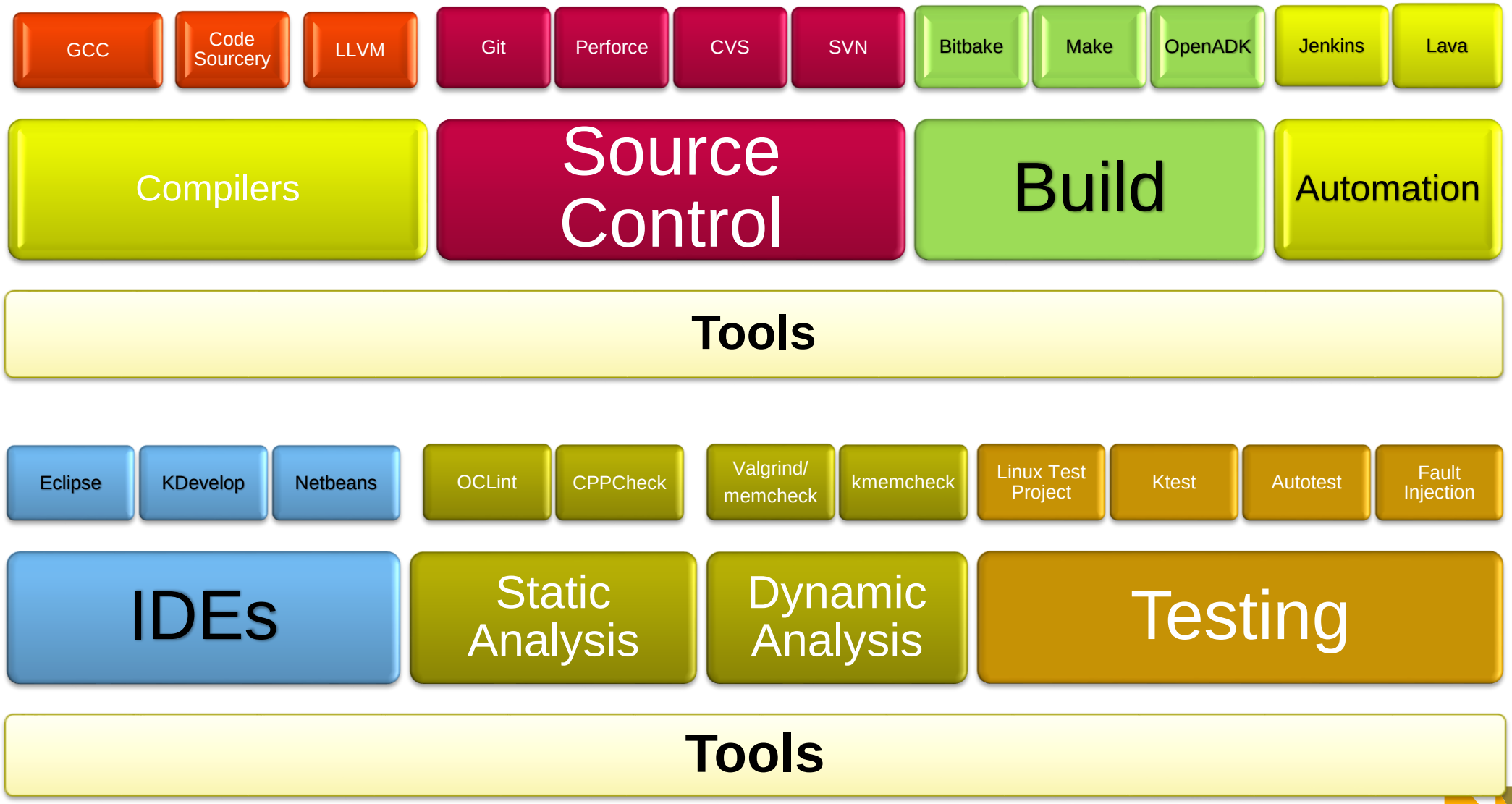




KVM is a type2 hypervisor
Part of vanilla Linux kernel, does not require additional porting effort
Supports ARM and PPC architectures
Uses ARM virtualization extension

XEN is a type1 baremetal hypervisor
Supports ARM architectures
Can use paravirtualization and Full Virtualization
Requires porting effort for new platforms

Tools


GCC
Code Sourcery
LLVM
Git
Perforce
CVS
SVN
Bitbake
Make
OpenADK
Jenkins
Lava
Compilers
Source Control
Build
Automation
Tools
Eclipse
KDevelop
Netbeans
OCLint
CPPCheck
Valgrind/ memcheck
kmemcheck
Linux Test Project
Ktest
Autotest
Fault Injection
IDEs
Static Analysis
Dynamic Analysis
Testing
Tools

Linux Security Overview

Cryptography

- SW & HW based cryptographic services
- Extensive list of ciphers supported
- Cryptographic services are offered in both kernel and userspace
- Dynamic crypto algorithm loading
- Asynchronous and synchronous support

Network Security

- Firewall support- controlling what if information is allowed in the system from the network
- IPSEC / VPN
- Packet Sniffing
- Identd is used for monitoring access to TCP services
- Linux Network IDS – an intrusion detection system for discovering unauthorized access

SELinux

- Access Control Policies
- Protects processes and users from faulty accesses
- Controls over process initialization and inheritance and program execution
- Controls over file systems, directories, files, and open file descriptors
- Controls over sockets, messages, and network interfaces

Seccomp

- Restricts the number of system calls a process can issue
- Extensively used for untrusted application

Others

- Use/Group Permissions
- Secure Boot
- Linux Security Modules (LSM)
- TOMOYO
- AppArmor
- Secure NFS
- Audit
- Smack
- Integrity Management
- Extended DAC
- Linux Security Patch
- Linux Namespaces
- Packet Sniffer

How is Linux Tested

- **The Linux Test Project (LTP)** delivers test suites to the open source community that validate the reliability and stability of Linux. The LTP test suite contains a collection of tools for testing the Linux kernel and related features. https://github.com/linux-test-project/ltp

- **Autotest** -- a framework for fully automated testing. It is designed primarily to test the Linux kernel, though it is useful for many other purposes such as qualifying new hardware, virtualization testing, and other general user space program testing under Linux platforms. It's an open-source project under the GPL and is used and developed by a number of organizations, including Google, IBM, Red Hat, and many others.http://autotest.github.io/

- Also there are certification systems developed by some major GNU/Linux distribution companies. These systems usually check complete GNU/Linux distributions for compatibility with hardware. There are certification systems developed by **Novell, Red Hat, Oracle, Canonical, Google**.

