Code, Compliance, and Confusion: Open Source in Safety-Critical Products

FOSDEM, Brussels, 31-Jan-2026

Philipp Ahmann, Sr. OSS Community Manager, ETAS GmbH

whoami - Philipp Ahmann

Sr. OSS Community Manager
Automotive OSS Process Lead

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

What is functional safety?

And what is the difference to cyber security?

What is functional safety?

What is functional safety?

- Definition of Safety
  - The freedom from unacceptable risk of physical injury or of damage to the health of people, either directly, or indirectly because of damage to property or the environment.
- Definition of Functional Safety
  - The part of safety that depends on a system or equipment operating correctly in response to its inputs.
  - Detecting potentially dangerous conditions, resulting either in the activation of a protective or corrective device or mechanism to prevent hazardous events or in providing mitigation measures to reduce the consequences of the hazardous event.

What is functional safety?

In Functional Safety you expect that:

Software:

- does behave as specified,
- does not interfere or impair other system components,
- and all possible erroneous events are addressed somehow or somewhere.

And you have sufficient evidence to prove this.

What is functional safety?

In some languages safety & security are the same word!

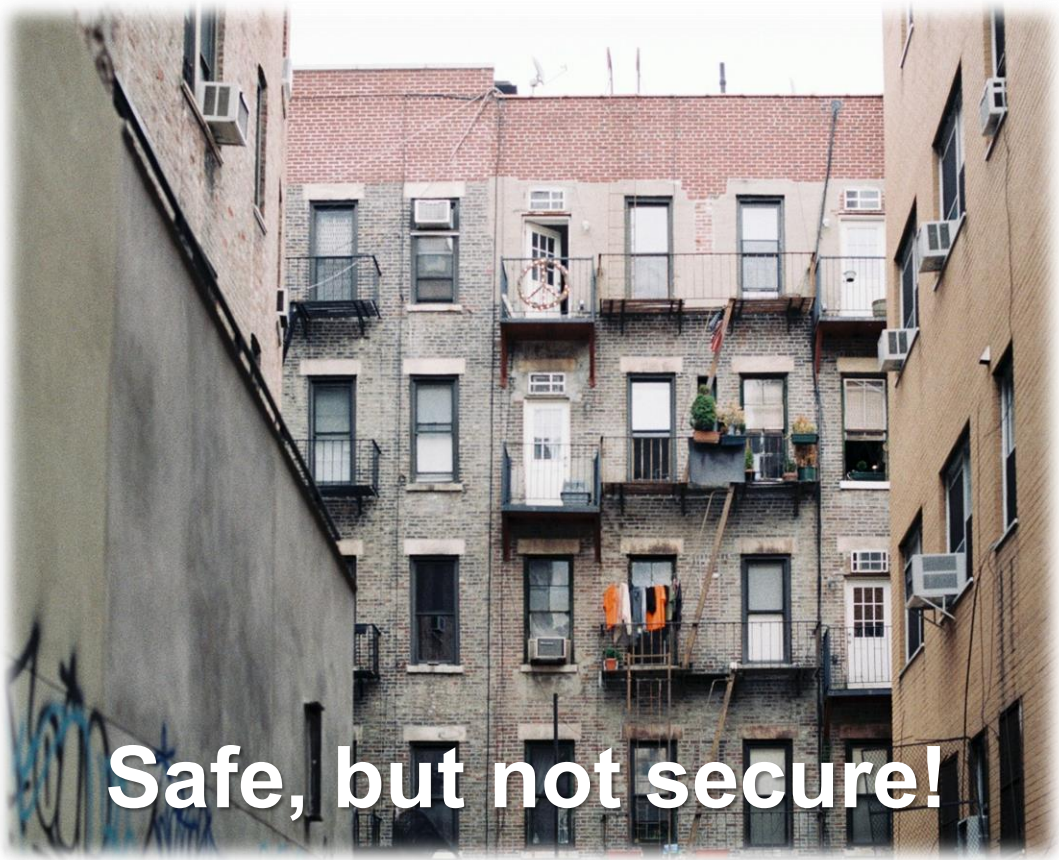


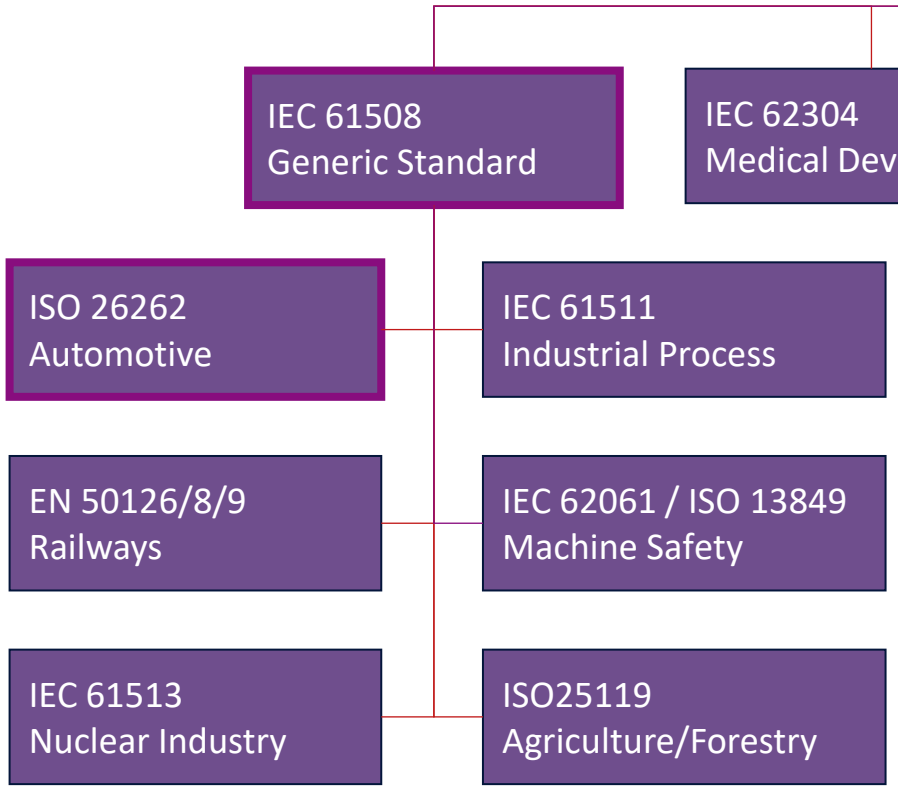
Photo by Annie Spratt on Unsplash

Photo by Jason An on Unsplash

What is functional safety?

Samples of safety (integrity) standards

ELISA
Enabling Linux in Safety Applications



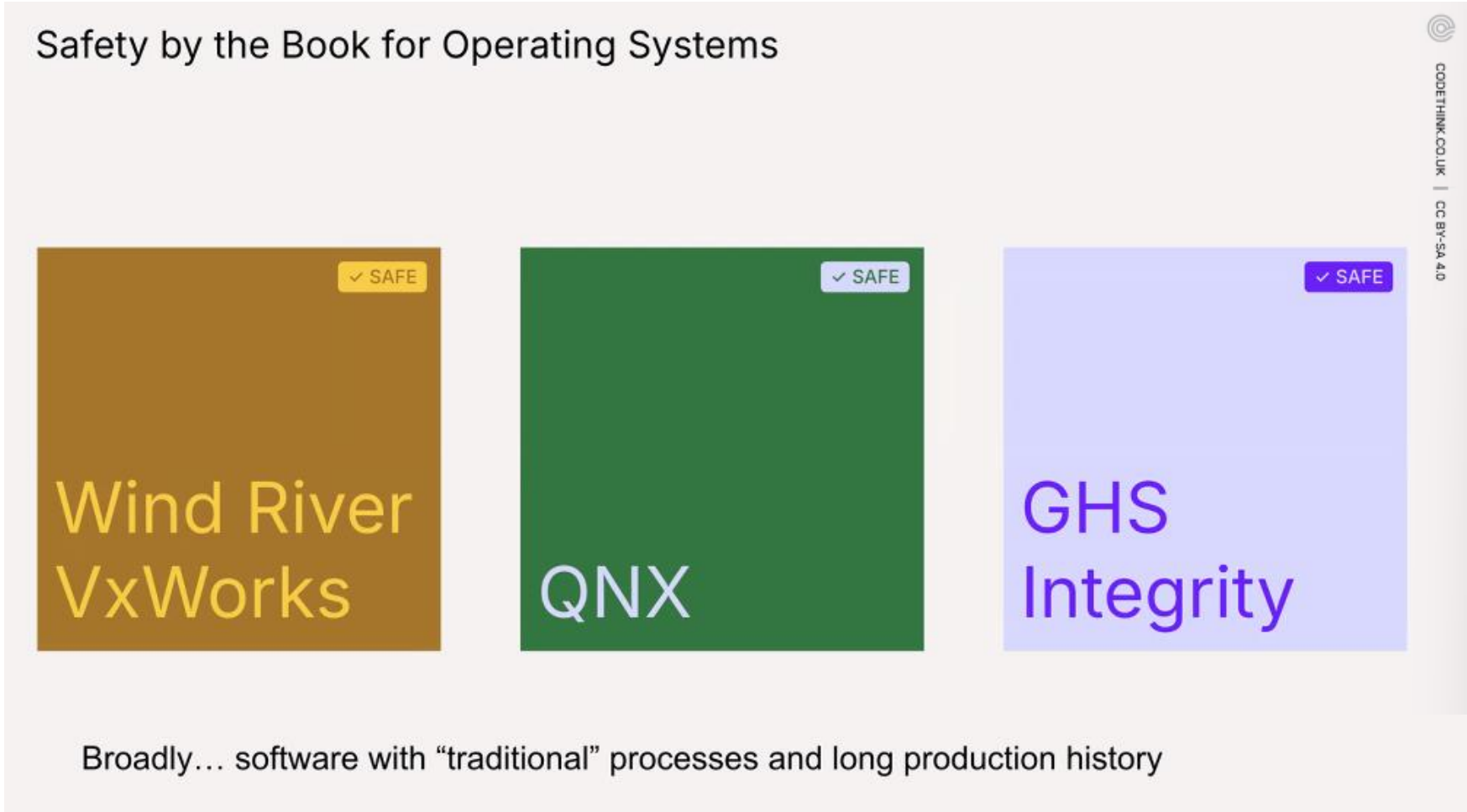
- Standards represent industry best practice and recommend to use these practices
- Share similar demands… requirements, documentation, testing
- Rigor on measures/demands may differ
- All system parts need to be known, tested and managed

