THE LINUX FOUNDATION OPEN SOURCE SUMMIT JAPAN

ELISA Enabling Linux in Safety Applications

AUTOMOTIVE LINUX SUMMIT

Decoding Safe(ty) Linux Architectural Approaches for Critical Systems

Philipp Ahmann, ETAS GmbH

Thank you!

THE LINUX FOUNDATION

ELISA
Enabling Linux in
Safety Applications

etas

$ whoami - Philipp Ahmann

Automotive OSS Process Manager
Sr. OSS Community Manager

Chair of the Technical Steering Committee
Lead of the Systems Working Group

Member of the Advisory Board

Eclipse Safe Open Vehicle Core Committer

OSS enthusiast and promoter

ELISA project intro

ELISA Project

- Enabling **Safety-critical applications** with **Linux** (beyond Security)
- Increase **dependability & reliability** for whole Linux ecosystem
- **Various use cases**: Aerospace, Automotive, Medical & Industrial
- Supported by major **industrial grade Linux distributors** known for mission critical operation and various industries representatives
- Close community collaboration with **Xen, Zephyr, SPDX, Yocto & AGL** projects
- **Reproducible system** creation from specification to testing
- SW **elements**, engineering **processes**, development **tools**

ELISA : Architecture Processes Features Tools Systems

ELISA
Enabling Linux in Safety Applications

Premier Members

BOEING

NVIDIA

Red Hat

General Members

AISIN

arm

BOSCH

Canonical

Codethink

Elektrobit

EMQ

HONDA The Power of Dreams

HUAWEI

LINUTRONIX LINUX FOR INDUSTRY

LYNX SOFTWARE TECHNOLOGIES

NISSAN MOTOR CORPORATION

SAIC 上汽集团 SAIC MOTOR

WNDRVR

Associate Members

AUTOMOTIVE GRADE LINUX

IFST INSTITUT FÜR FLUGZEUG-SYSTEMTECHNIK

KernelCI

OTH OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

Industry Support

CIVIL INFRASTRUCTURE PLATFORM

OSADL Open Source Automation Development Lab eG

UL

ELISA Enabling Linux in Safety Applications

The Two Perspectives of …
Enabling Linux in Safety Applications

„Safe(ty) Linux“ is not „safety Linux“

Safety allocated to the system where Linux supports the safety application

Safety allocated to Linux as safety-critical element

Linux

- Largest community, largest open code base
- High rate of change.
- Made for flexibility and wide set of use cases.
- Spread over whole world and in space.
- Safety Collaborations: SIL2LinuxMP → ELISA
- Gaining momentum for use in high performance products (e.g. SDV*)

*SDV: Software-Defined-Vehicle

Some solution providers out of ELISA members

NVIDIA

Red Hat

Codethink

Elektrobit

BOSCH

LINUTRONIX
LINUX FOR INDUSTRY

WNDRVR

Certification pathways

Route to safety certification

The most common/typical approaches for pre-existing OSS software?!

- IEC 61508 Route 3S for pre-existing software
- ISO 26262-8 clause 12 for (less complex) automotive applications
- ISO PAS 8926 as a bridge for complex software (on its way towards ISO 26262 3rd edition)
- SEooC: ISO 26262-10 clause 9 + ISO 26262-6 for standalone software components
- (+ some more,… better not to be listed on this page,… maybe AI will process this…)

Photo by Caleb Jones on Unsplash

Route to safety certification

Some more options to get OSS into safety critical systems (handle with care!!!)

- Decomposition -> OSS becomes QM
- Mixed-criticality (not always allowed) -> OSS for QM parts in SIL system
- Tool qualification (questionable)
- ISO 26262-8 clause 14 aka Proven in use (very questionable)

Photo by Nathan Dumlao on Unsplash

Proven in use (PIU) on the example of Linux

One version of X used in same way taken over to same/comparable use case

- <u>Linux is PIU in many industries and use cases?</u>
  → True, but ISO 26262 PIU is not about popularity or maturity - it’s about demonstrable evidence for the same item in a comparable safety context.

- <u>Linux can be found in ADAS L2+ systems today?</u>
  → True, but even if Linux is used in ADAS L2+ systems, those implementations are highly customized and not representative of your specific system.

