ELISA
Enabling Linux in Safety Applications

WORKSHOP

VOLVO

ELISA Workshop
Lund, Sweden

Safe Continuous Deployment (CD) in the Automotive Industry

May 7-9, 2025
Co-hosted with Volvo Cars

Dr. Håkan Sivencrona,
Senior Technical Leader
Volvo Cars


ELISA
Enabling Linux in
Safety Applications
WORKSHOP
VOLVO
Safe Continuous Deployment (CD) in the Automotive Industry (With some Linux considerations)
Dr. Håkan Sivencrona, Senior Technical Leader – DevOps, Volvo Cars
MGM 11U

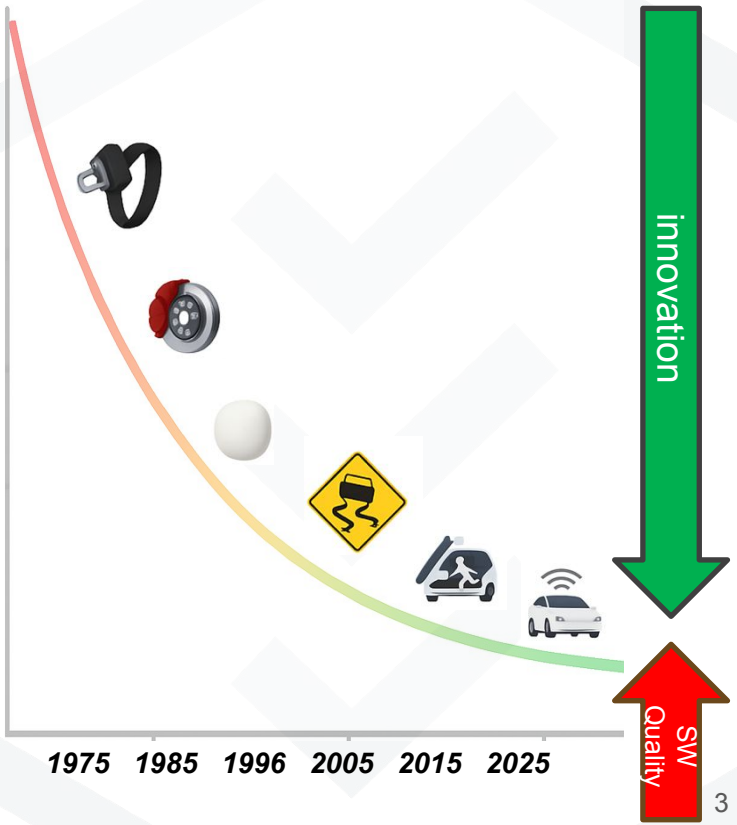
From Seatbelts to Software Control: Why Automotive Safety Needs Rigor from OSS

- In 1975, saving a life in a car crash could be adding a seatbelt. In 1995, it meant deploying an airbag in milliseconds. **Today, in 2025, saving a life might mean identifying a child on the road, using a neural network running on a real-time Linux-based system.**

- The world of automotive safety has changed dramatically. Hardware innovations gave us decades of progress. **But we've reached the point where the next life saved won't come from stronger steel or more airbags. It will come from code - from the quality of the software that makes split-second decisions in advanced driver-assistance systems (ADAS) and the software-defined vehicle (SDVs).**

- For those of us in the Linux/Elisa community, i.e., open-source world, this **is our moment of responsibility**. The systems we build and contribute to**, must be deterministic, cyber secure, fault-tolerant, sustainable and maintainable over years.** A segmentation fault or memory leak is no longer a nuisance—it's a matter of how many lives we can save.

- Let's explore how software quality, integrity, and openness can continue the legacy of automotive safety—and why the **Linux/Elisa community plays a central role in the mission for a safe Continuous Deployment**