Setting the scene

Code, Compliance & Confusion

Where do we come from?

Credits to Codethink, where I took the slide from!

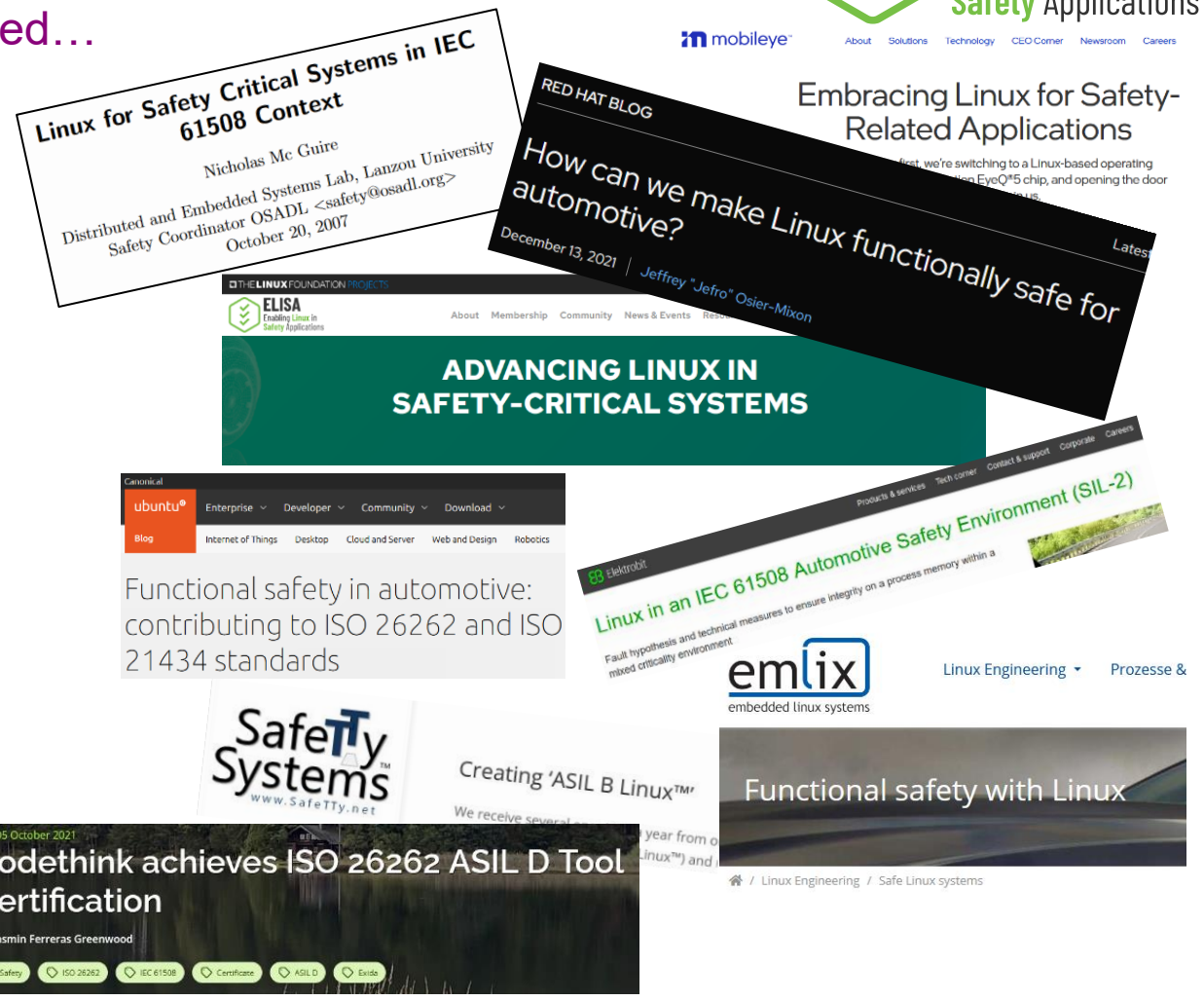


Where are we going to… - The Linux world

Safe <x>, Safety <x>, <x> for safety, SIL qualified…

- Open source software superlative.
- Largest community, largest source base.
- Made for flexibility and wide use cases.
- Spread over whole world and in space.
- Several attempts with certification path.
- Gains again momentum for high performance products (e.g. SDV*)
- Prominent open space examples: SIL2LinuxMP and ELISA

**SDV: Software-Defined-Vehicle*



Example of the safety open source landscape

It is not all about Linux … spot check

Towards safety certification

Compliance

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,… maybe AI will read this…)

Photo by Caleb Jones on Unsplash

Route to safety certification

Okay, and 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

ELISA
Enabling Linux in Safety Applications

– Linux is PIU in many industries and use cases?
  → 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.

