東京 2025

LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025

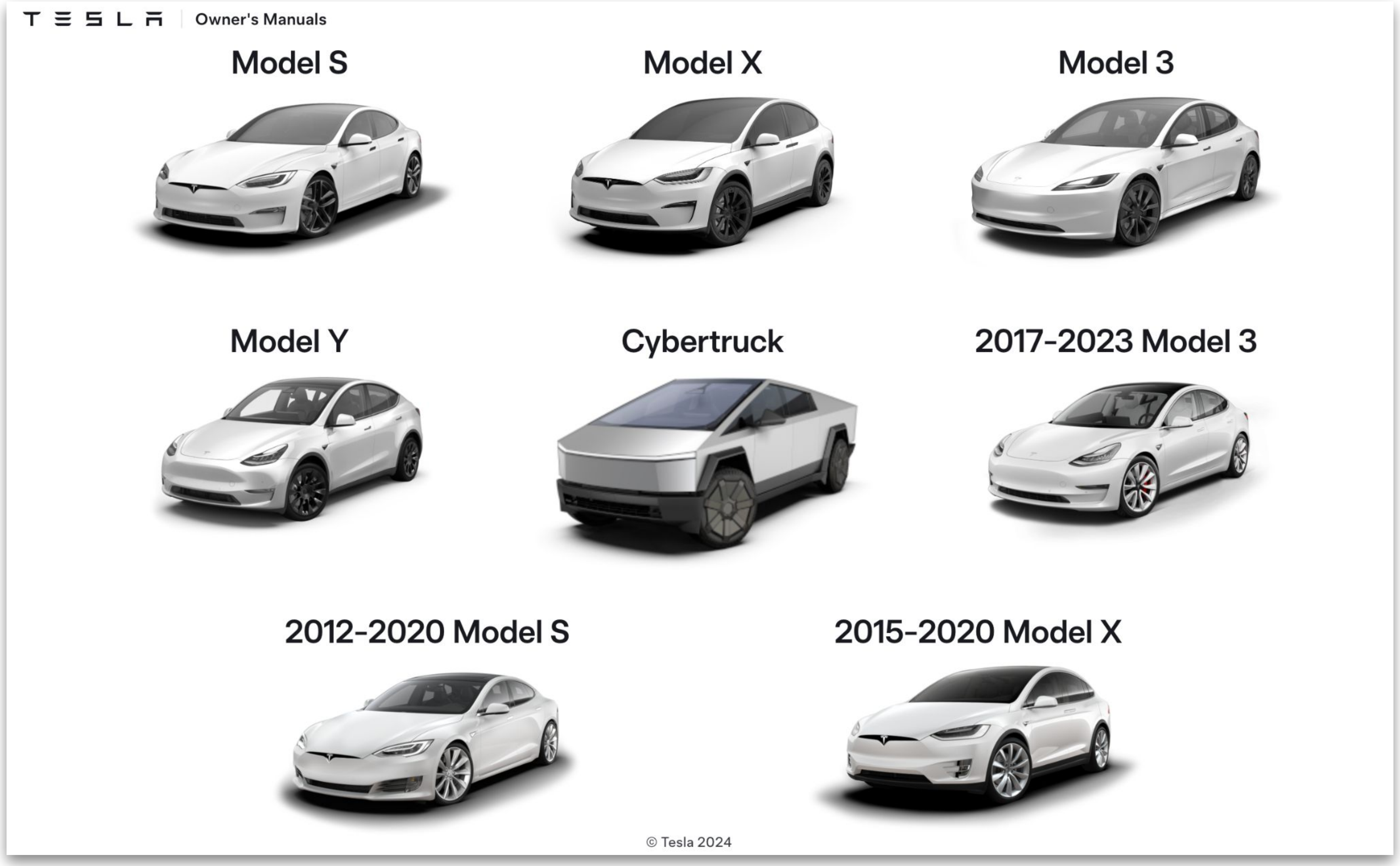
Aspects of Dependable Linux Systems

Kate Stewart, Linux Foundation
Philipp Ahmann, Etas GmbH (BOSCH)

Safe Systems with Linux MC

東京 2025
LINUX
PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

Linux is being used in Safety Critical Systems today...