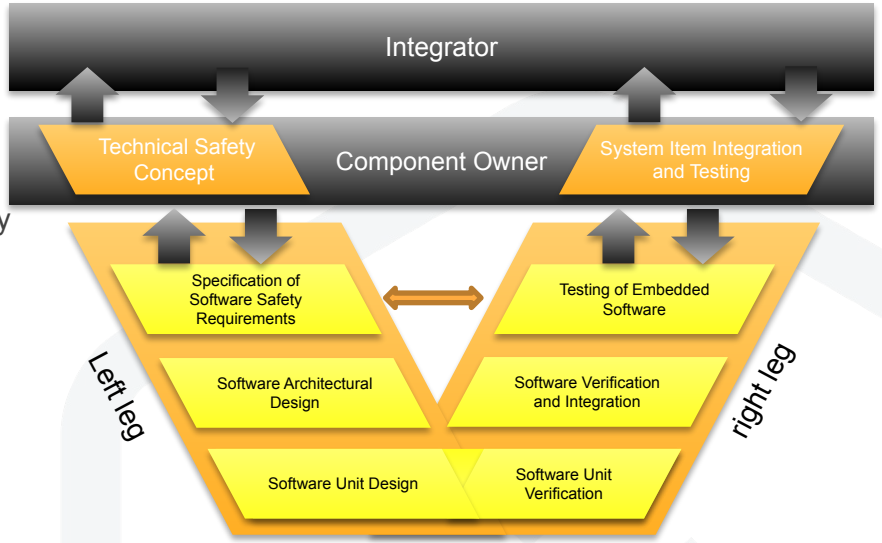
ELISA Enabling Linux in Safety Applications

WORKSHOP

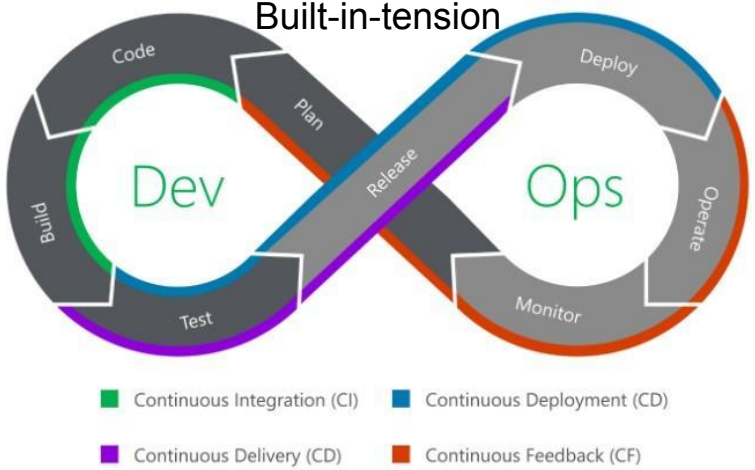
The Automotive Shift

Built in tension between vehicle development culture that needs linearity and determinism, driving, the project towards its milestones, e.g. A-, B-, C-samples and the more organic way of doing complex software development in a POSIX environment

- Traditional V-cycle vs Agile & Software-Defined Vehicles, SDV
- **(Updated) Safety Management Systems – SMS (QMS)**
- Connected cars enable over-the-air (OTA) updates
- Efficient use of Data from Fleet /Data Driven Development – Accurate Fleet data needed
- Going from “SOP projects”, i.e., one release to the “Always releasable”