– Linux can be found in ADAS L2+ systems today?
  → 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

Tool qualification on the example of Qt Safe Renderer

Confidence in the use of software tools

- Carefully read the certificate as first indication.
- Try to get the full assessment report
  (e.g. mentioned, but not listed at QT website)

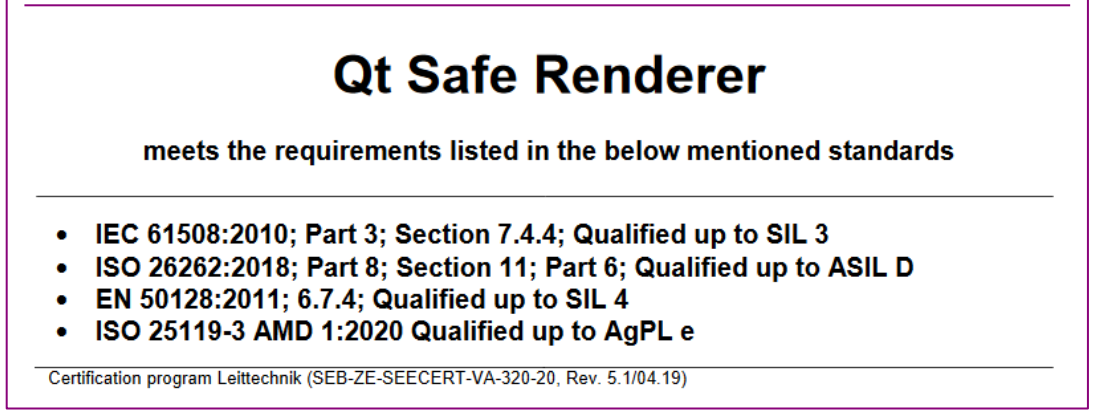
- ISO 26262 has part 6 mentioned up to ASIL-D
- The other standards refer to tool qualification

- Question:

Is the certification really relevant for me?

"The Qt Safe Renderer provides a UI rendering component that can be used to render safety-critical items, such as warning indicators, in functional safety systems."

https://doc.qt.io/QtSafeRenderer/
https://doc.qt.io/QtSafeRenderer/qtsr-delivery.html



Qt Safe Renderer

meets the requirements listed in the below mentioned standards

- **IEC 61508:2010; Part 3; Section 7.4.4; Qualified up to SIL 3**
- **ISO 26262:2018; Part 8; Section 11; Part 6; Qualified up to ASIL D**
- **EN 50128:2011; 6.7.4; Qualified up to SIL 4**
- **ISO 25119-3 AMD 1:2020 Qualified up to AgPL e**