TESLA | Owner's Manuals

Model S

Model X

Model 3

Model Y

Cybertruck

2017-2023 Model 3

2012-2020 Model S

2015-2020 Model X

© Tesla 2024

source: https://www.tesla.com/ownersmanual

source: https://www.spacex.com/mission/

東京 2025
LINUX
PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

ELISA
Enabling Linux in
Safety Applications

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.

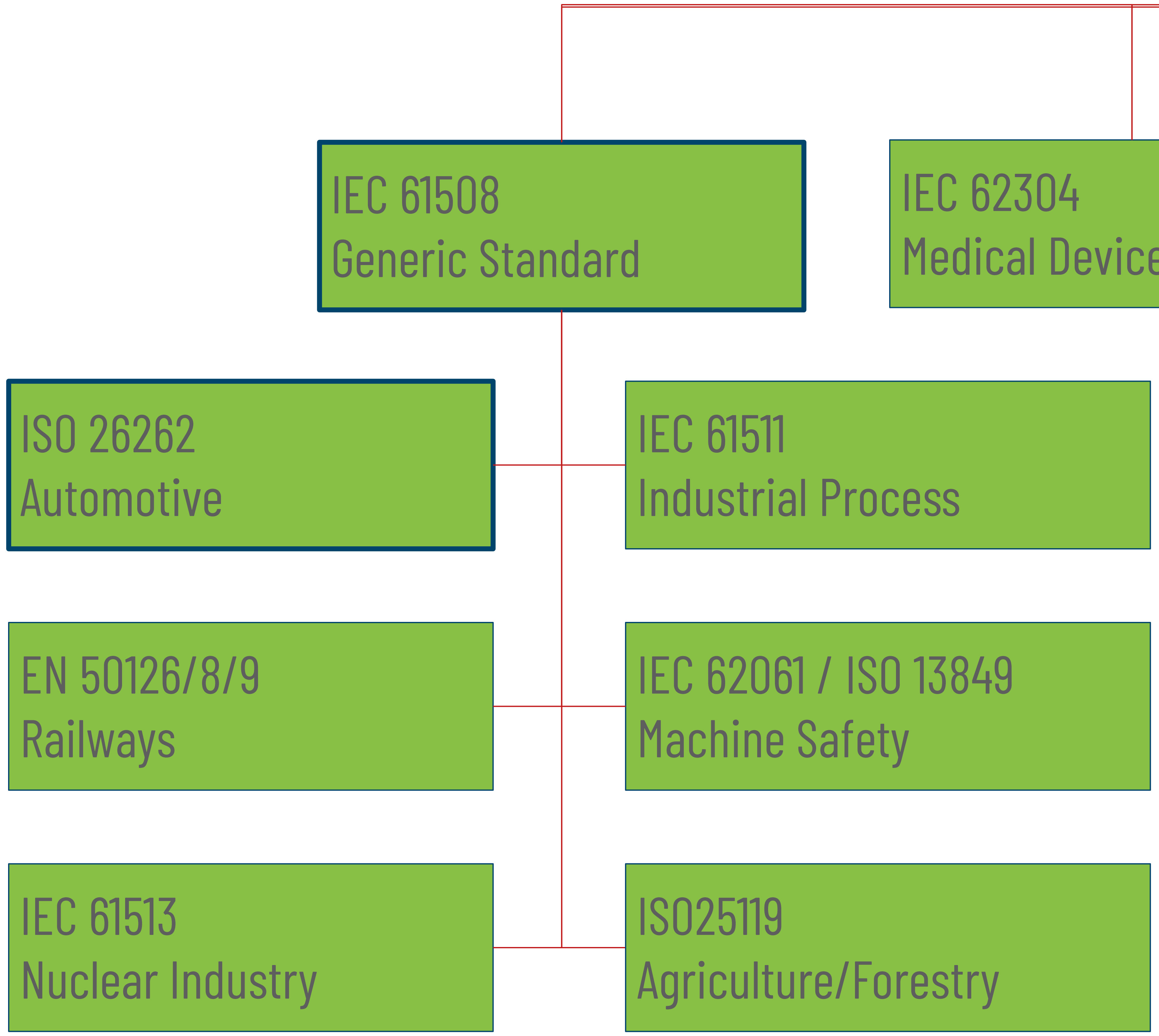
In Functional Safety you expect…

…that the software:

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

and you have sufficient evidence to prove this.

