HISC
HIGH INTEGRITY SOFTWARE CONFERENCE
OCT 22, 2024

Using Linux in Safety Critical Systems

Olivier Charrier – Wind River

v1.2

WNDRVR

Agenda

1 – Introduction

2 – Linux and DO-178C DAL-D

3 – Linux at DAL-C and above

4 – ELISA

Introduction

RTCA DO-178C / EUROCAE ED-12C

DAL	Failure Condition	Process Objectives	Code Coverage
Level A	**Catastrophic (may be total loss of life)**	**71**	**Level B + 100% of Conditions (MCDC)**
Level B	**Hazardous/Severe (may be some loss of life)**	**69**	**Level C + 100% of Decisions**
Level C	**Major (may be serious injuries)**	**62**	**Level D + 100% of Lines**
Level D	Minor (may be minor injuries)	26	100% of Requirements
Level E (5%)	No Effect (no impact on passenger or aircraft safety)	0	None

Operating System

- From https://en.wikipedia.org/wiki/Operating_system

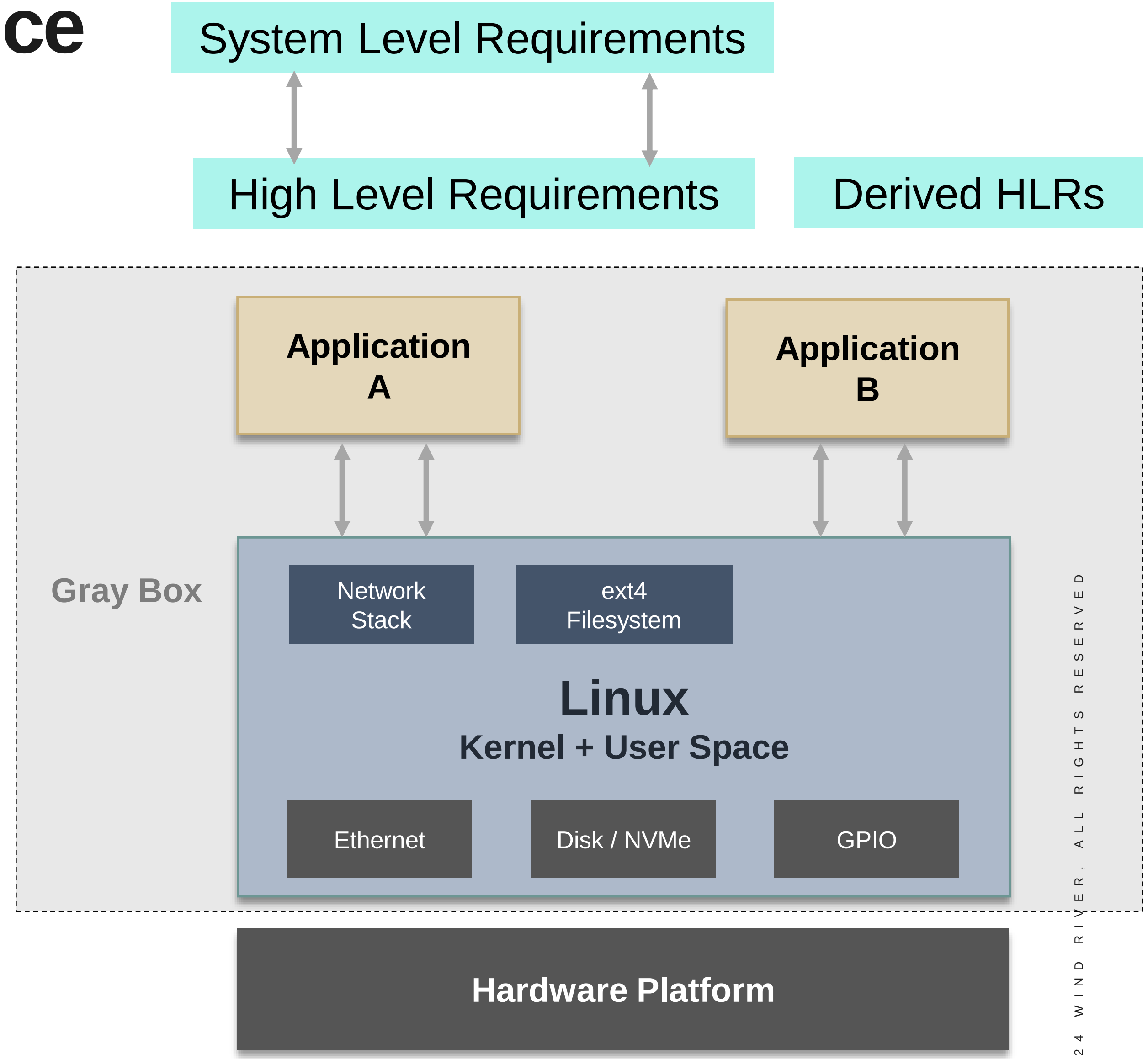
  "An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs."