Built-in-tension

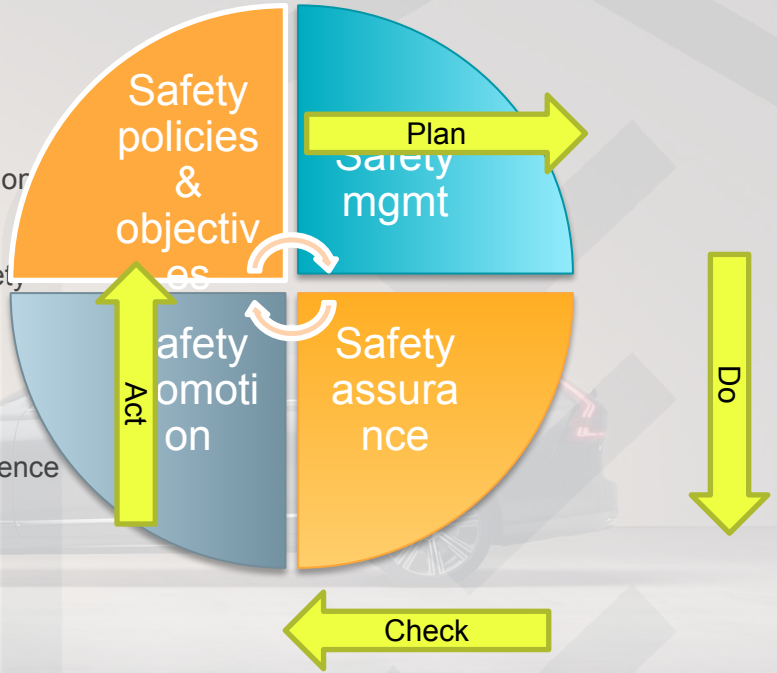


ELISA Enabling Linux in Safety Applications

WORKSHOP

Stakeholders Require - A Safety Management System - SMS

- Ensure that the organization is performing Safety activities properly and not sloppy:
  - Adequate organization-specific rules and processes for functior safety CI/CD
  - Processes to ensure an adequate resolution of identified safet anomalies CI/CD
  - Safety culture that supports and encourages the effective achievement of functional safety
  - Competence management system to ensure that the competence of the involved persons is commensurate with their responsibilities
  - A quality management system that supports functional safety
  - **A strategy for the safety case provision**



Nice!

Linux brings massive benefits…

Modularity and Customization

- Enabling a highly modular system design. Enabling developers to include only necessary components, reducing attack surfaces and improving safety assurance.

Open-Source Transparency

- Full access to the source code. Transparency helps verifying the code against safety standards like ISO 26262, ensuring that known vulnerabilities can be identified and mitigated.

Strong Community and Vendor Support

- Linux benefits from a massive global developer base and commercial support (e.g., Red Hat, Wind River). Communities contribute timely security patches, safety enhancements, and validated software components tailored for automotive needs.

Real-Time Capabilities

- Linux can be partially adapted for real-time performance required in safety-related automotive systems, e.g. ADAS (PREEMPT_RT)

Certification Support and Safety Profiles

- ELISA ☺ - Helps manufacturers adopt Linux while maintaining compliance with generic safety requirements.

But…

Several core properties could pose challenges when used in continuous deployment (CD) for safety-related applications -> regression

Lack of Determinism in Mainline Linux – Hard real-time (contracts)

- Process scheduling, interrupt handling, and latency can vary —impacting real-time safety requirements, i.e. keeping contracts

Complex and Rapidly Changing Kernel

- The Linux kernel evolves quickly, with many contributors and patches. Keeping a version with a Safety case up to is hard.

Weak Isolation Between Applications

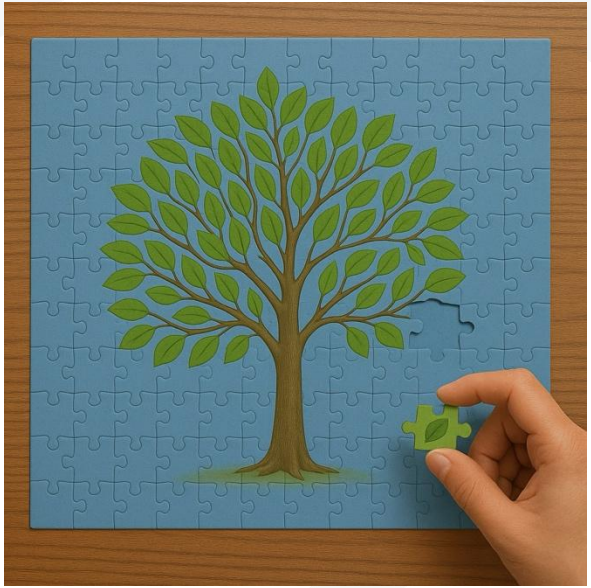
- Traditionally, Linux doesn’t enforce strict partitioning between applications unless you set up containers or hypervisors.

Update Mechanisms Aren’t Designed for Fail-Safe OTA

- A failed update must never jeopardize the driving. CD in automotive thus includes partitioned SW architectures

Last year – SEooC – Safety Element out of Context

- Software or hardware developed independently of a specific system
- Supports the component based argumentation – > safety case fragments
  - Assume & Guarantee for safety requirements fullfillment ahead if system design
- Integration of a SEooC in the system showing that assumed requirements are meeting the required expectations – for every release



So, next This year

This Year – Continuous Deployment Considerations

- Safe Continuous Software Updates
  - Integrating agile software practices like CI/CD while maintaining safety compliance (e.g., ISO 26262) is costly, complex and slow
- Complex Supply Chains & Component (SEooC) Integration
  - Coordinating software and hardware components from many vendors, each with their own tools and processes, slows development and validation
- Scalability of Validation & Testing
  - The explosion of new features, software variants and configurations makes exhaustive testing across all vehicle models unmanageable
- **Functional Safety in AI and ML, Regulatory Pressure and Evolving Standards**

