Welcome to Day 2!

ELISA
Enabling Linux in Safety Applications

WORKSHOP

Work in Progress - License: CC-BY-4.0

Thank you to our host:

VOLVO

ELISA
Enabling Linux in Safety Applications

WORKSHOP

Work in Progress - License: CC-BY-4.0

Session Schedule

Work in Progress - License: CC-BY-4.0

Day 2 - Thursday 8th - Morning

8:30 Coffee and Warm-up

09:00 **Safety Linux vs Safe(ty) Linux** (Philipp Ahmann, *Paul Albertella*)

10:30 Fika

10:45 **How far do we go at the hardware level?**
An analysis of current state of kernel and integration
(*Olivier Charrier, Alessandro Carminati*)

12:00 Lunch

ELISA Enabling Linux in Safety Applications

WORKSHOP

Work in Progress - License: CC-BY-4.0

Day 2 - Thursday 8th - Afternoon

13:00 Special topic: **PX4Space** *(Pedro Roque)*

13:30 Special topic: **SPDX Safety Profile**, *(Nicole Pappler)*

14:00 Special topic: **Safe Continuous Deployment** (*Håkan Sivencrona)*

14:30 Special topic: **Resilient Safety Analysis and Qualification** (*Igor Stoppa)*

15:00 Fika

15:15 **KernelCI, BASIL & Testing** (*Luigi Pellecchia, Gustavo Padovan*)

16:30 **Requirements Traceability** (*Kate Stewart, Gabriele Paoloni*)

17:45 **Day 2 wrap-up** (*Philipp Ahmann, Kate Stewart)*

18:00 Day 1 ends

18:00 **Pizza party on-site**

ELISA Enabling Linux in Safety Applications

WORKSHOP

Work in Progress - License: CC-BY-4.0

ELISA
Enabling Linux in Safety Applications

WORKSHOP

Safety Linux vs Safe(ty) Linux

Philipp Ahmann (ETAS), Paul Albertella (Codethink)
ELISA WS - Lund, May 7-9, 2025

Route to Safety Certification

- IEC 61508 Route 3S for pre-existing software
- ISO 26262-8 clause 12 approach for automotive applications
- ISO PAS 8926 as a bridge for complex software
- Challenges increase with increased system complexity (like Linux systems)

Work in Progress - License: CC-BY-4.0

back to agenda

Community Challenges For All Projects

- Argument of „OSS development is not organized like commercial software“
- Less influence on maintainers
  (positive & negative – no traditional supplier management)
- Harder to train/direct developers
- Liability of a community?
  (but commercial provider may be liable – insurance)
- Development process: Requirements, traceability, v-model,…
  mapping safety integrity standards

Work in Progress - License: CC-BY-4.0

What you find in the wild (is not always correct)!

Let us ask Claude 3.7 Sonnet: What are approaches to use Linux in safety critical systems? (Not reflecting ELISA's opinion)

Architectural Approaches

Incomplete and (partially) incorrect AI responses:

What you find in the wild (is not always correct)!

- **Hypervisor Isolation**
  - Run Linux alongside a certified safety RTOS on a hypervisor
  - Keep safety-critical functions on the certified RTOS
  - Linux handles non-critical functions with strong isolation
- **Mixed-Criticality Systems**
  - Use Linux for lower safety integrity levels (SIL 1-2)
  - Implement Freedom From Interference (FFI) between components
- **Safety Element out of Context**
  - Develop safety-critical components as separate modules
  - Integrate with Linux through well-defined, certified interfaces

Implementation Strategies

Incomplete and (partially) incorrect AI responses:

What you find in the wild (is not always correct)!

- **Real-Time Linux Extensions**
  - PREEMPT_RT patch for deterministic behavior (Remark, we know it is no patch anymore)
  - Xenomai or RTLinux for hard real-time capabilities
- **Certification-Ready Distributions**
  - Use pre-certified Linux distributions
- **Safety Components**
  - Implement safety monitors to detect Linux failures
  - Deploy watchdog timers and fallback mechanisms
  - Use memory protection and process isolation

Industry Initiatives

Incomplete and (partially) incorrect AI responses:

What you find in the wild (is not always correct)!

- **ELISA Project** (Enabling Linux in Safety Applications)
  - Open-source working groups developing safety-critical Linux architectures
  - Creating tools and processes for certification evidence
- **Safety Case Development**
  - Document comprehensive safety arguments
  - Maintain traceability from hazard analysis to implementation
  - Implement systematic verification and validation processes
- **Qualified Tool Chains**
  - Use certified development tools and compilers
  - Implement rigorous testing frameworks for Linux components

What you find in the wild (is not always correct)!

Each approach involves making deliberate 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.

Understanding the system

back to agenda

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.

Work in Progress - License: CC-BY-4.0

The Two Perspectives of …
Enabling Linux in Safety Applications

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

Safe: Safety allocated to the system where Linux supports the safety application.

Safety: Safety allocated to Linux as safety-critical element.