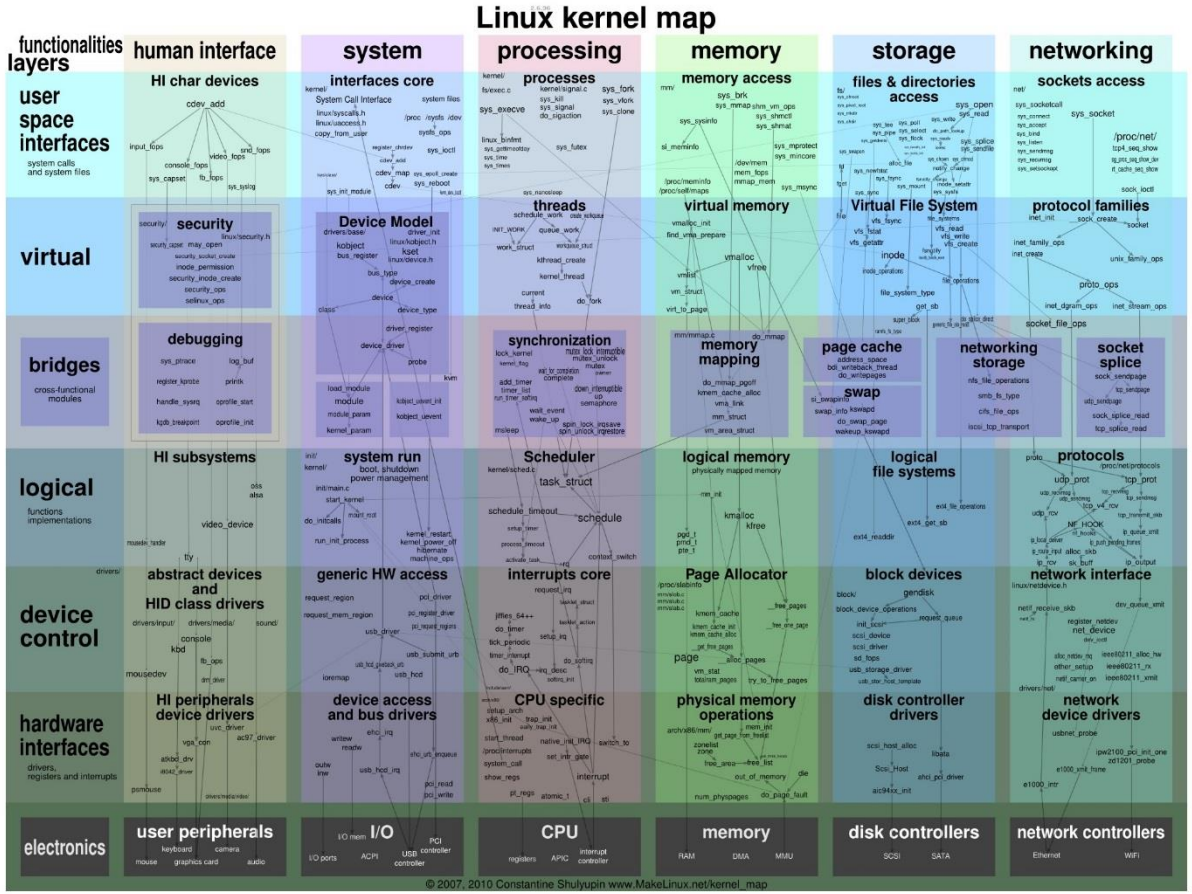
Linux in other industries

- *Android, which is based on Linux and is open source, is the most popular mobile platform. During the second quarter of 2013, 79.3% of smartphones sold worldwide were running Android. Android tablets are also available.*
  - *Source: Wikipedia*
- *In May 2014, W3Techs estimated that 67.5% of the top 10 million (according to Alexa) websites run some form of Unix, and Linux is used by at least 57.2% of all those websites which use Unix*
  - *Source: Wikipedia*
- *99% of the TOP500 supercomputers run Linux*
  - *Source: Wikipedia*

CHALLENGES AND HOW TO SOLVE THEM

Complexity of Linux

- With 15 millions of Linux of code, Linux becomes extremely complex
- Thousands of programmers are open source code developing code in each release



Linux Kernel Development Process vs Automotive Software Process

- No certification
- No formal change control management tool
- Optional Static/Dynamic Code Analysis
- Some Documentation Projects
- No formal requirements

- SPICE or ISO26262 certification
- Very strict change control management
- Static Code/Dynamic Code analysis as a requirement
- Code coverage
- Comprehensive documentation
- Requirements Traceability

THE SAFETY STORY

Concerns over Safety Aspects for Open Source Software

- No formal processes
- Complexity increases the risk of potential bugs
- Testing is performed by developers writing code – there are no formal test deliverables
- Lack of formal documents and formal process and tools
- Liability

Safety Research Linux Kernel Development

- Compliance of vanilla Linux kernel with ISO26262 is unfeasible
- Safety research was performed on a BSP that is based on OSS
- Since Linux kernel is modular, separation of non-safety compliant components is possible
- Addition of safety related features to Linux kernel is possible (i.e SMPU)

