



# Aspects of Dependable Linux Systems

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

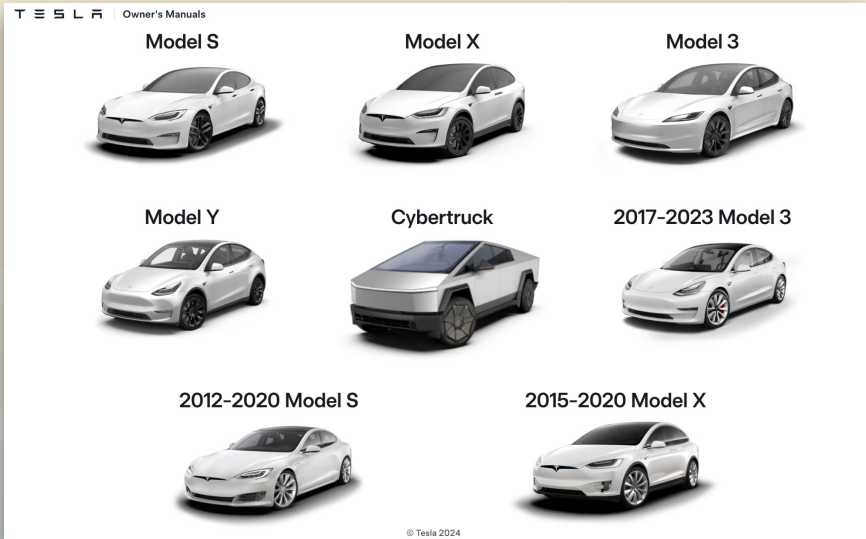
Safe Systems with Linux MC



LINUX  
PLUMBERS  
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

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



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



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



**LINUX  
PLUMBERS CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026



# What is Functional Safety?

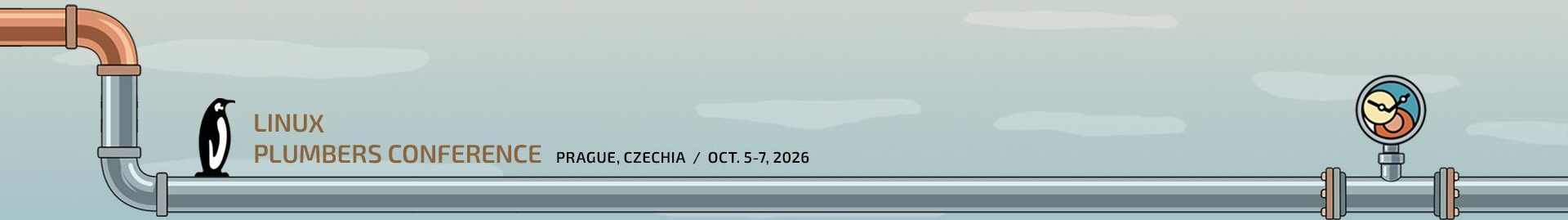
## Definition of Safety

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.