ELISA Enabling Linux in Safety Applications

WORKSHOP

Work in Progress - License: CC-BY-4.0

The Two Perspectives of …
Enabling Linux in Safety Applications

Safe Linux = QM Linux: "Safe Linux indicates that safety can be allocated to the system which involves Linux, with Linux functioning as a support for safety applications. However, there is no direct safety argumentation or explicit safety certification requirement applied specifically to Linux or the Kernel, as this is achieved in other system layers (application, middleware, etc). For example, Linux might be considered to meet e.g. Quality Management (QM) criteria as mentioned under ISO26262."

Safety-Qualified Linux: "Safety Linux suggests that responsibility for safety argumentation is directly allocated to Linux as the operating system or the Linux Kernel. This implies that Linux or the OS are subject to rigorous safety assessments (and possibly certifications), ensuring that they meet specific safety integrity levels (SILs) required e.g. by ISO26262 or IEC61508."

General rule: Safety can only really be understood in terms of a system, as opposed to an intrinsic property of a component.

Work in Progress - License: CC-BY-4.0

back to overview ⭐

Some solution providers out of ELISA members

Red Hat

Codethink

Elektrobit

WNDRVR

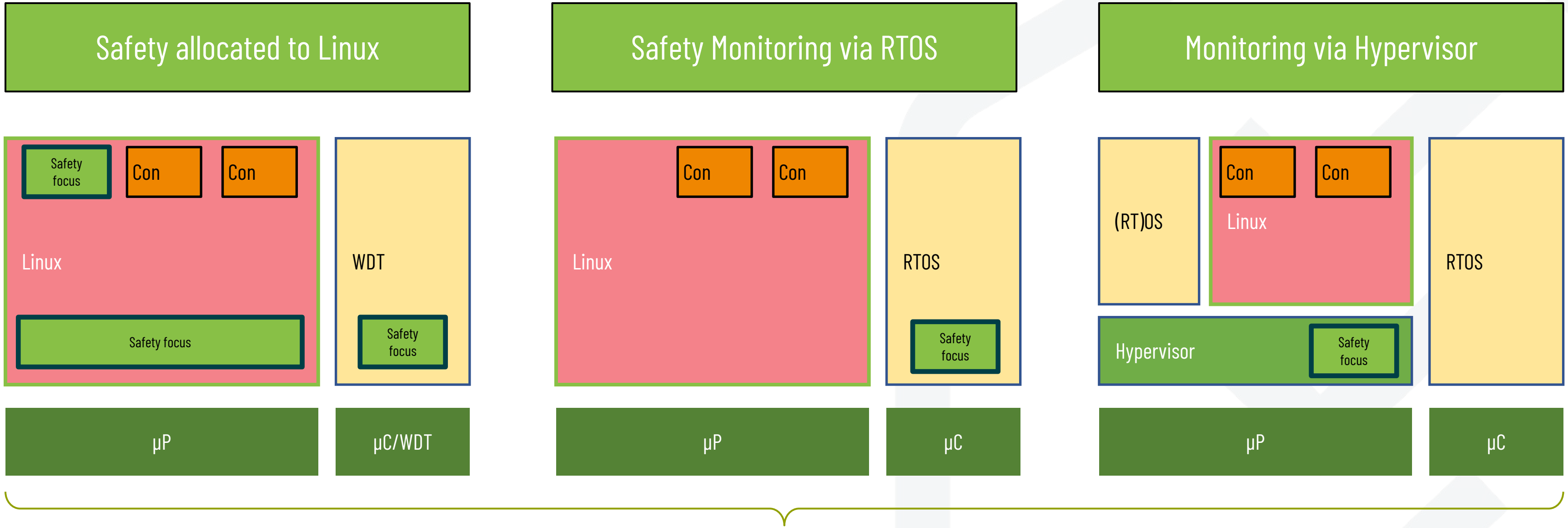
BOSCH

LINUTRONIX

Work in Progress - License: CC-BY-4.0

back to agenda

Typical concepts and approaches



Watchdog is an essential element in various concepts

Work in Progress - License: CC-BY-4.0

External Watchdog

- 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”.

Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units

ELISA Enabling Linux in Safety Applications

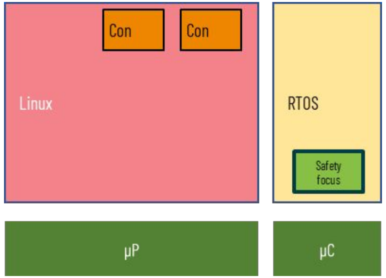
WORKSHOP

Work in Progress - License: CC-BY-4.0

Photo by Marii Siia on Unsplash

Linux in (software-defined) cars beyond IVI

	In the wild & past	Under development	Future?
Conservative	• Rear View Camera • Tell tales (IC Warnings) • E-Mirror • Surround View	• Interior Monitoring • ADAS L2	• ADAS L4
Aggressive	• ADAS L2+ systems	• ADAS L4	• ADAS L3? (cost driver)



Work in Progress - License: CC-BY-4.0

Safe Linux vs. Safety Linux

Let us discuss this!

ELISA
Enabling Linux in Safety Applications

Work in Progress - License: CC-BY-4.0

Photo by Felicia Buitenwerf on Unsplash