Samples of safety (integrity) standards



IEC 61508
Generic Standard

IEC 62304
Medical Devices

DO178B/C
Aeronautics

ECSS Space
(ESA)

ISO 26262
Automotive

IEC 61511
Industrial Process

EN 50126/8/9
Railways

IEC 62061 / ISO 13849
Machine Safety

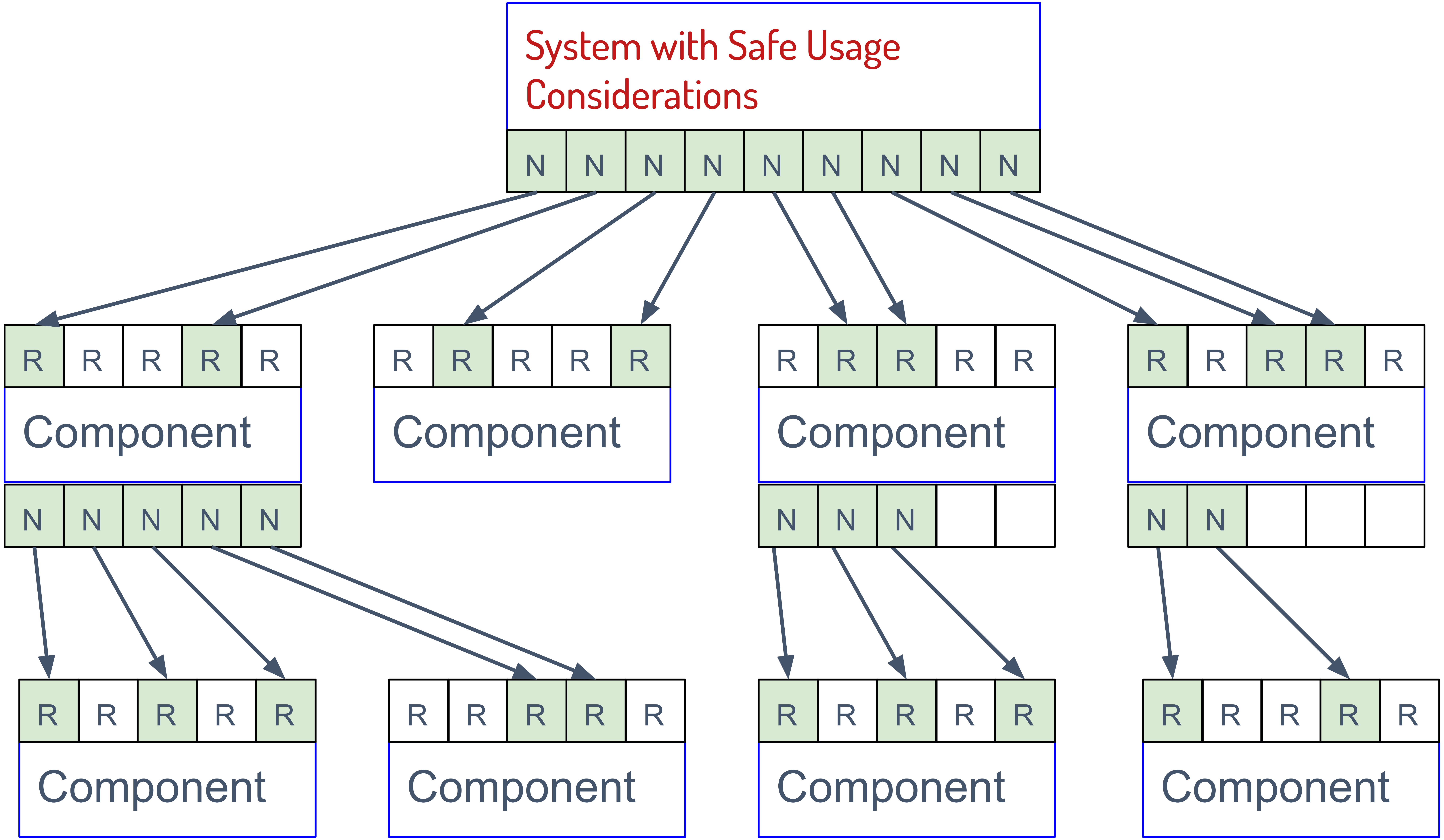
IEC 61513
Nuclear Industry

ISO25119
Agriculture/Forestry

- Standards represent industry **best practices developed over 50+ years**
- Share similar demands for requirements and evidence
- Rigor on measures/demands may differ
- **All system parts need to be known, tested and managed**