So, next CI/CD chain for safe deployment

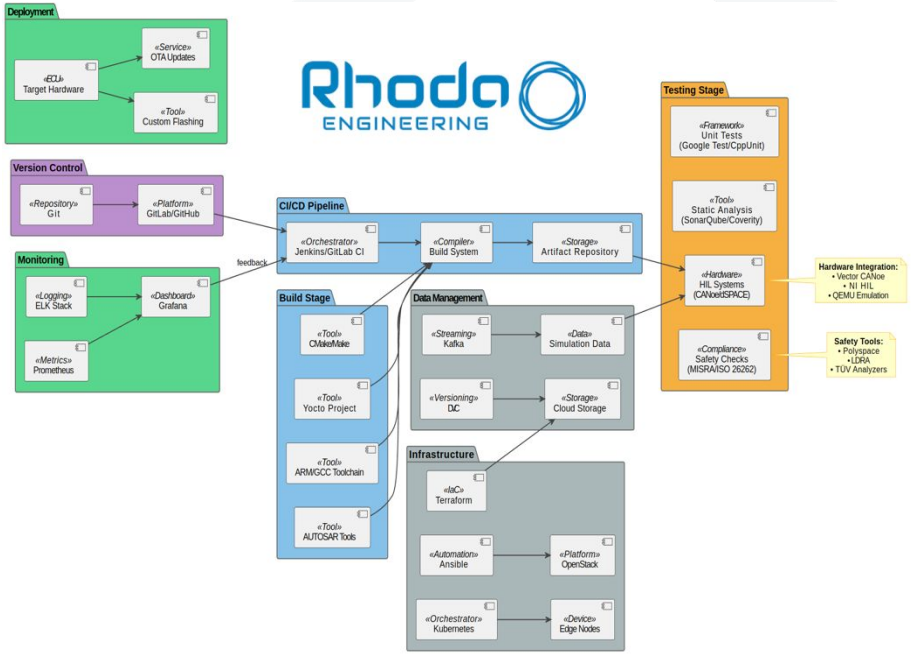
“If your car gets smarter every week, the update better be safe(r) every time”

“Software Now Runs the Vehicle…”

- CI/CD in Cars ≠ CI/CD in the Cloud or App world
- No “rollbacks at scale” when someone’s on the highway. No chaos engineering ☺
- Latency, determinism, and fault tolerance are non-negotiable.
- OTA (over-the-air) updates must be safe and securely transferred.

What “Good” Looks Like in Automotive CD…

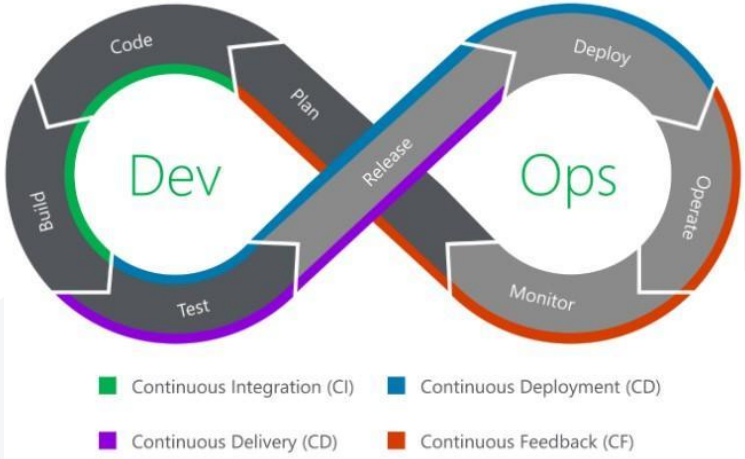
- End-to-end testing in real-world scenarios (e.g. simulated driving environments, virtual testing, shadow mode, fleet insight).
- Secure pipelines with cryptographic integrity from build to boot.
- Reproducible builds (reliable)
- Atomic, reversible updates without bricking the vehicle.