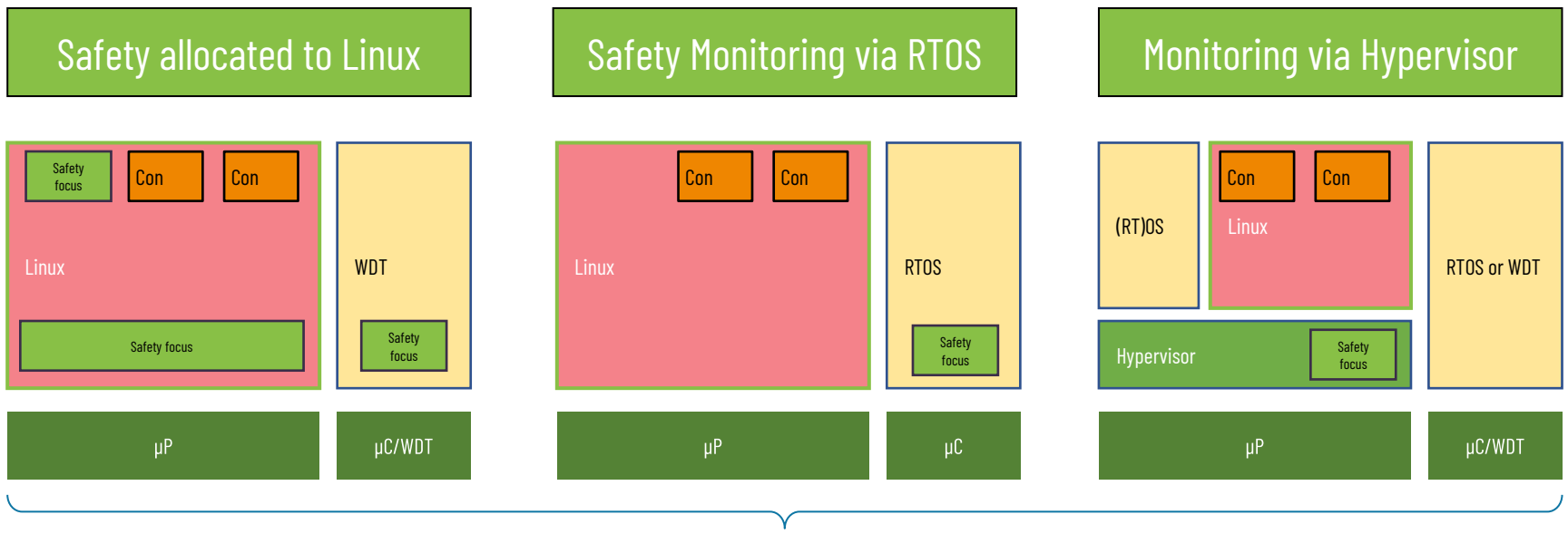
Certification program Leittechnik (SEB-ZE-SEECERT-VA-320-20, Rev. 5.1/04.19)

Towards safety certification

Compliance & Concepts

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

Choose your battle wisely



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”.