Standards seek to increase system quality

- Requirements
- Testing
- Documentation
- Traceability



System with Safe Usage Considerations

N N N N N N N N N

R R R R R
Component
N N N N N

R R R R R
Component

R R R R R
Component
N N N

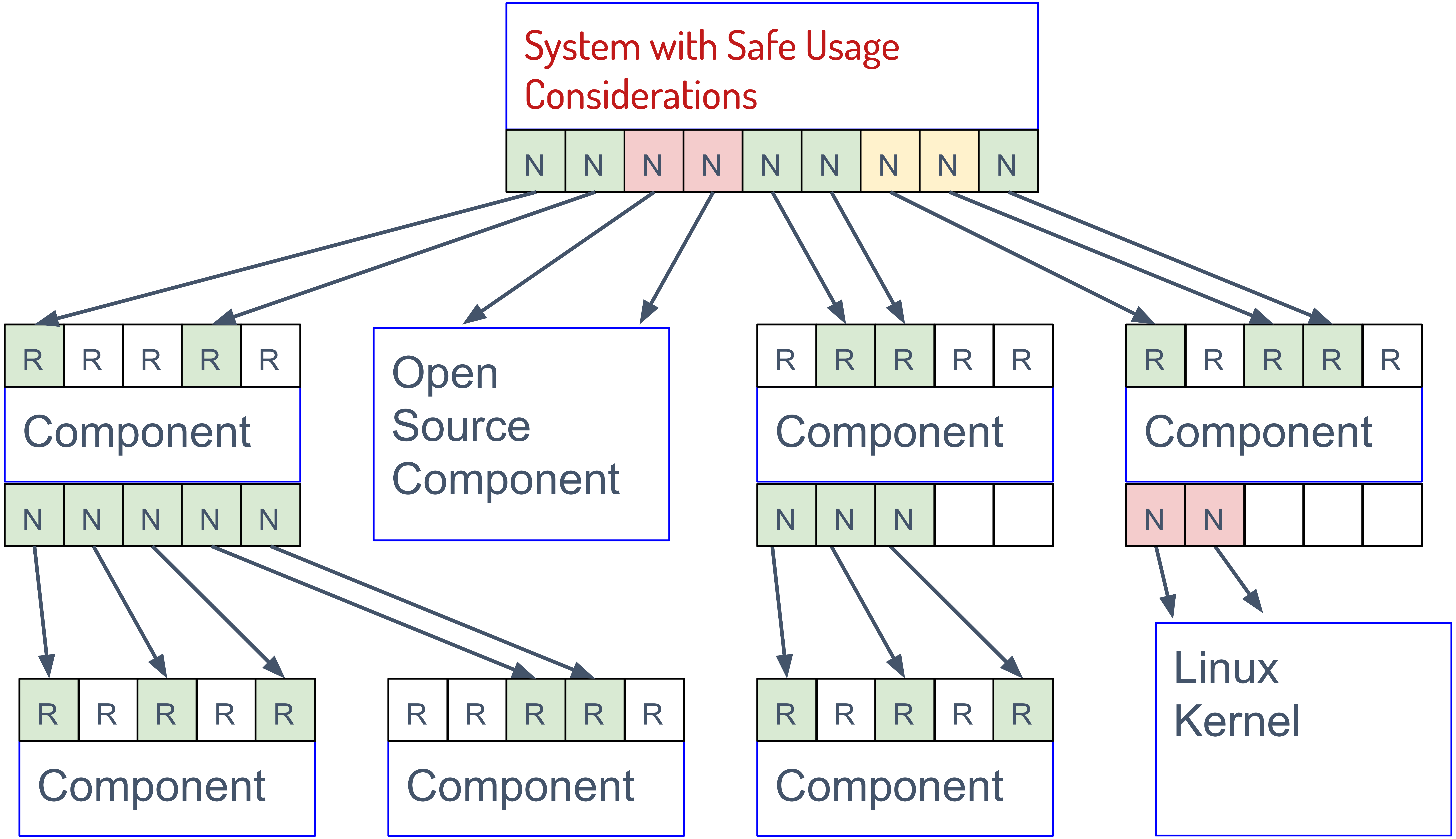
R R R R R
Component
N N

R R R R R
Component

R R R R R
Component

R R R R R
Component

R R R R R
Component



System with Safe Usage Considerations

N N N N N N N N N

R R R R R

Component

N N N N N

Open Source Component

R R R R R

Component

N N N

R R R R R

Component

N N

R R R R R

Component

R R R R R

Component

R R R R R

Component

Linux Kernel

Challenge with Safe Usage of Linux Kernel

Each patch has a reason for being added to tree - "what" & "why"

- Frequently contained in patch series overview, but may be part of email discussion.