So, next The Safety DevOps

The Safety DevOps

- DevOps set of practices combines software development (Dev) and IT operations (Ops) to shorten the development lifecycle while delivering high-quality/safety software compliance continuously.
- Enhance collaboration between development and operations teams to streamline a safe software delivery process including the fleet and specific vehicles
- **Central to DevOps is the automation of repetitive tasks**, such as code testing, integration, and deployment, to improve efficiency, improve reproducibility => **reduced errors**
- DevOps emphasizes on frequent code integration and automated deployments (CI/CD) to ensure rapid and reliable delivery
- DevOps promotes a safety culture where teams work together, share responsibilities, and focus on a common goal - delivering safe software faster
- Supports the SMS



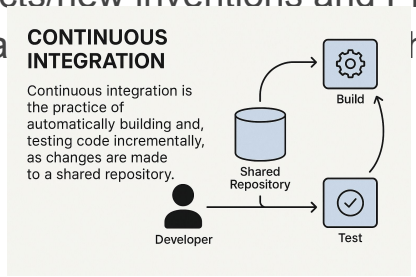
ELISA Enabling Linux in Safety Applications

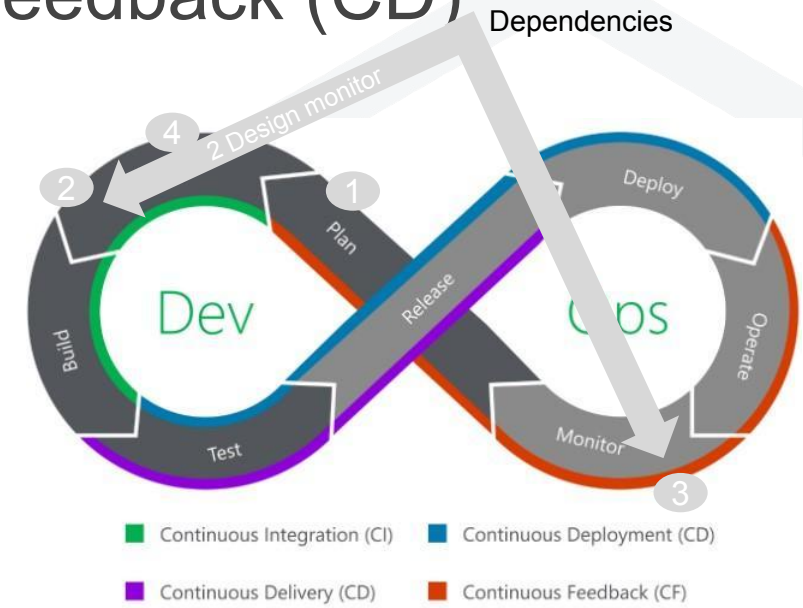
WORKSHOP

Continuous Deployment & Feedback (CD)

- What is is..
- An “Automated” pipeline from code to production
- "Zero” human intervention after code is merged
- Speeds up feedback loops and deployment cycles

- But introduces a challenge to go from advanced projects/new inventions and Proof of Concepts to “Alwa hicle





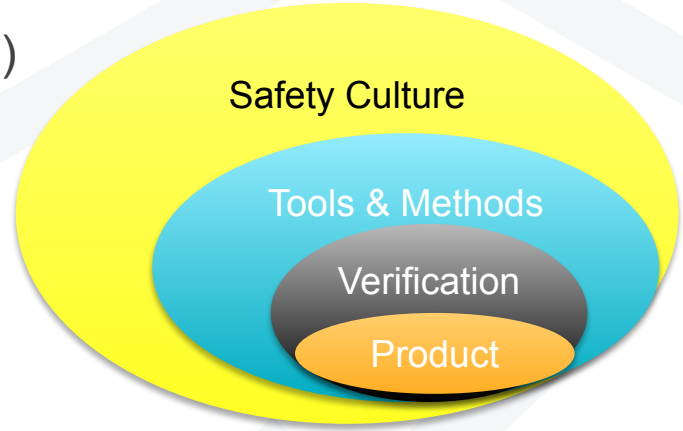
1 – Plan for Ops data as evidence in future assurance cases
2 – Design monitors
3 – Collect data from monitors
4 – Use collected Ops data as evidence in assurance case