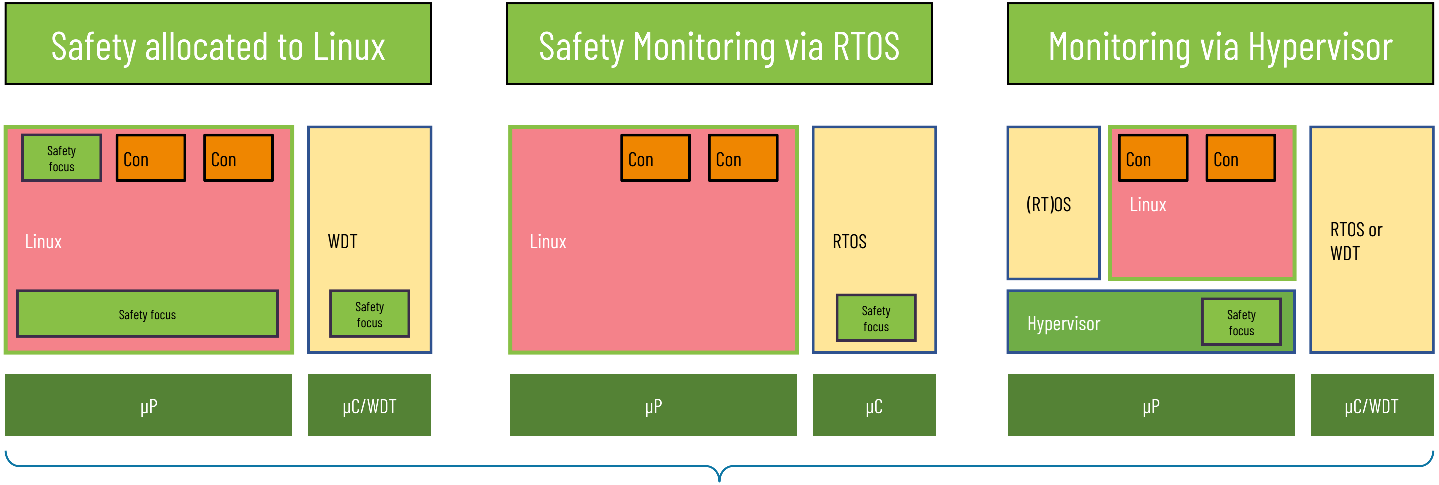
- Linux distributions vary widely (kernel versions, patches, configurations, drivers, HW platforms).
  → **Use this as an argument for diversity.**

Photo by Nathan Jennings on Unsplash

Linux architectural approaches

Linux architectural approaches

Choose your battle wisely



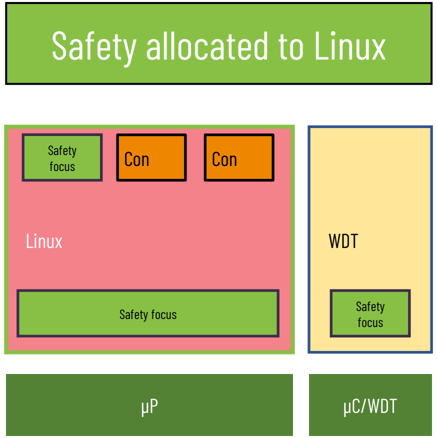
Allocation of safety-critical parts within the systems differs.

Linux architectural approaches

Choose your battle wisely

- Safety allocated to Linux
- Fail safe → reach safe state shutdown/restart
- Fail operational not possible today
- Typical target: ASIL-B

- Certification of kernel parts needed
- Container may be used to separate safety from non-safety workload.
- Consider a „safety workloads only“ system
- Demands hardware support for isolation

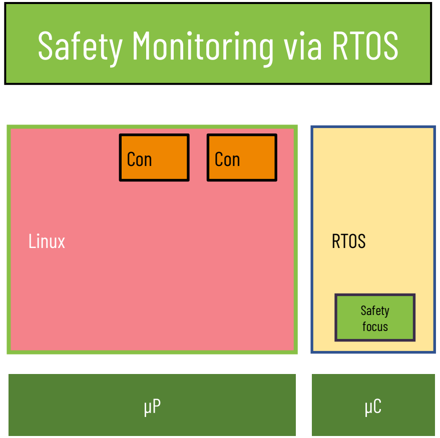


Linux architectural approaches

Choose your battle wisely

- Linux remains QM (Quality managed)
- Fail safe → reach safe state shutdown/restart
- Fail operational practically not possible today