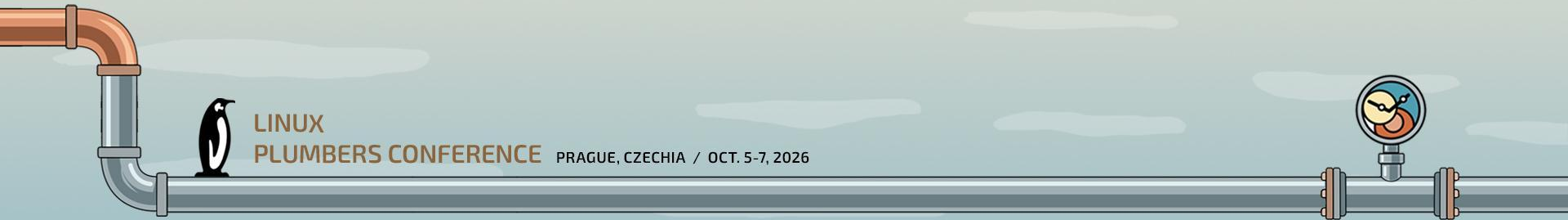


# In Functional Safety you expect...

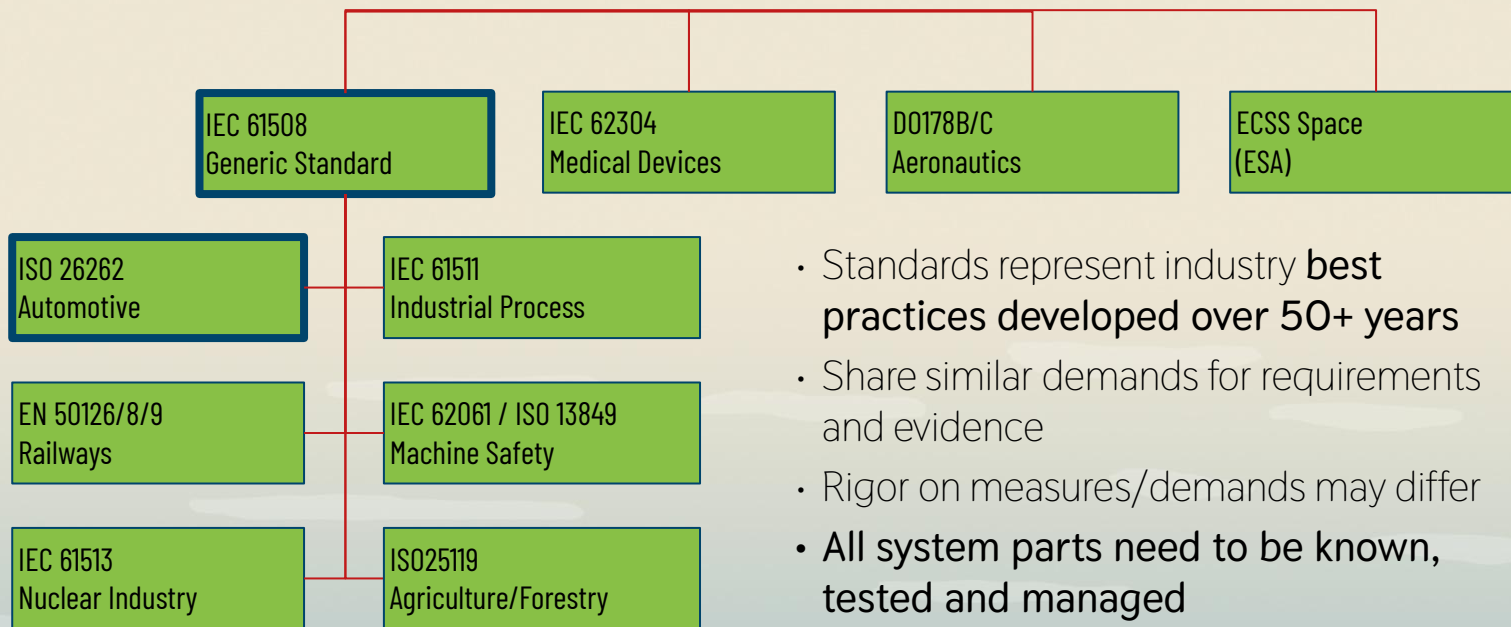
...that the software:

- 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.



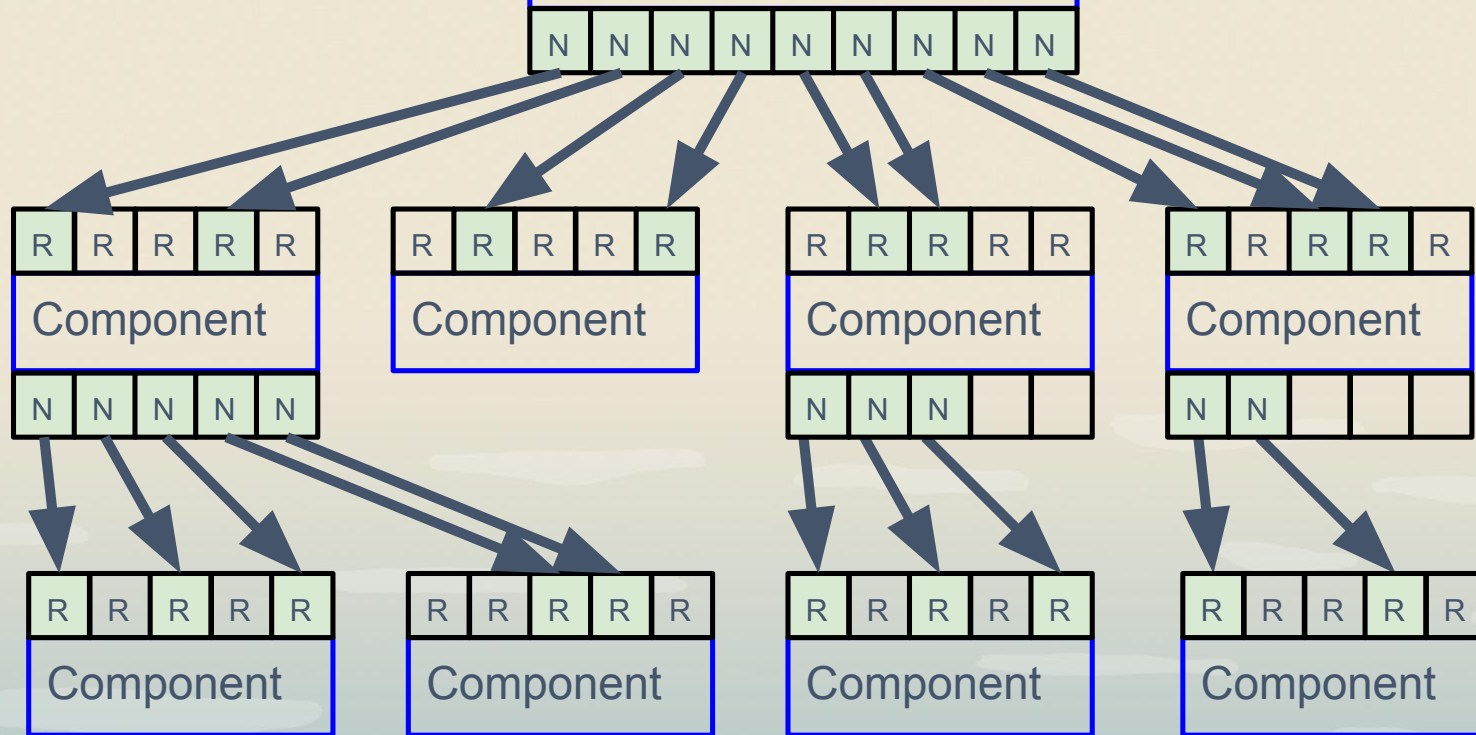
# Samples of safety (integrity) standards