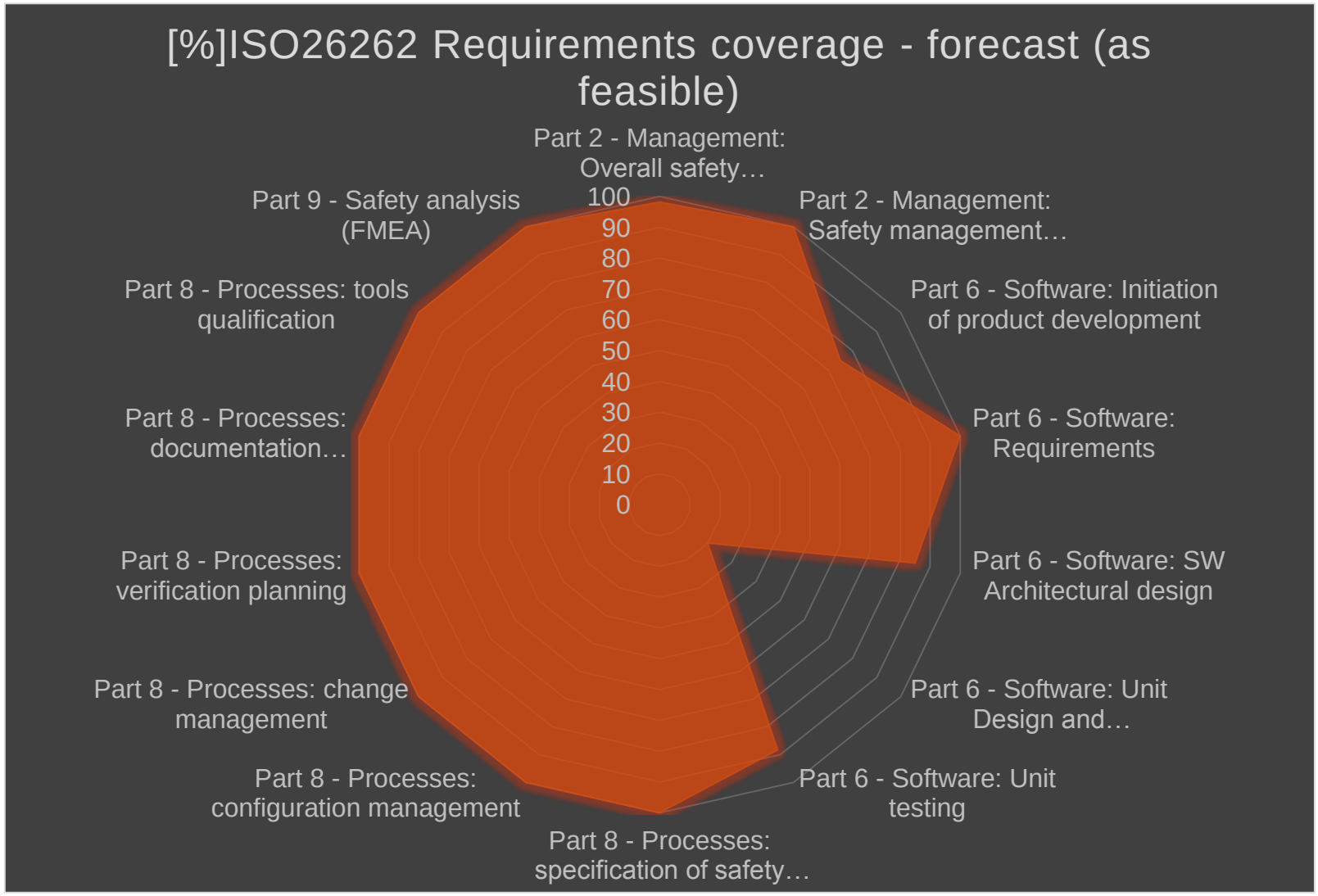
Linux Tools and Safety

- There are also systems for dynamic analysis of Linux kernel:
- **Kmemleak** is a memory leak detector included in the Linux kernel. It provides a way of detecting possible kernel memory leaks in a way similar to a tracing garbage collector with the difference that the orphan objects are not freed but only reported via /sys/kernel/debug/kmemleak.
- **Kmemcheck** traps every read and write to memory that was allocated dynamically (i.e. with kmalloc()). If a memory address is read that has not previously been written to, a message is printed to the kernel log. Also is a part of Linux Kernel
- **Fault Injection Framework** (included in Linux kernel) allows for infusing errors and exceptions into an application's logic to achieve a higher coverage and fault tolerance of the system.

Linux Kernel Development and Safety in NXP

ISO26262	Comments
Part 2: Management	Linux BSP follows (with small exception like detailed design) the NXP AMP SW ISO26262 compliant process Creation of safety plan, safety case,
Part 6: Software	Phasing and Task planning Linux high level requirements definition Linux detailed requirements for new drivers Linux high level architecture and documentation for OSS Detailed design for written from scratch components Static Code Analysis / Dynamic Code Analysis Linux Coding Guidelines Unitesting Code Coverage using gcov Test Traceability Test documentation (Test plan, Test Specification and Traceability Matrix) Checkpatch.pl Linux Kernel Fault Injection Framework
Part 8: Processes	Requirements management, Ticketing Peer reviews Configuration management, **Tools Qualification**
Part 9: Analyses	FMEA

Safety Compliance Research



Break

TECHNICAL SOLUTION

Linux Kernel Tiny Configuration

- Minimal Kernel Configuration with reduced feature set and optimized for size
- UART console and minimal driver set
- 800kb in size
- Booting is done in less than one second
- Typically runs from flash
- User interaction through serial console