- This is a toolbox, in the end, supporting a Safety Critical System Design with:
  - Language support (C-Lib, C++-Lib, etc.)
  - Multi-tasking Scheduling capabilities
  - Memory Management
  - Critical Section Management (data protection, semaphores, etc.)
  - Hardware abstraction layer
  - Middleware Services (Networking, File System, etc.)
  - Etc.
- There are typically 3 kinds of Operating Systems:
  - Roll-Your-Own (RYO)
  - Open Source Software (OSS)
  - Commercial-Off-The-Shelf (COTS) Operating Systems

Linux and DO-178C DAL-D

Approach-1: All Software at once

- Benefits
  - Minimize the exposure of Linux
  - No need to shrink Linux to a bare minimum
  - No need to detail all Linux capabilities in the Software Architecture
- Activities
  - Create HLDs
  - Master Integration Process (even if no source code is really required)
  - Scrutiny is at Functional Level (each and every HLR shall be tested)
- Assumptions
  - Take responsibility on the Linux component



System Level Requirements

High Level Requirements

Derived HLRs

Gray Box

Application A

Application B

Network Stack

ext4 Filesystem

Linux

Kernel + User Space

Ethernet

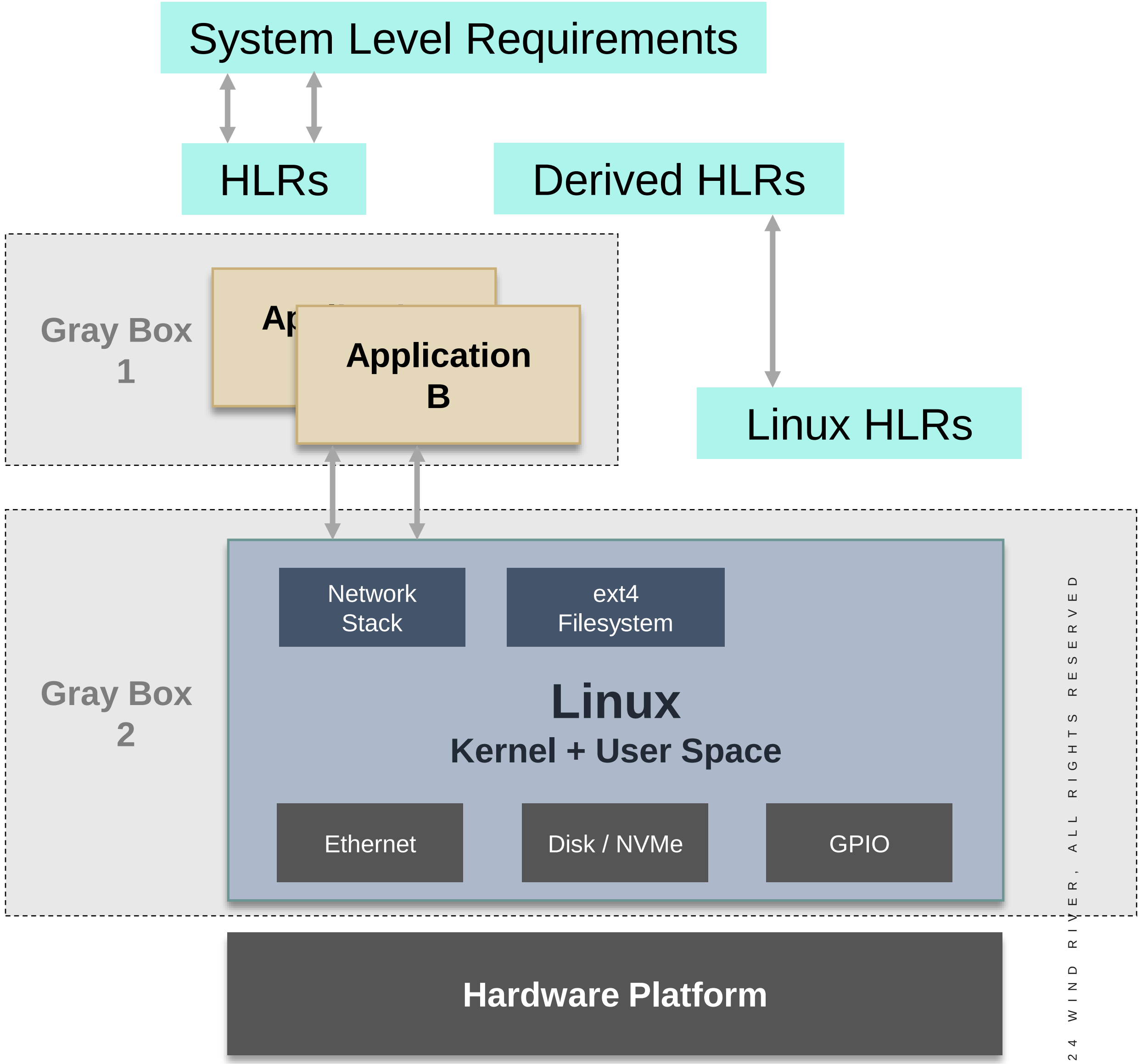
Disk / NVMe

GPIO

Hardware Platform

Approach-2: Split Model

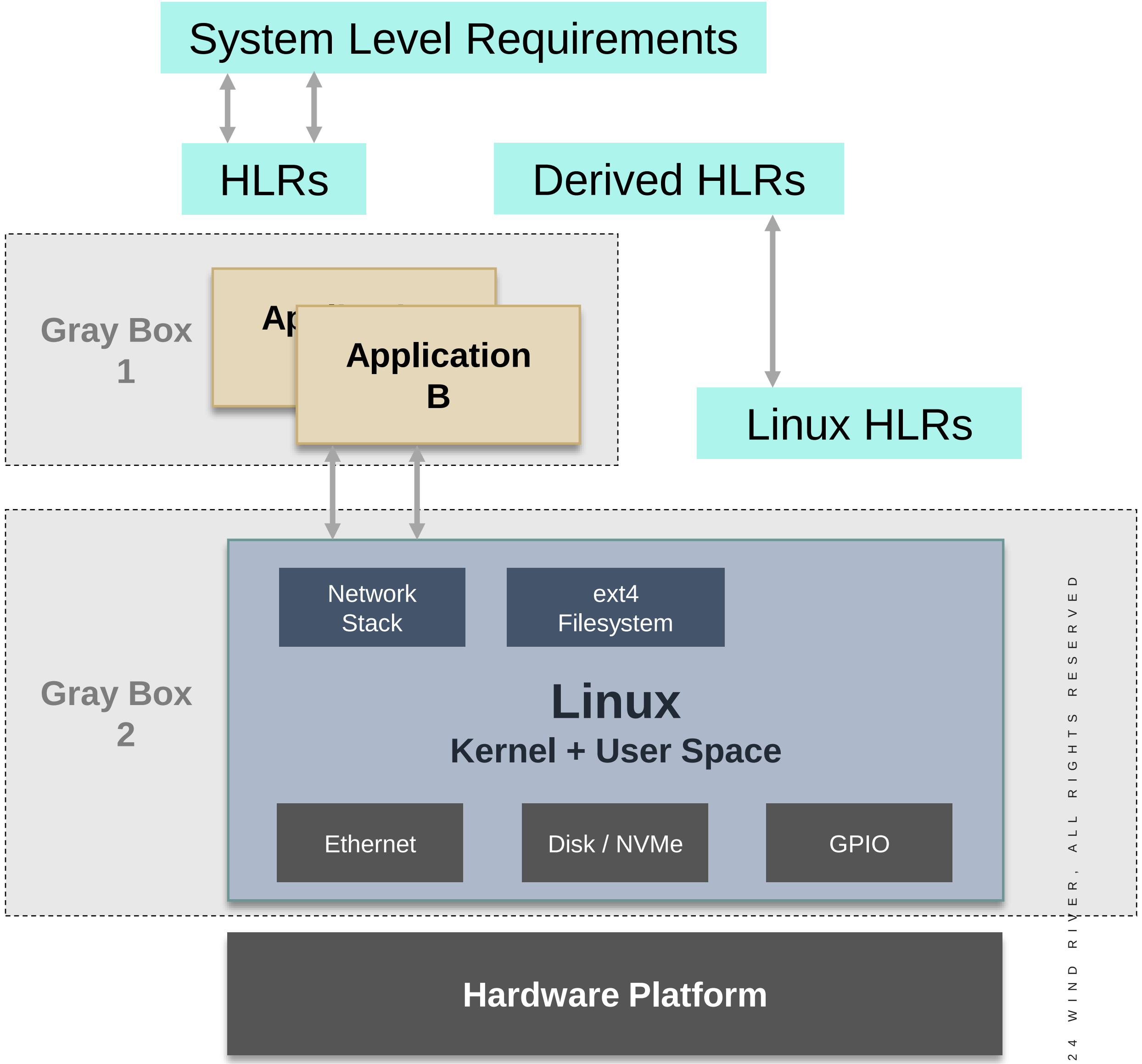
- To be implemented when you do not master Linux or get the Linux component from a board or silicon vendor
- Impact compared to Approach-1
  - 2x Certification Data Packs
  - Increase the exposure of Linux in term of HLRs and Software Architecture
  - Define a Software/Software integration layer
  - Assign responsibility of the Linux component to another stakeholder knowledgeable on Linux
  - Clarify where is the responsibility of the Hardware Support (usually driven by System Requirements)