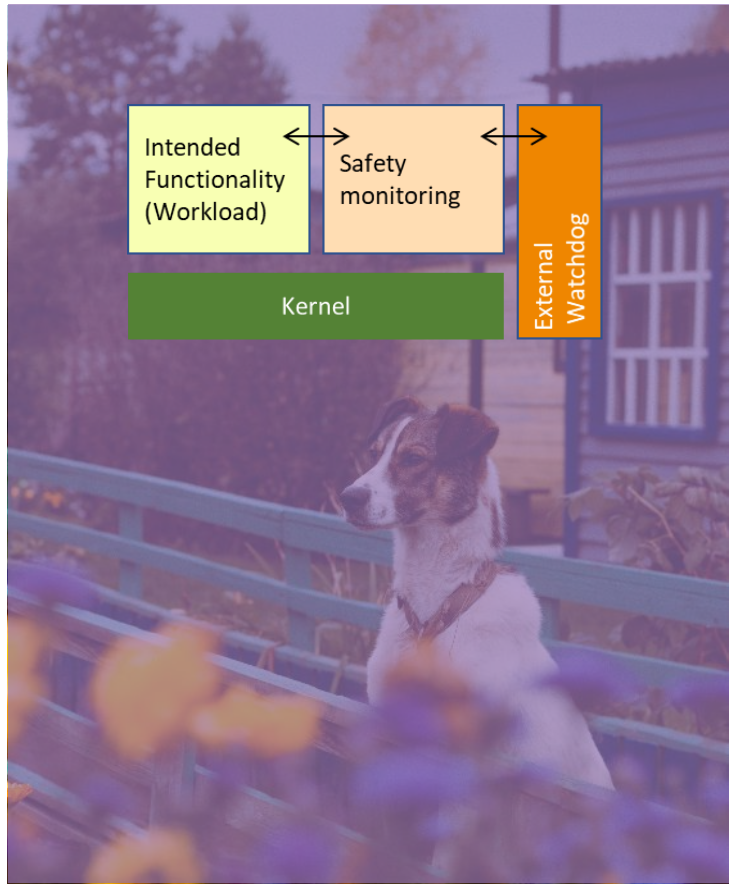


Photo by Marii Siia on Unsplash

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

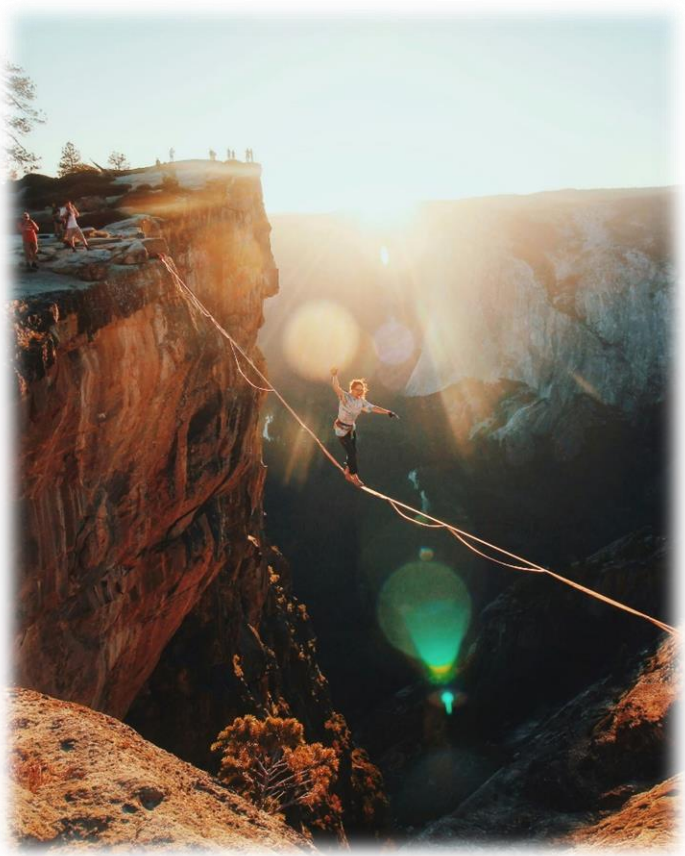


Photo by Jesse Bowser on Unsplash

?

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

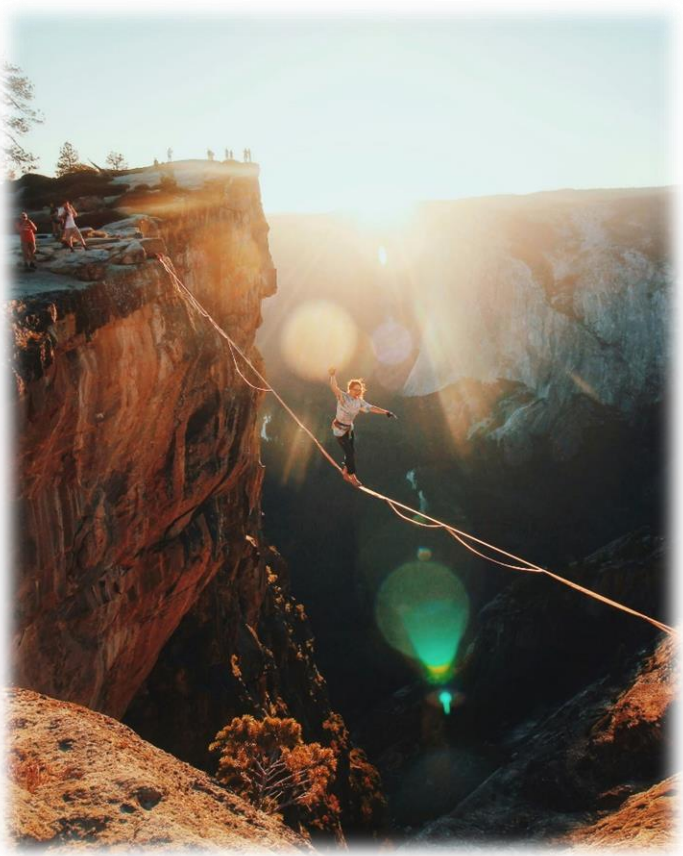


Photo by Jesse Bowser on Unsplash

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!