Userspace Drivers

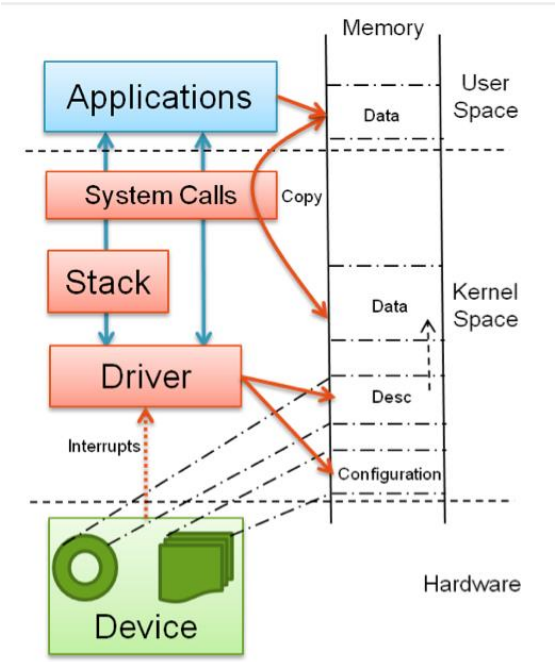


Figure 1: Kernel space network driver

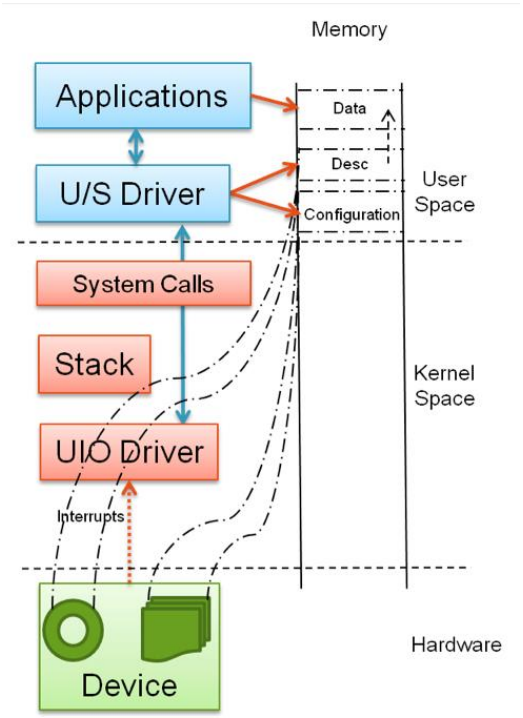
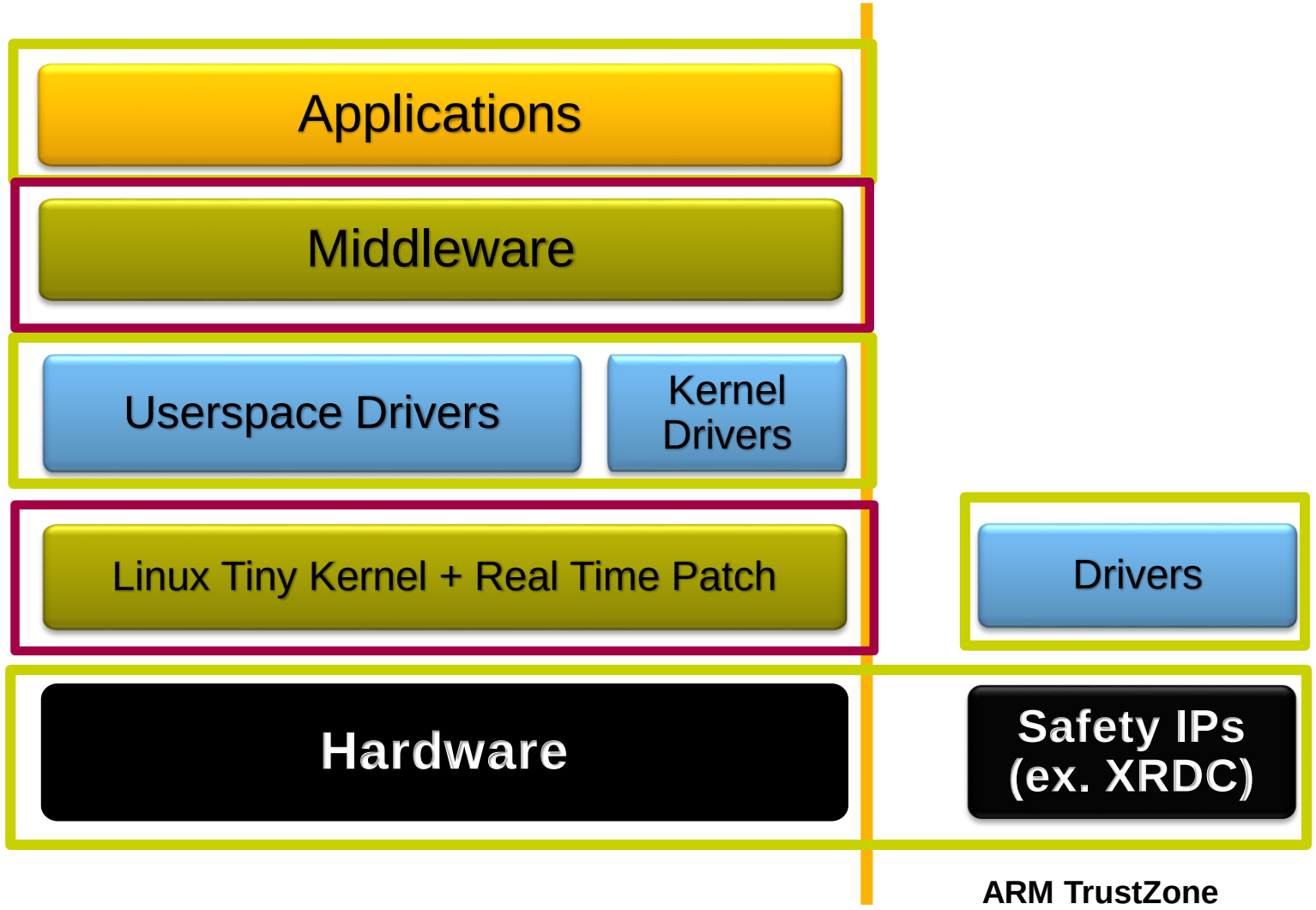