WNDRVR

Approach-2: Split Model (cont.)

- Result
  - More work to be done
  - Create HLRs and Software Architecture out of Linux Man Pages and Source Code review?
  - Develop Linux specific HLRs tests



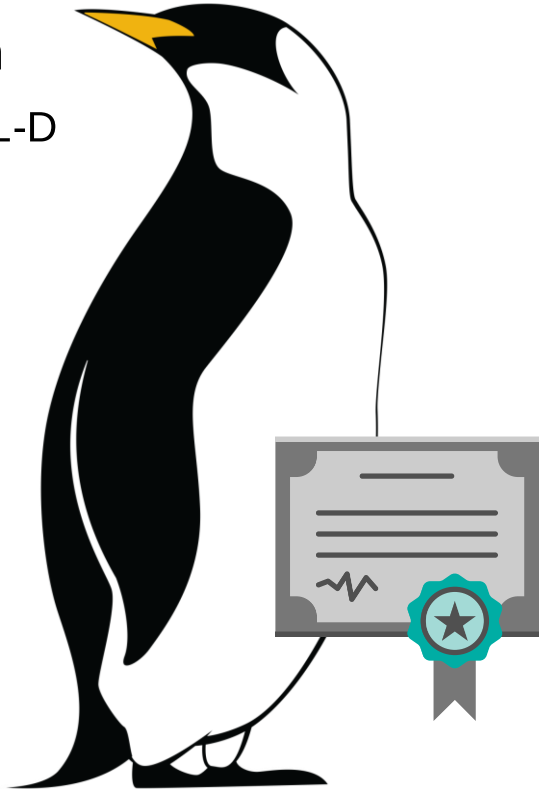
Linux at DAL-C and above

Switching to a full “White Box” approach

- Raising to DAL-C and above means full exposure of the Linux Operating System
  - Obviously, the level of work and so the price will seriously increase, compared to DAL-D
  - Less differences between Approach-1 and Approach-2 indicated for DAL-D

- Activities
  - Create HLDs/LLDs for the complete Linux OS
  - Provide a complete Design Documentation
  - Master the complete Development Process (source code is really required)
  - Scrutiny is moved to the level of any line of code (drives a slim profile definition)
  - Develop additional tests to reach 100% Statement Code Coverage
  - Etc.