- Understanding "what" the code should do, is considered as a "**requirement**" on a component (like the kernel) when doing functional safety system analysis.
- Testing the functionality for when it **works**, and when it **does not work** is needed as "evidence" that is required to assess "Safe Usage".

Challenge: The Linux Kernel has no way of systematically capturing "what" the code is expected to do in a machine readable form.

If the "assertions about the code" (may be referred to as specifications or requirements) are reverse engineered by others, where should they be stored, so they can be reviewed by maintainers and other experts?

What mechanisms should be used to link the code & tests to these requirements?

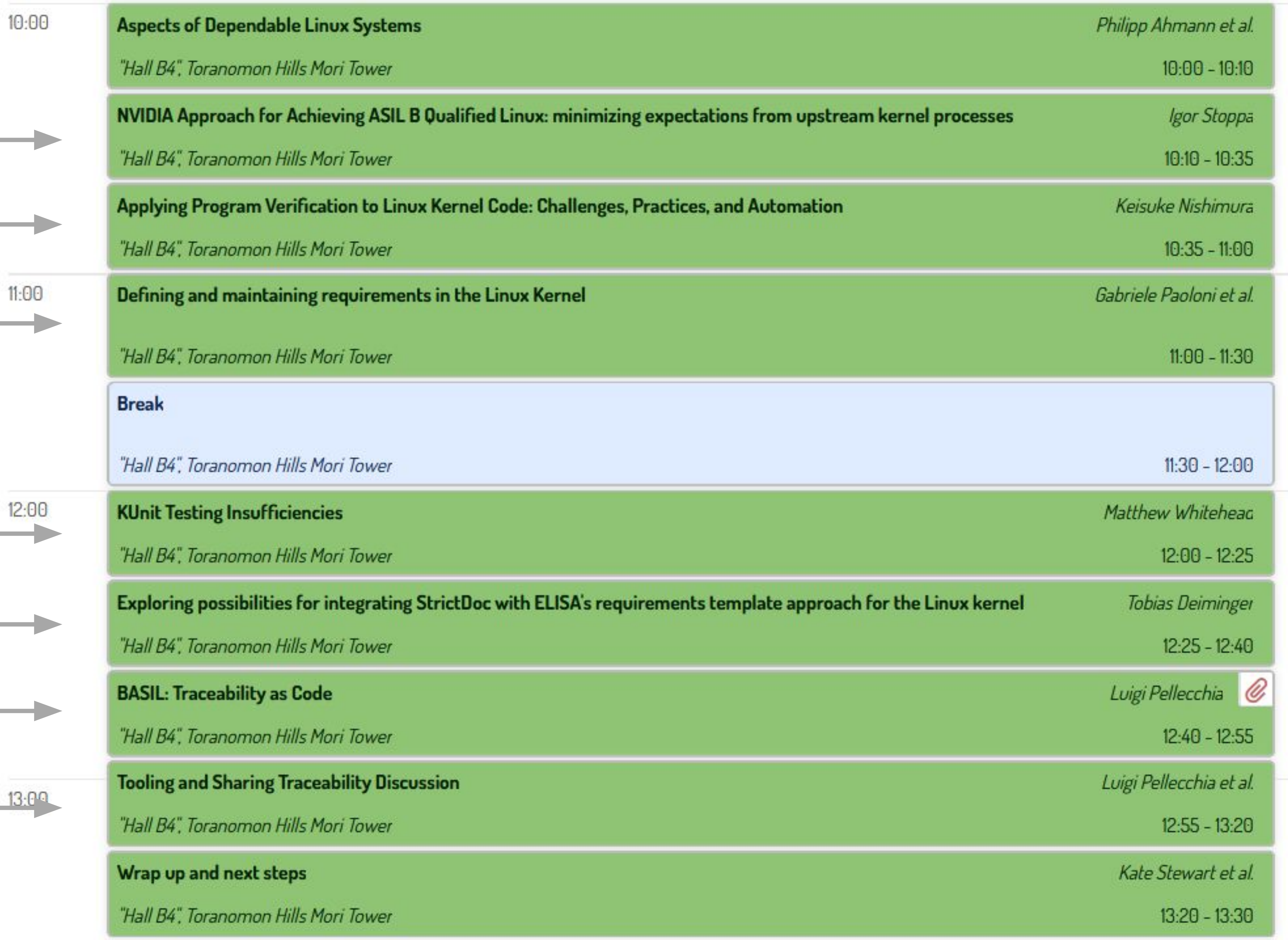
A morning towards „Safe Systems with Linux“