Figure 2: User space network driver

Source: www.embedded.com

- Userspace drivers are linked directly to application code
- Userspace drivers Other licenses than GPL can be used
- A userspace driver is less prone to crash the entire system – thus it is more safe
- Userspace programming is less restrictive than kernel programming
- Userspace drivers are more portable

Software Architecture



Safety Compliance

Safety Compliant

Partial Safety Compliant

Source of Development

Open Source Community

SOC Provider

Others

Other aspects

- Kernel upgrades should be restricted to minimum – as changes to the kernel need to be minimized

- Safety and Security are two separate aspects – Linux kernel is secure

- Userspace code is more protected

S32V234 AND SAFETY FEATURES AND LINUX

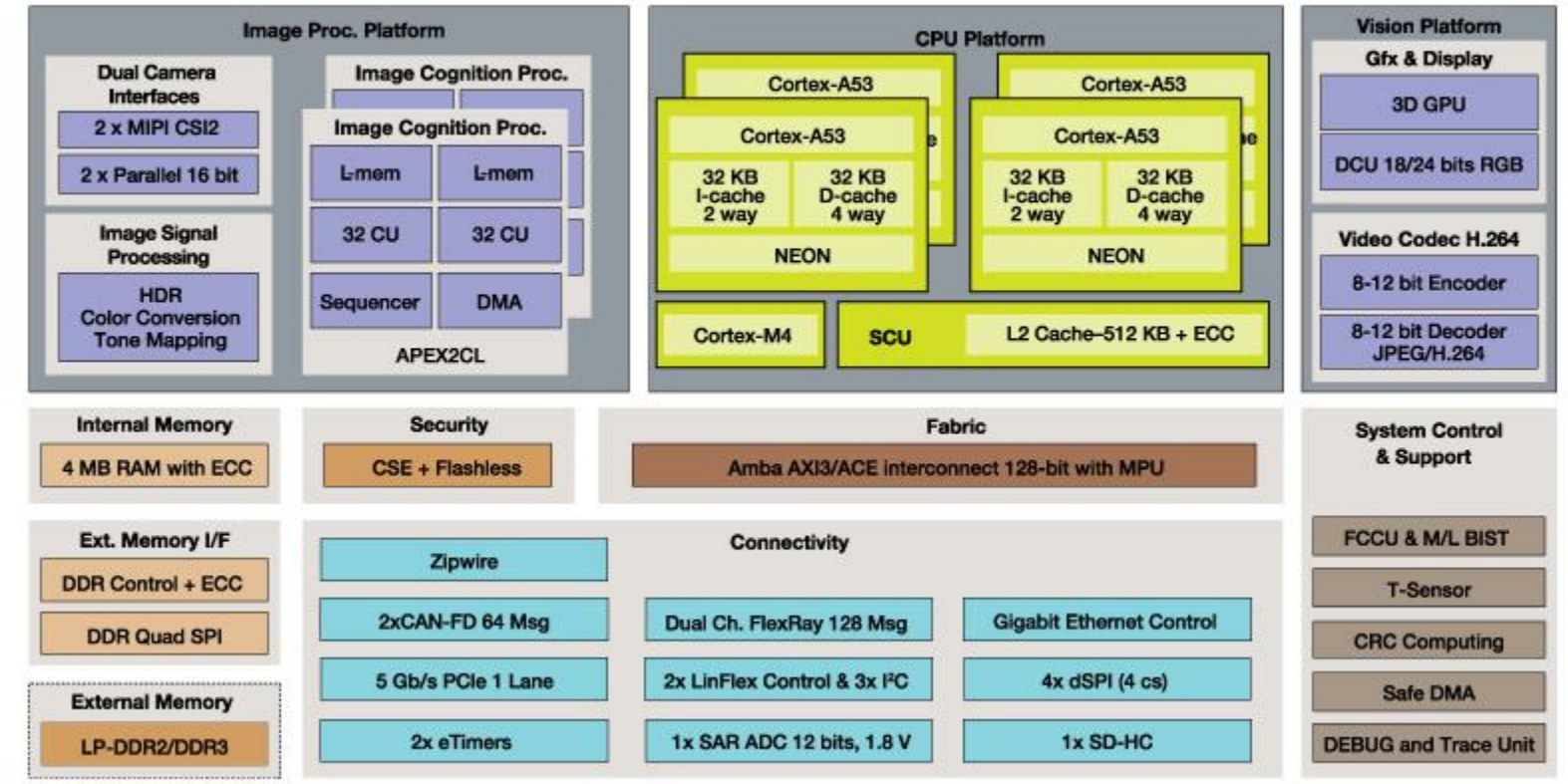
S32V234 Platform

- Targets ISO 26262 ASIL B applications
- Quad ARM Cortex®-A53 cores running at 1GHz
- Dual APEX-2 image cognition processor cores enabled by OpenCL™
- Hardware security encryption
- 3D GPU (Vivante GC3000)
- MIPI CSI2 and parallel image sensor interfaces
- 4MB on chip system RAM
- Embedded image signal processing for HDR, color conversion, tone mapping, etc.