- Assumption
  - Linux Kernel Contributor onboard
  - Good relationship with the Linux Kernel Community



Options?

- Options are usually driven by the final goal considering the associated budget
  - Reduce the scope of Linux to reduce the amount work to be done
  - Leverage the cost on multiple projects
  - Work at System Level, to mix different operating systems around a hypervisor (or not) for example.
  - Assumption: Knowledge on the Internals of Linux

- Reduce the content of Linux (removal of code, not just “deactivation”)
  - If reduced too much, it may not be re-usable for other projects, is it then worth the cost?
  - If not reduced enough, the level of work may not fit into the project budget

- Create new tools, processes, and leverage the cost on multiple projects
  - Define a Core Linux environment that can be used on multiple projects
  - Contribute with other companies to a bigger piece of work (methods, tooling, artifacts, etc.) that could be re-used on multiple projects.

- An example of resources information and contribution to a bigger piece of work: ELISA

ELISA

Enabling Linux in Safety Applications

ELISA Project Overview

Enabling Linux In Safety Applications (ELISA) project aims to make it easier for companies to build and certify Linux-based safety-critical applications – systems

ELISA - Where to start?

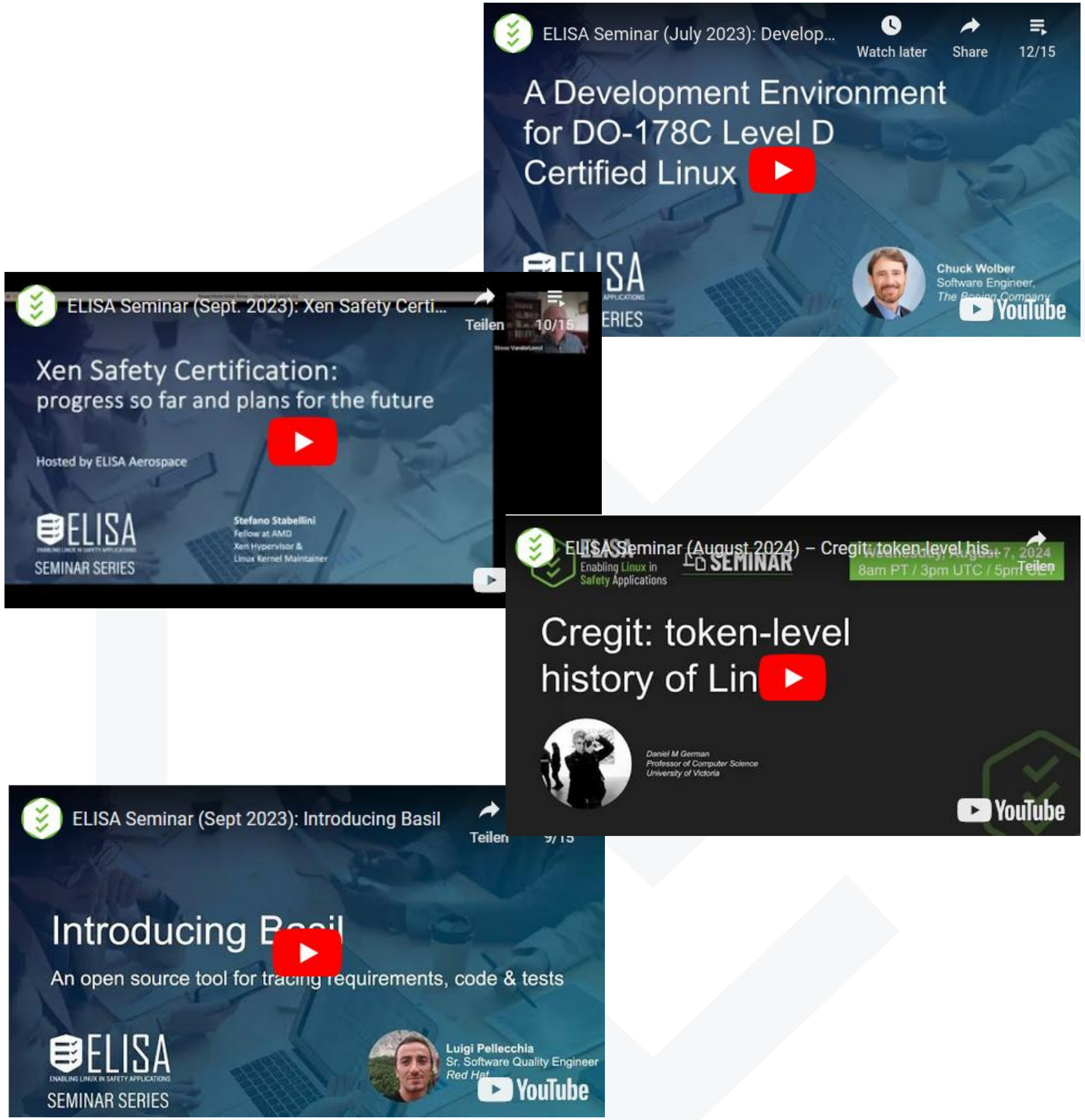
- Home Page of this Linux Foundation Project: https://elisa.tech/
  - Charter: https://elisa.tech/wp-content/uploads/sites/75/2020/08/elisa_technical_charter_082620.pdf
  - Members: https://elisa.tech/membership/members/
  - Q1 Newsletter: https://email.linuxfoundation.org/elisa-enabling-linux-in-safety-applications-q1-2024-newsletter
- Events – 2024 Update: https://www.youtube.com/playlist?list=PLuDNrzTpK8zouoi5lP3DbWKWO-dQgcz_f
- Events – Workshops, ELISA Face2Face meetings: https://elisa.tech/workshop-series/
- Events – Seminars, Subject matter presentations: https://elisa.tech/seminar-series/
- Resources – Case Studies: https://elisa.tech/case-studies/
- Resources – White Papers: https://elisa.tech/white-papers/