ELISA
Enabling Linux in Safety Applications

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

Project examples

Code

Project approaches

Various starting points exist…

– „Enable Safety afterwards“ → touched in previous slides

– Start from “Security footprint & Critical industries” project

– “Certification in mind” (from beginning)

– Control the development environment & OSS project (“Single vendor OSS”)

– Certification before going open source

→ Solution needs to be viable

Security footprint & Critical industries

Example: Xen Project

– Situation: Since Xen for embedded, security WG was started in parallel (in 2010)
  – Widely adopted in critical production environment (Data center, Desktop & Embedded)
  – Community brings quality awareness

– Implemented measures:
  – Rigorous Quality Process. Full commit traceability.
  – Security & isolation are project's top priority
  – Commits are tested with 2 CI loops.
  – Misra compliance

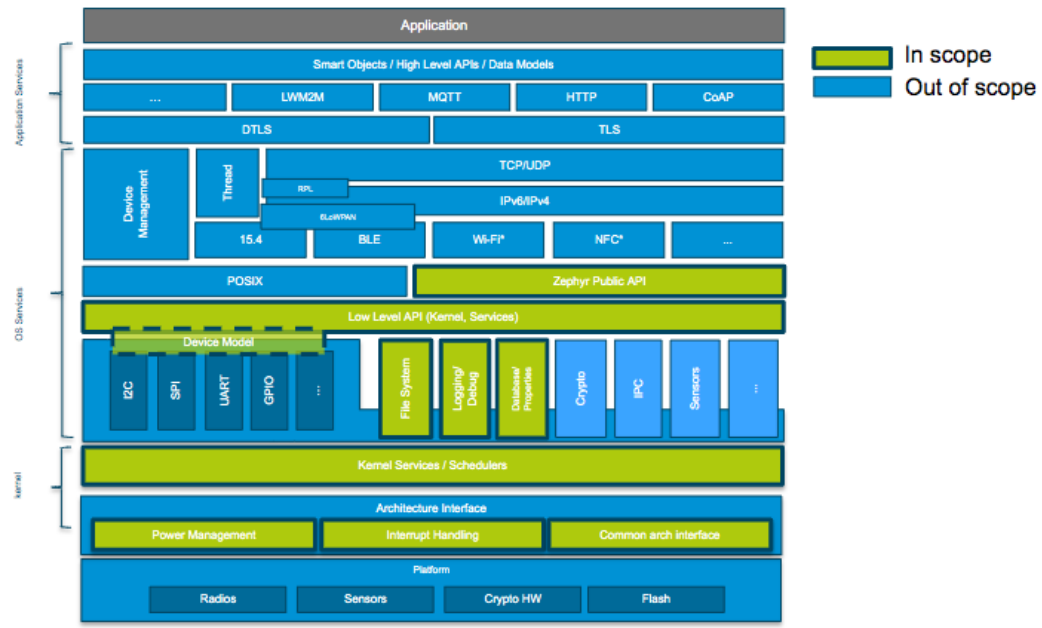
Certification in mind (wide use cases)

Example: Zephyr RTOS

– Situation: Targeting wide use cases beyond safety & security, but consider safety certification from the beginning

– Resulting challenge: heterogenous community with manifold non-safety-critical use cases

– Implemented measures:
tools and processes established early
  – Traceability & Requirements → StrictDoc
  – MISRA checks
  – Certification of code subsets.
  – IEC 61508 Route 3S



https://www.zephyrproject.org/introduction-of-coding-guidelines-for-zephyr-rtos/

Single vendor open source

Example: L4Re

- Situation:
  - Ownership by a single vendor enables full control about the software
- Benefits by projects:
  - Easy adoption of processes to new demands
- Resulting challenges:
  - Harder to create a community
    (Typically projects under foundations perform better)
  - Less sharing of development cost
  - Soft Lock-In
  - License change to proprietary license

Hypervisor: L4Re by Kernkonzept
https://l4re.org/overview.html

- The L4Re Hypervisor and L4Re Micro Hypervisor form the base for virtualization platform for hosting workloads of general-purpose, real-time, security and safety kinds.
- It consists of a small kernel, a microkernel, and a user-level infrastructure that includes basic services such as program loading and memory management up to virtual machine management.

Doing the “hard thing” before

Example: ThreadX (former Azure RTOS)

- Situation/Benefits:
  - Certified when going open source

- Risks/Challenges for the project:
  - Onboarding community
  - Add community contributions to the established process
  - Liability of an OSS foundation
  - Certification cost covering

- Measures taken:
  - Certification organization
  - Commercial model

RTOS: ThreadX at Eclipse (Microsoft)

https://threadx.io/

- This RTOS is designed for deeply embedded applications. It provides advanced scheduling facilities, message passing, interrupt management, and messaging services.