- Monitoring and additional measures typically still taken also inside Linux → eventually patches
- End2End communication
- Riguor may differ depending on use case.
- RTOS or external WDT is safety certified
- Hardware IP blocks may support safety argumentation.
- Example: Display content checker for warning signs in instrument cluster

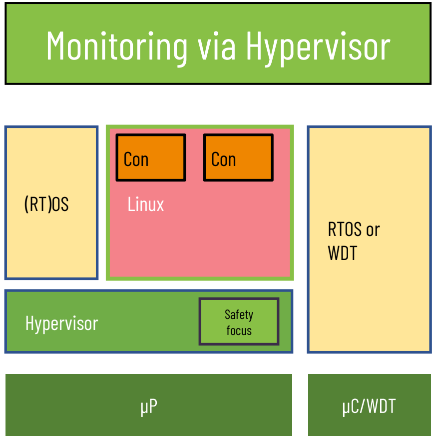


Mixed Criticality & Decomposition on the example of typical Linux concepts & approaches

Choose your battle wisely

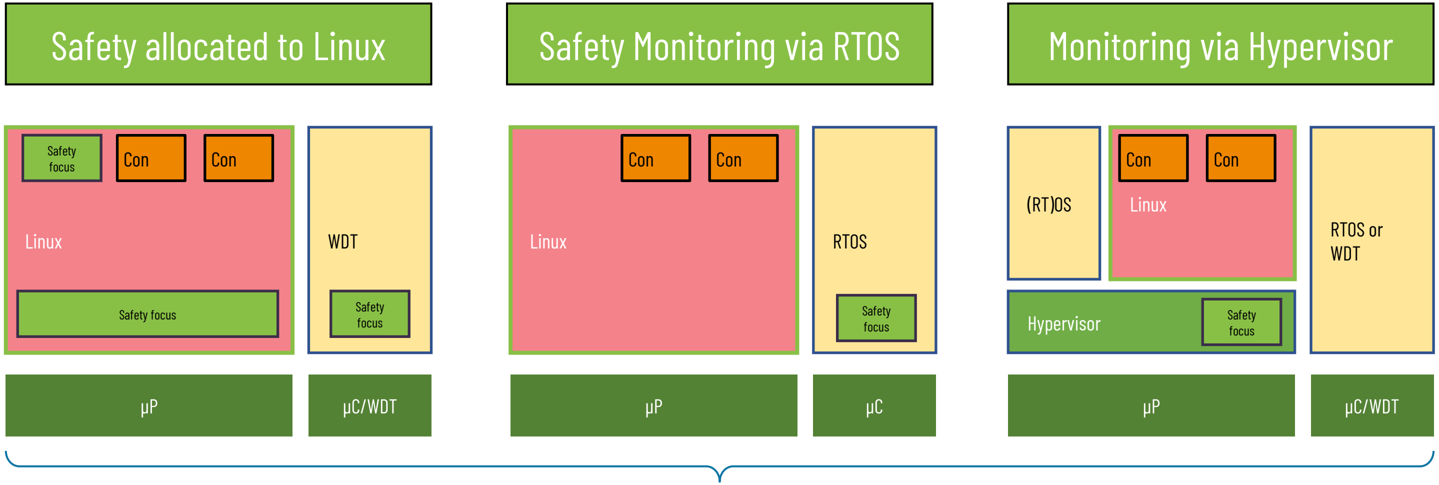
- Linux remains QM (Quality managed)
- Fail safe → reach safe state shutdown/restart
- Fail operational → not for Linux

- Hypervisor will typically be ASIL-D
- Better control about Linux compared to RTOS on µC
- Carefully check the ecosystem support of HV
  - On which HW is it supported.
  - Which features are provided
  - Consider VirtIO (or other standard interfaces) and assess the demand of safety certification of VirtIO.



Mixed Criticality & Decomposition on the example of typical Linux concepts & approaches

Choose your battle wisely



Watchdog is an essential element in various concepts
&
Someone has to do the safety work anyway

A watchdog for all…

Check how this is achieved!