ELISA
Enabling Linux in
Safety Applications

ELISA Seminar Series

- https://elisa.tech/seminar-series/
- Training & Awareness
- Inside ELISA & outside

- Linux (PREEMPT_RT, page table, …)
- Safety process (SEooC, Automotive, Avionics, ...)
- Tools (BASIL, cregit, RTLA, …)
- Communities (Xen, stress-ng, KernelCI, …)



ELISA Seminar (July 2023): Develop...

A Development Environment for DO-178C Level D Certified Linux

Chuck Wolber
Software Engineer,
The Boeing Company

ELISA Seminar (Sept. 2023): Xen Safety Certi...

Xen Safety Certification: progress so far and plans for the future

Hosted by ELISA Aerospace

Stefano Stabellini
Fellow at AMD
Xen Hypervisor &
Linux Kernel Maintainer

ELISA Seminar (August 2024) – Cregit: token-level his...

Wednesday, August 7, 2024
8am PT / 3pm UTC / 5pm CET

Cregit: token-level history of Lin

Daniel M German
Professor of Computer Science
University of Victoria

ELISA Seminar (Sept 2023): Introducing Basil

Introducing Basil

An open source tool for tracing requirements, code & tests

Luigi Pellecchia
Sr. Software Quality Engineer
Red Hat

ELISA
Enabling Linux in
Safety Applications

ELISA Technical Side and Working Groups

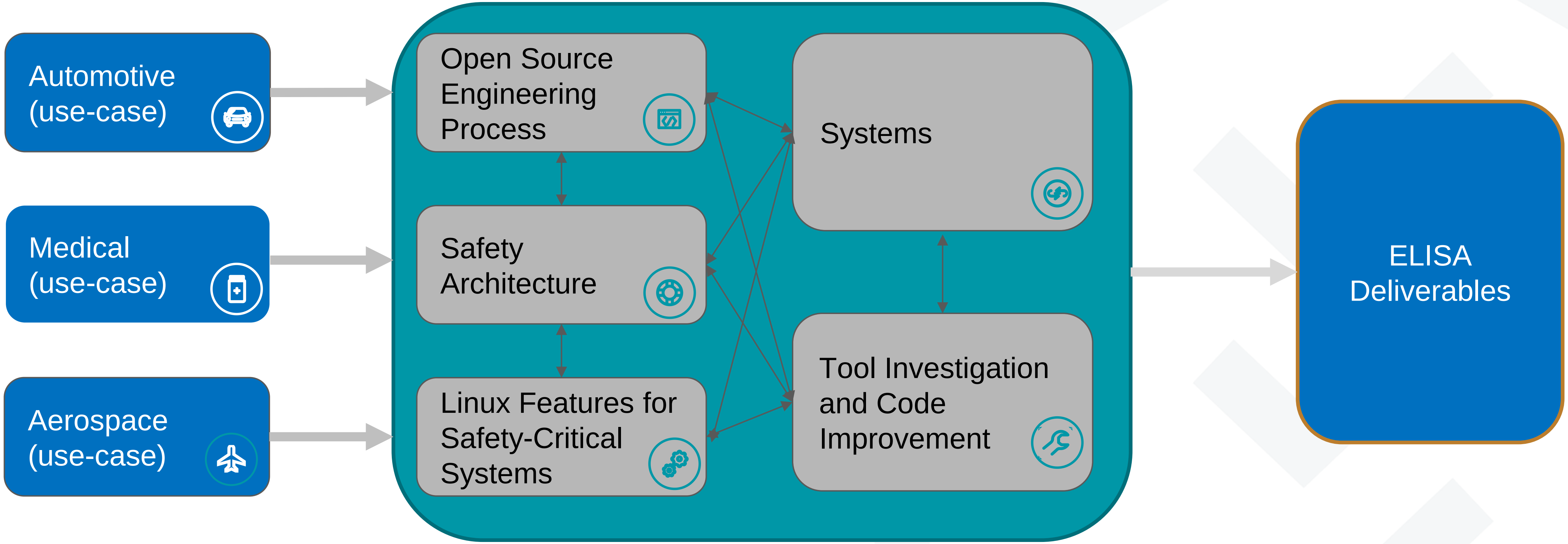
- Technical Forum: https://lists.elisa.tech/g/devel
- Community Google Drive: https://drive.google.com/open?id=1Y6Uwqt5VEDEZjpRe0CBCIibdtXPgDwlG
- GitHub: https://github.com/elisa-tech (contains minutes and presentations, etc.)
- Subgroups/Working Groups (WG) : https://lists.elisa.tech/g/main/subgroups

- Vertical Working Groups (provide use cases)
  - Aerospace WG: https://lists.elisa.tech/g/aerospace
    - Use case under definition, in particular a Space Grade Linux
  - Automotive WG: https://lists.elisa.tech/g/automotive
    - Telltale use case
  - Medical-Devices WG: https://lists.elisa.tech/g/medical-devices
    - Open Artificial Pancreas System (OpenAPS) Project use case

ELISA Working Groups (cont.)

- Horizontal Working Groups
    - Linux Features for Safety-Critical Systems (LFSCS) WG: https://lists.elisa.tech/g/linux-features
    - Open-Source Engineering Process (OSEP) WG : https://lists.elisa.tech/g/automotive
    - Safety-Architecture WG: https://lists.elisa.tech/g/safety-architecture
    - Systems WG: https://lists.elisa.tech/g/systems
    - Tool Investigation and Code Improvement WG: https://lists.elisa.tech/g/tool-investigation

ELISA Working Groups



Automotive (use-case)

Medical (use-case)

Aerospace (use-case)