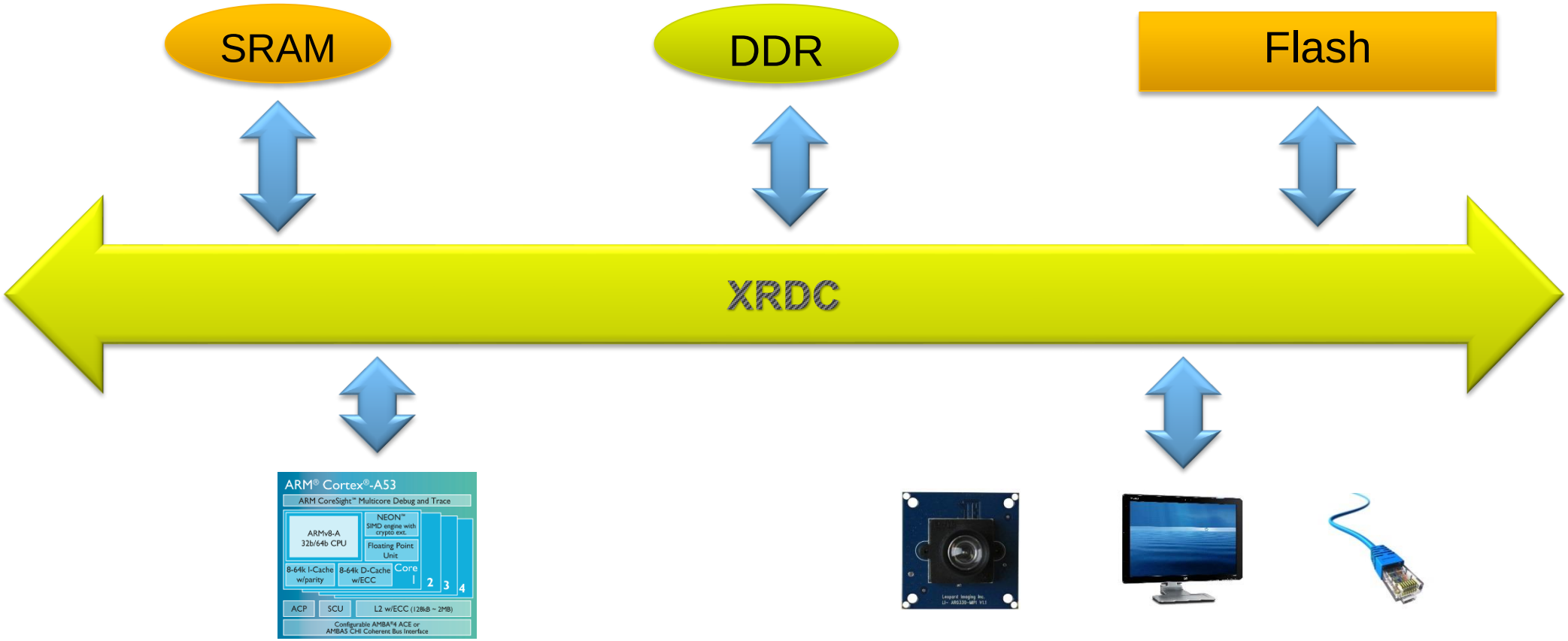
Safety Features

- ASIL B Compliant
- Extended Resource Domain Controller
- Redundancy Control and Checker Unit
- Fault Collection and Control Unit
- Memory Error Management Unit

S32V234 Block Diagram



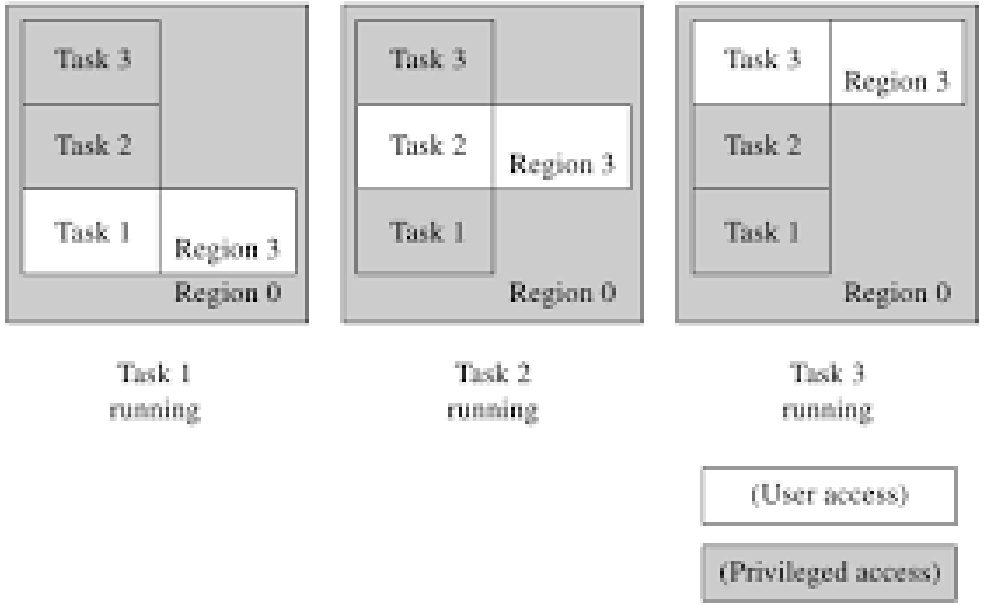
Extended Resource Domain Controller



The Extended Resource Domain Controller (XRDC) provides an integrated, scalable architectural framework for access control, system memory protection and peripheral isolation.