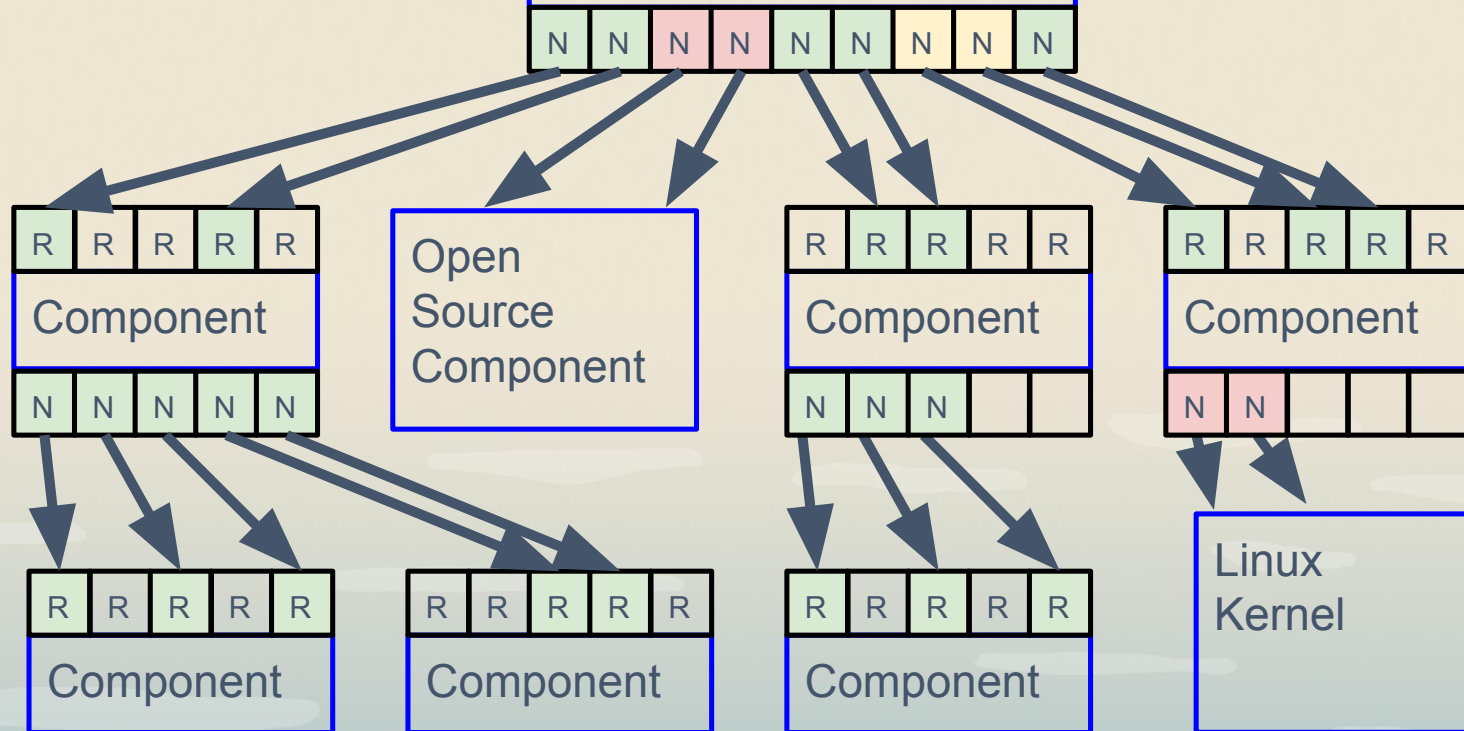
# Standards seek to increase system quality