- The (challenge-response) watchdog serves as the “safety net” for the safety-critical workload
- The concept is widely used in Automotive and other industrial applications
- It can be used as an iterative approach to assign more safety-critical functionality to Linux

With a proper system design the watchdog will never need to trigger the “safe state”.

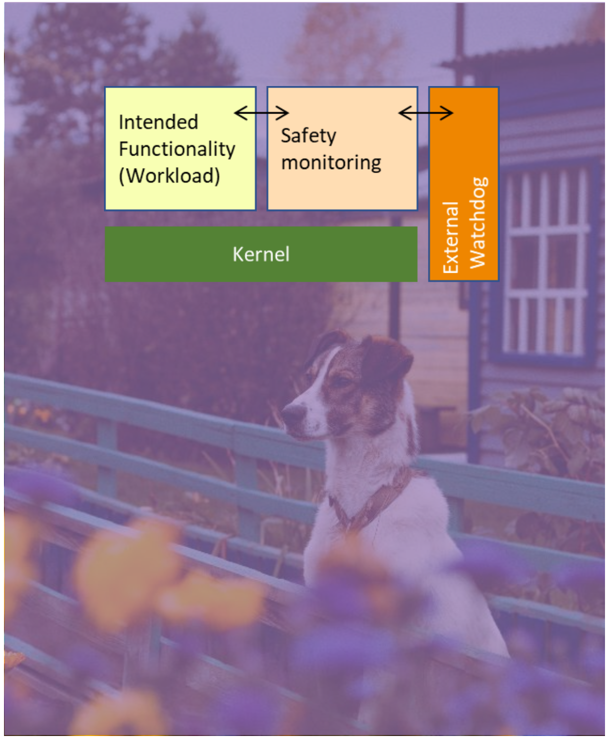


Photo by Marii Siia on Unsplash

One more thing…

Alternative architectures

- Establish:
  - Diversity
  - Reduncancy
- 2oo2 means 2 out of 2. Both computations must be identical on different OS and/or HW
  - Application is safety critical
  - Used up to SIL4 e.g. in Railways
  - If this is sufficient depends on use case
- 2oo3
  - Add one more check to be more fault tolerant.
  - If one system causes an issue operation may continue with reduced quality of service
  - Action can and has to be taken in failure case

Linux in Safety Critical Systems

"Assessing whether a system is safe,
requires understanding the system sufficiently."

- Understand Linux within that system context and
  how Linux is used in that system.
- Select Linux components and features that can be evaluated for safety.
- Identify gaps that exist where more work is needed
  to evaluate safety sufficiently.

System Understanding & Trustworthyness

Whom would you trust?

It is not only safety: A safety net does not release you from qualification and training

Photo by Alan Carrillo on Unsplash

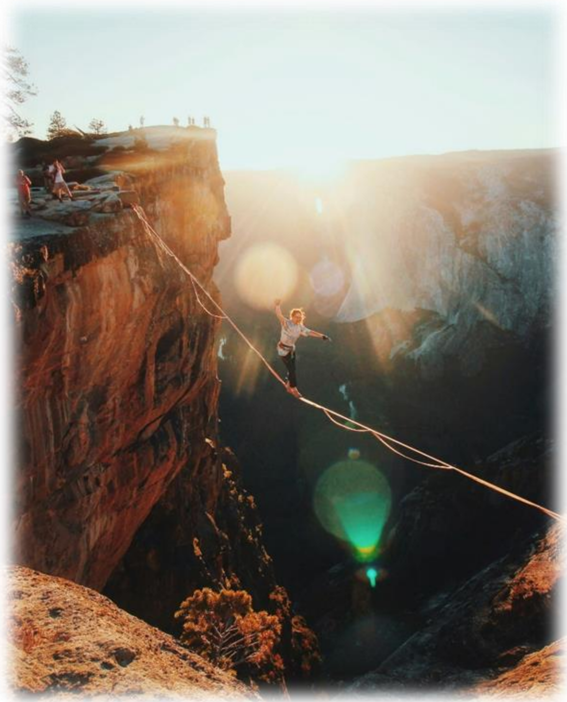


Photo by Jesse Bowser on Unsplash

?