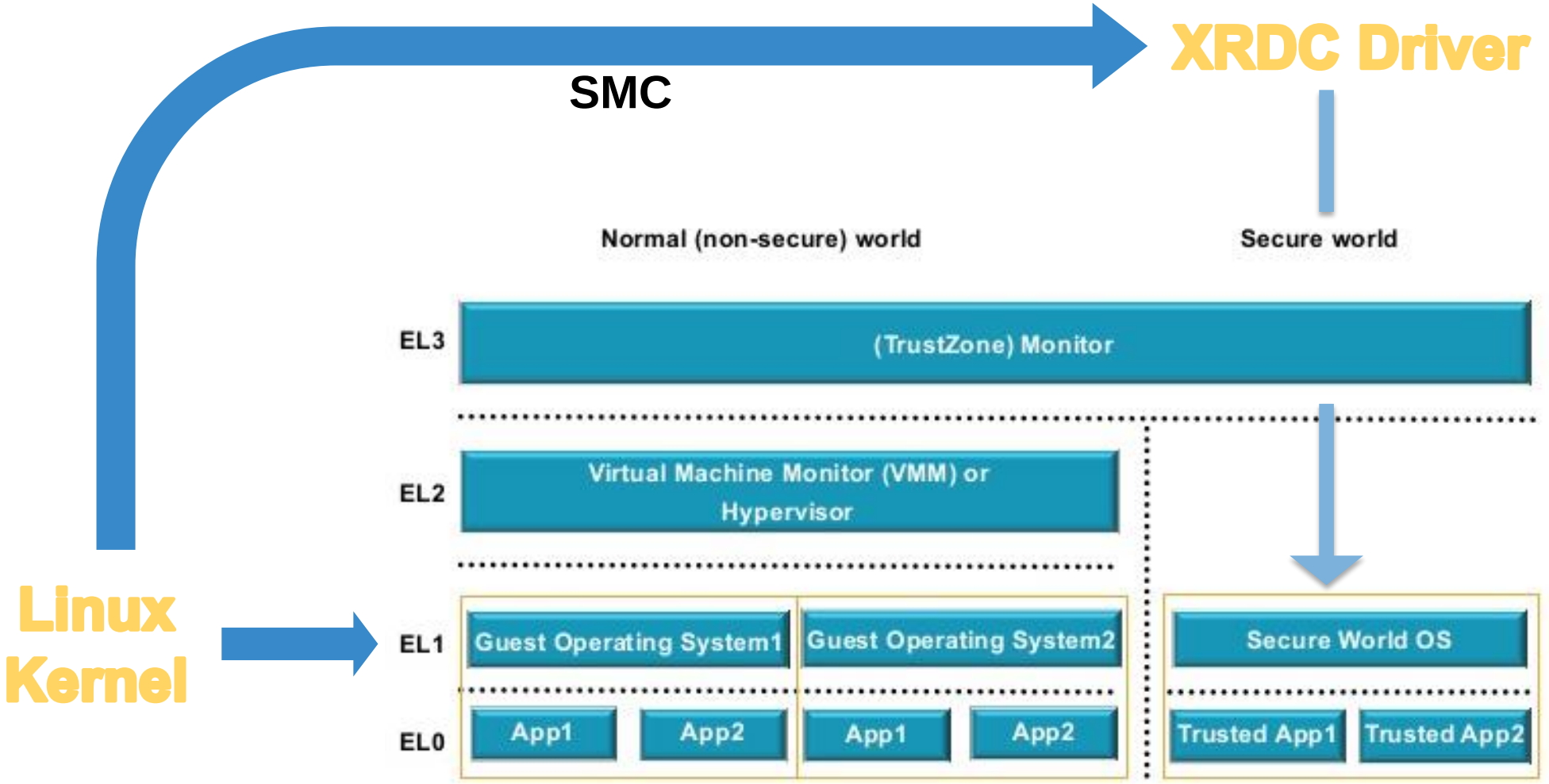
Enabling Safety Features in Linux

- Extended Resource Domain Controller
  - Assign to each task a memory descriptor
  - Separate tasks and ensure they do not overlap



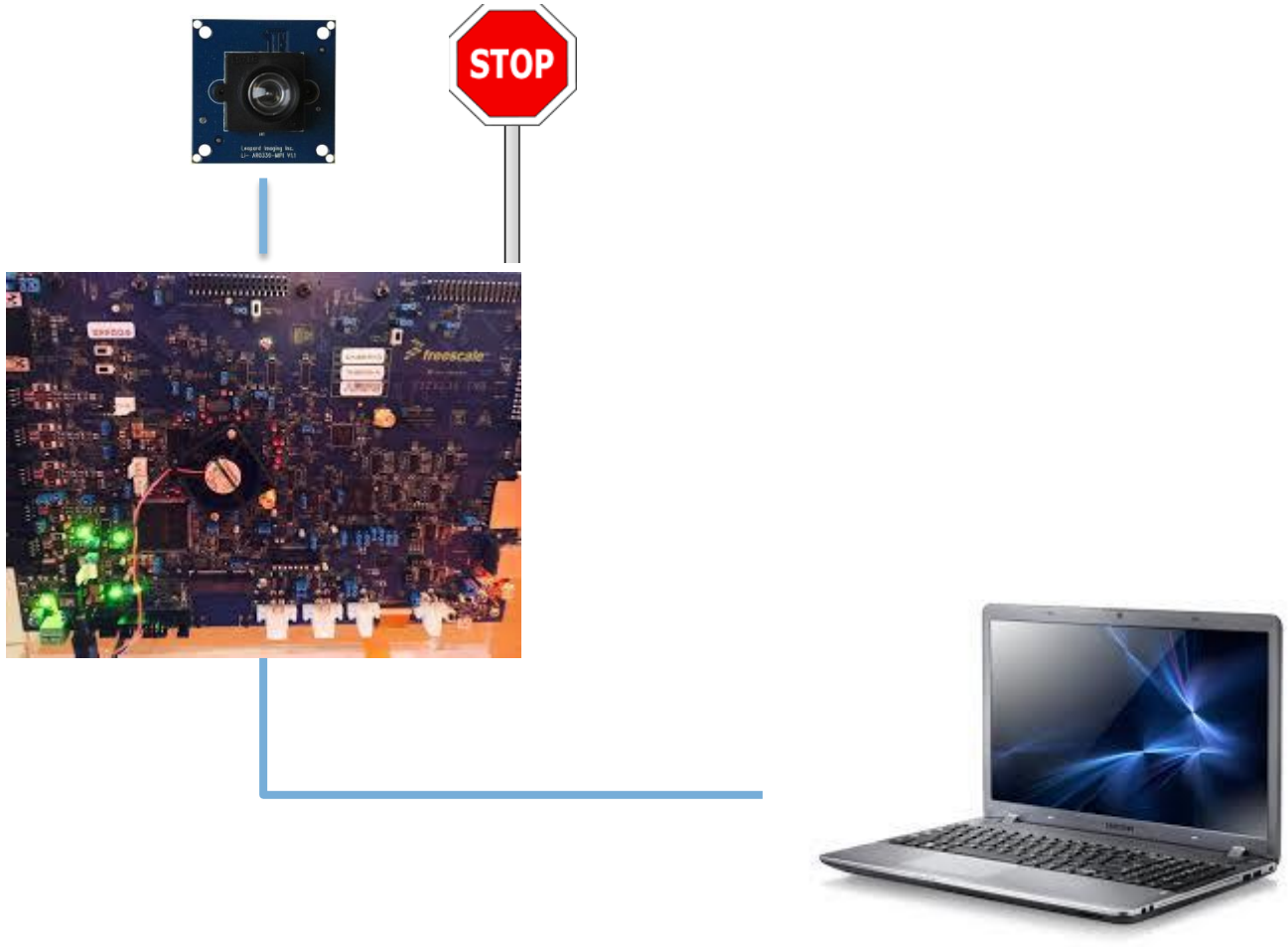
- Implement task restart mechanism
  - But do not crash the entire system