- Requirements
- Testing
- Documentation
- Traceability

## System with Safe Usage Considerations



## System with Safe Usage Considerations



# 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?



**LINUX  
PLUMBERS CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026



# A morning towards „Safe Systems with Linux“

## Addressing code

- Program Verification
- Practical Qualification efforts  
(common pitfalls & errors)

## And things around it (like tools)

- Testable expectations
- Linux Kernel Requirements
- Safety SBOM for Linux

|       |                                                                                                                                     |                                                 |
|-------|-------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| 10:00 | <b>Aspects of Dependable Linux Systems</b><br><i>"Club A", Prague Congress Centre</i>                                               | <i>Philipp Ahmann et al.</i><br>10:00 - 10:10   |
|       | <b>Towards Program Verification of the Linux Kernel Library XArray</b><br><i>"Club A", Prague Congress Centre</i>                   | <i>Corinn Tiffany</i><br>10:15 - 10:40          |
| 10:40 | <b>Common pitfalls and errors, when using Linux in Safety Applications</b><br><i>"Club A", Prague Congress Centre</i>               | <i>Igor Stoppa</i><br>10:45 - 11:30             |
|       | <b>Coffee Break</b><br><i>"Club A", Prague Congress Centre</i>                                                                      | 11:30 - 12:00                                   |
| 12:00 | <b>Scaling the Design Definition</b><br><i>"Club A", Prague Congress Centre</i>                                                     | <i>Chuck Wolber</i><br>12:00 - 12:25            |
|       | <b>Defining Linux Kernel Requirements and Test Specifications out of the Kernel Tree</b><br><i>"Club A", Prague Congress Centre</i> | <i>Gabriele Paoloni et al.</i><br>12:30 - 12:55 |
| 12:55 | <b>From Manual Argument to Machine-Readable Graph: Toward a Safety SBOM for Linux</b><br><i>"Club A", Prague Congress Centre</i>    | <i>Nicole Pappler</i><br>13:00 - 13:25          |
|       | <b>Wrap up</b><br><i>"Club A", Prague Congress Centre</i>                                                                           | <i>Philipp Ahmann et al.</i><br>13:25 - 13:30   |



**LINUX  
PLUMBERS CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026

# Safety Critical Systems

*“Assessing whether a system is safe,  
requires understanding the system sufficiently.”*

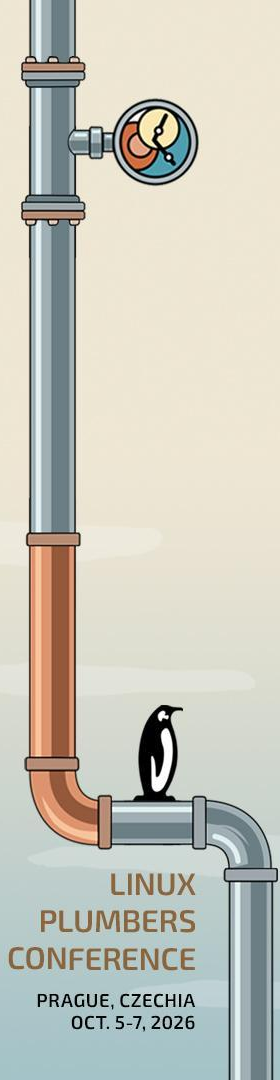
- Understand your system element within that system context and how it is used in that system.
- Select system components and features that can be evaluated for safety.
- Identify gaps that exist where more work is needed to evaluate safety sufficiently.



**LINUX  
PLUMBERS CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026





**LINUX  
PLUMBERS  
CONFERENCE**

PRAGUE, CZECHIA  
OCT. 5-7, 2026