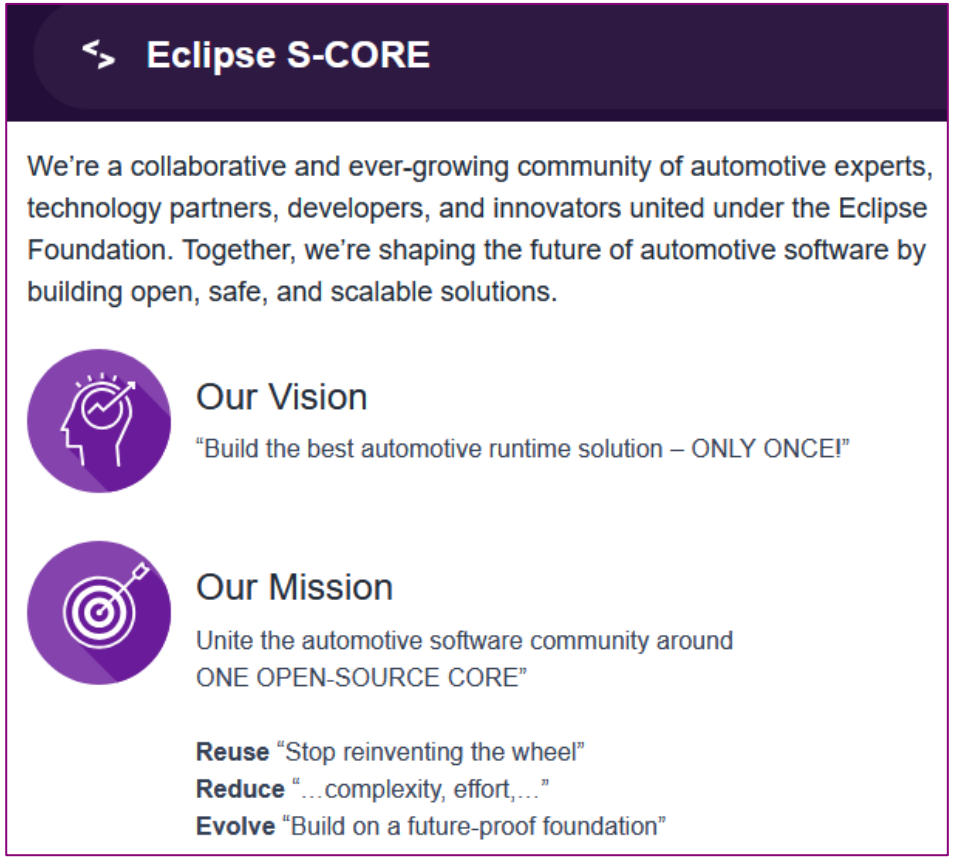
- Eclipse ThreadX RTOS has many advanced features, including picokernel architecture, preemption threshold, event chaining, and a rich set of system services.

Combining various approaches

Example: Eclipse Safe Open Vehicle Core (S-Core)

- Situation:
  - Security footprint & Critical industries *Xen*
    → Automotive focus
  - Certification in mind *Zepyhr*
    → Develop process & SW in parallel
  - Overcome single vendor OSS *L4Re*
    → Execute project in Eclipse Foundation
  - Doing the hard thing before *ThreadX*
    → Liability and certification cost transferred.

- Challenges:
  - Clash of Automotive & OSS world
  - Community focus on Automotive professionals
  - “Certification ready”, but no actual certification



Repeating theme: Not fully open everything*

Open Core – your business case

- Source code is typically available

- Limited view on tests, requirements, traceability…

- Safety artifacts often behind a paywall
  - By commercial company
  - Through membership fees

- Secure business with:
  - Commercial extensions and add-ons
  - Alternative & compatible implementations (safety & non-safety derivative)

Business: Liability/Certification, Add-Ons, SDK/IDE, Customer support & maintenance

**Ferroscene Certified Rust Compiler opens everything*

Concluding thoughts

Confusion

Safe <x>, Safety <x>, <x> for safety, SIL qualified…

Wording does not tell you what it is!

ELISA
Enabling Linux in Safety Applications

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

Photo by Nick Fewings on Unsplash

Last words or “The benefit of open“

Philipp

Can there be ”safety by obscurity“?

My very personal opinion

Can we put IP/patents (commercial interests) of proprietary software over safety?

Any proprietary development would have a chance to become “safer“ by providing the source code to the public (as e.g. expert outreach is increased).

“If you want to go fast, go alone;
if you want to go far, go together”

Photo by Kenny Eliason on Unsplash

Thank you

https://www.linkedin.com/in/philipp-ahmann/