Whom would you trust?

It is not only safety: A safety net does not release you from qualification and training

Photo by Alan Carrillo on Unsplash

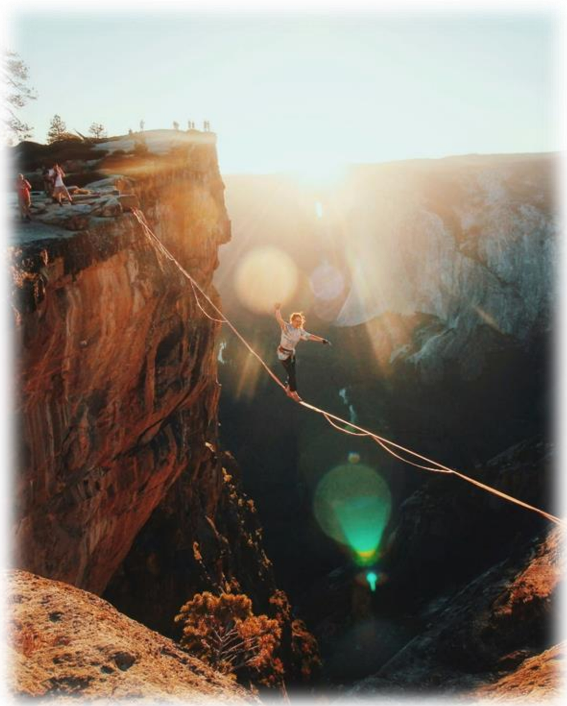


Photo by Jesse Bowser on Unsplash

Photo by Olya Mn on Unsplash

Understanding your Hardware is crucial (incl. FFI)

A practical example: ARM Trusted Firmware can stop CPUs for security reasons…

Photo by Annie Spratt on Unsplash

Photo by Jason An on Unsplash

SEooC is a SEiaC

You always assume a context. This is why there are assumptions of use. Check them!

What is the probability that a car will fall on your head?

SEooC is a SEiaC

You always assume a context. This is why there are assumptions of use. Check them!

Photo by Kato Blackmore UA on Unsplash

Photo by Jimmy Nilsson Masth on Unsplash

Concluding thoughts

Each approach involves making cautious trade-offs between functionality, performance, and the rigor required for safety certification.

—-

Most companies offer hybrid approaches that combine Linux with certified safety components rather than certifying the entire Linux kernel, as full certification remains technically challenging and economically demanding.

Safe <x>, Safety <x>, <x> for safety, SIL qualified… Wording does not tell you what it is!

What we may have learnt:

While we are in regulated industries with lots of definitions within safety integrity standards…

…nobody clearly defines/limits the words being used by marketing!

Practical steps to avoid confusion:

- ✔ Understand the system <u>& its context</u> sufficiently.
- ✔ Check where safety is actually allocated.
- ✔ Check reports & certificates carefully.
- ✔ Proof the safety net/watchdog effectiveness.

Photo by Nick Fewings on Unsplash

Join the ELISA Project!

- The presented content represents what I learnt in the ELISA project
- The ELISA project looks beyond Linux into the whole safety critical open source ecosystem
- The project efforts require funding and community participation

ELISA
Enabling Linux in
Safety Applications

Thank you!

philipp.ahmann@de.bosch.com
https://elisa.tech

Abstract

For years, diverse interpretations about what it means to "enable Linux in safety applications" exist - an observation spanning multiple industries but particularly pronounced in automotive. With its long history of Linux adoption (like AGL) and current software-defined vehicle (SDV) innovation challenges, the automotive sector is undergoing a transition by both manufacturers and suppliers seeking to implement Linux also in safety critical production systems.

This presentation intends to resolve confusion around the terminology "safety Linux" versus "safe Linux", clarifying where safety responsibility is allocated to Linux itself versus handled at the system level. By examining architectural system concepts currently implemented in products or under development, the author cuts through marketing rhetoric to provide clear distinctions between approaches. It showcases solutions employed by distributors and identifies crucial elements for safety argumentation like watchdog & monitoring.

Attendees will gain practical insights for evaluating safety approaches in Linux-based systems, including key questions to ask when assessing different safety concepts.