Addressing code

- Practical Qualification efforts
- Program Verification
- Requirements definition

& tools

- KUnit Test improvements
- StrictDoc next to Kernel documentation
- Kernel Requirements tool
- Tooling & Traceability discussions



10:00

Aspects of Dependable Linux Systems *Philipp Ahmann et al.*
"Hall B4", Toranomon Hills Mori Tower 10:00 - 10:10

NVIDIA Approach for Achieving ASIL B Qualified Linux: minimizing expectations from upstream kernel processes *Igor Stoppa*
"Hall B4", Toranomon Hills Mori Tower 10:10 - 10:35

Applying Program Verification to Linux Kernel Code: Challenges, Practices, and Automation *Keisuke Nishimura*
"Hall B4", Toranomon Hills Mori Tower 10:35 - 11:00

11:00

Defining and maintaining requirements in the Linux Kernel *Gabriele Paoloni et al.*
"Hall B4", Toranomon Hills Mori Tower 11:00 - 11:30

Break
"Hall B4", Toranomon Hills Mori Tower 11:30 - 12:00

12:00

KUnit Testing Insufficiencies *Matthew Whitehea*
"Hall B4", Toranomon Hills Mori Tower 12:00 - 12:25

Exploring possibilities for integrating StrictDoc with ELISA's requirements template approach for the Linux kernel *Tobias Deiminge*
"Hall B4", Toranomon Hills Mori Tower 12:25 - 12:40

BASIL: Traceability as Code *Luigi Pellecchia*
"Hall B4", Toranomon Hills Mori Tower 12:40 - 12:55

13:00

Tooling and Sharing Traceability Discussion *Luigi Pellecchia et al.*
"Hall B4", Toranomon Hills Mori Tower 12:55 - 13:20

Wrap up and next steps *Kate Stewart et al.*
"Hall B4", Toranomon Hills Mori Tower 13:20 - 13:30

東京 2025 LINUX PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

ELISA
Enabling Linux in Safety Applications

Safety Critical Systems

“Assessing whether a system is safe,

requires understanding the system sufficiently.”

- Understand your system element within that system context

  and how it is used in that system.
- Select system components and features that can be evaluated for safety.
- Identify gaps that exist where more work is needed

  to evaluate safety sufficiently.

東京 2025

LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025