ARMv8 Architecture and how it can be used with Linux



DEMO

Layout



LEGAL ASPECTS

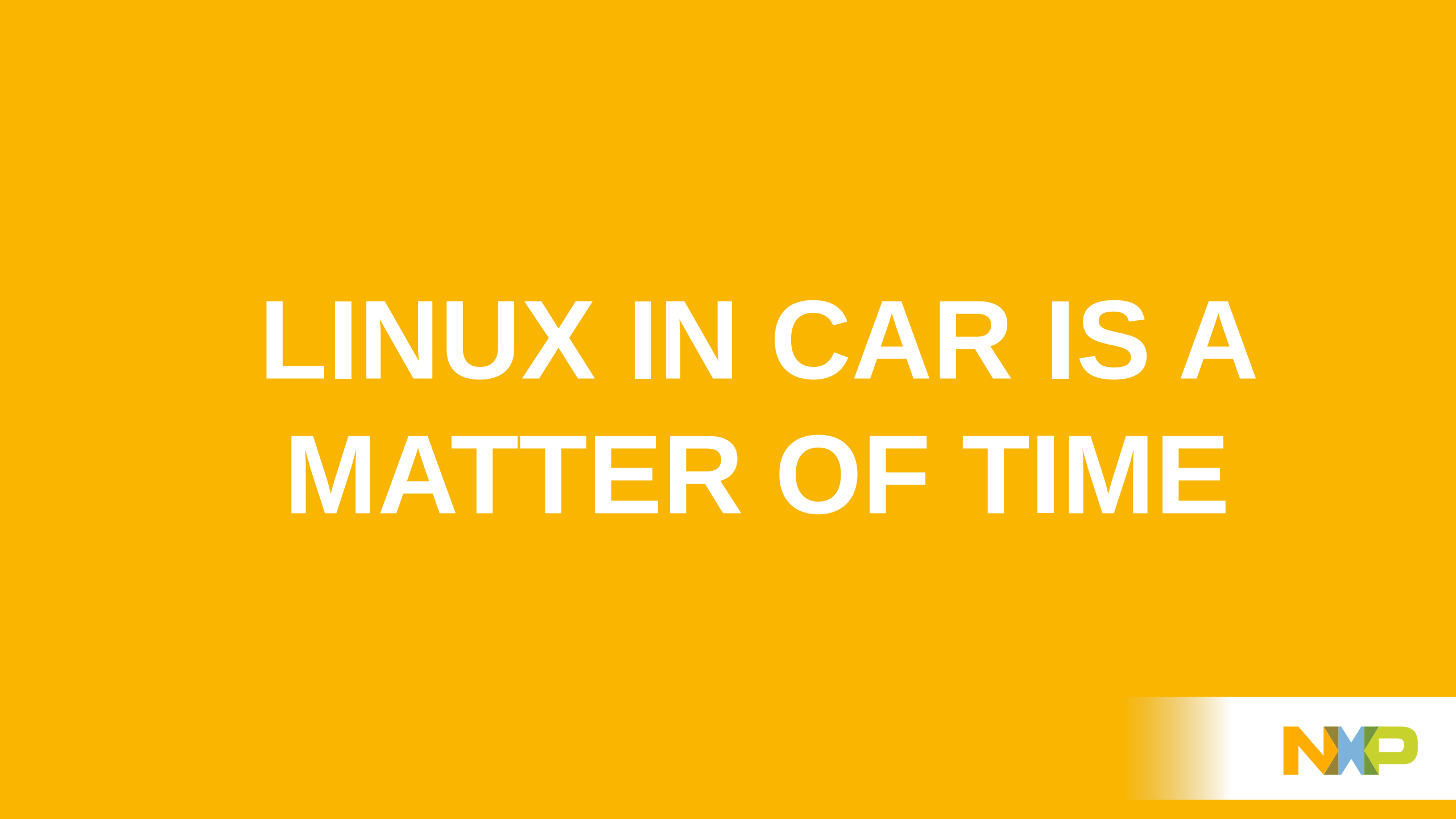
Legal Aspects

- GNU General Public License (GPL version 3), has a clause forbidding the payments of royalties on copies of the OSS
- When using an OSS, you receive it in an “as is” basis; the community does not take liability for bugs found in the kernel code
- All drivers developed in kernel need to share their source to everybody using it
- Protecting your code means userspace drivers

CONCLUSION

- Software importance to Autonomous Drive is divinatory
- Linux and OSS offer enough components and tools to cover Autonomous Drive needs
- Safety is still a challenge and Safety standards need adaptation

LINUX IN CAR IS A MATTER OF TIME



SECURE CONNECTIONS
FOR A SMARTER WORLD