Open Source Engineering Process

Safety Architecture

Linux Features for Safety-Critical Systems

Systems

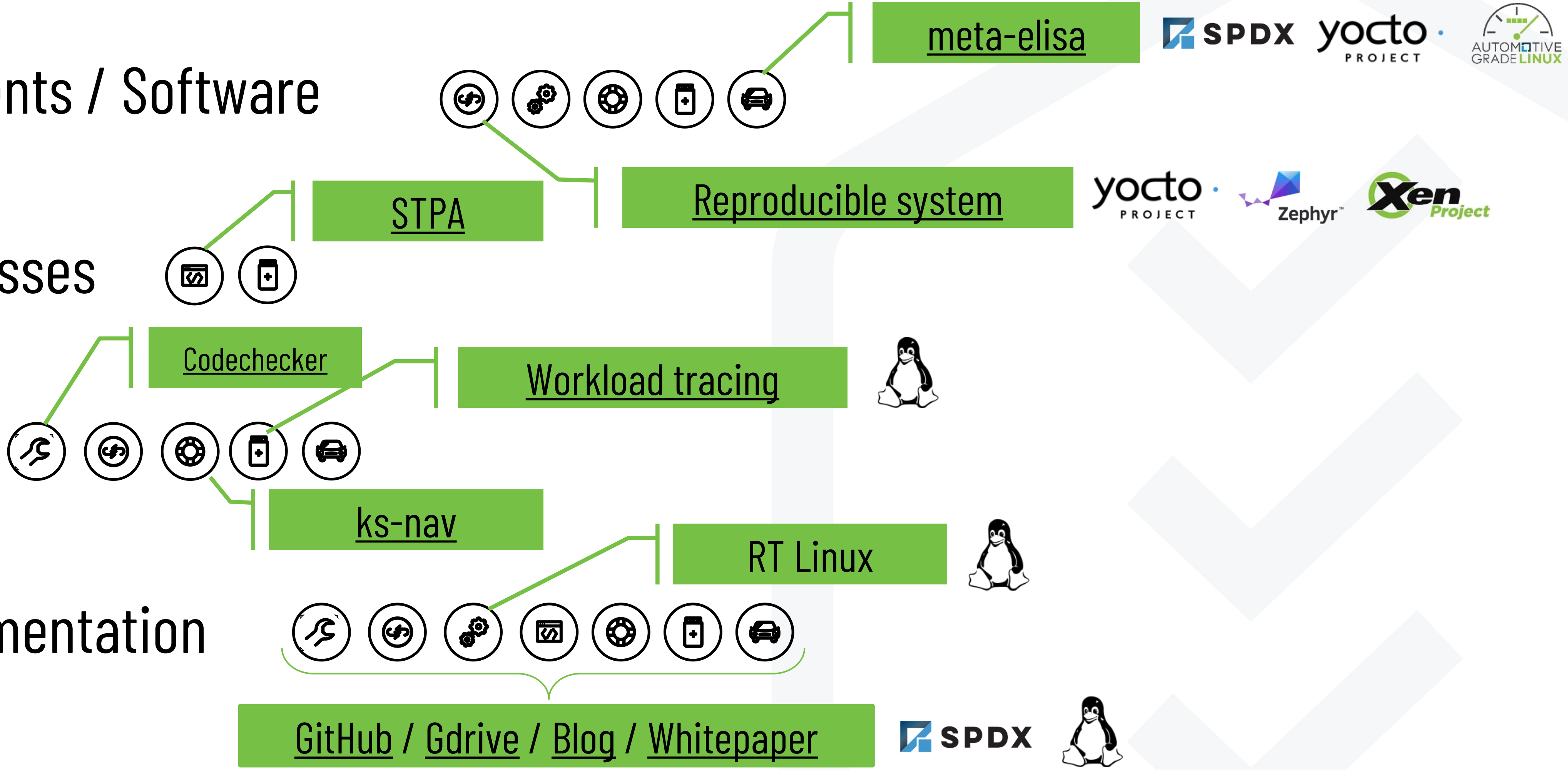
Tool Investigation and Code Improvement

ELISA Deliverables

ELISA
Enabling Linux in
Safety Applications

ELISA Working Groups - Deliverables

- Elements / Software
- Processes
- Tools
- Documentation



meta-elisa

Reproducible system

STPA

Codechecker

Workload tracing

ks-nav

RT Linux

GitHub / Gdrive / Blog / Whitepaper

Conclusion

- For DAL-C and above, there is a quite large amount of work to be achieved
- Collaborative work and involvement of the Linux community is key here to build an affordable solution
- Automation and Tooling is certainly required to cope with the amount of work
  - To create mandatory certification artifacts
  - To help with impact analysis to adopt Linux updates

Interested in Safety with Linux ?
Join ELISA and Contribute!

WNDRVR