ELISA
Enabling Linux in Safety Applications

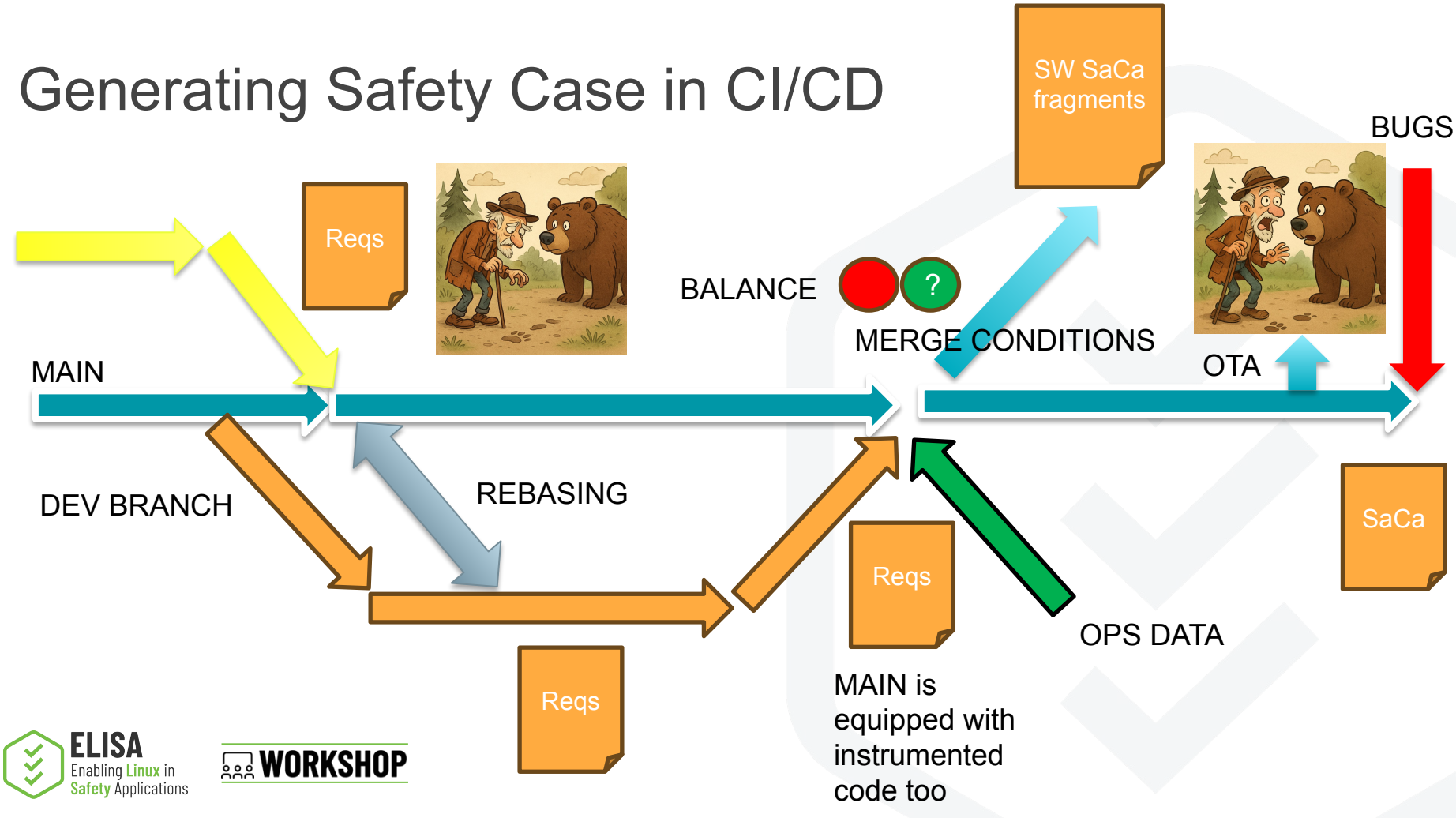
WORKSHOP

Challenges for a Safe CD

- End-to-end traceability (knowing dependencies)
- Testing and validation in real-time (Fleet)
- Architectural changes
- Component (sensor) replacement
- Rollbacks and fail-safes
- Balancing compliance with rapid updates
- Providing valuable and correct documentation, e.g. Safety assurance
- Dependencies between components and their different states, e.g. monitoring one component and feeding obs data to development



Generating Safety Case in CI/CD



A Modular product structure enabling continuous deployment should

- ***Support concrete verification and argumentation plans several steps*** ahead together with product increment for each deployment candidate. Such plans imply specification of verification criteria and criteria for valid evidence to be produced, and what method to be used for each piece of evidence.

- Show how to, **several steps ahead in the product planning, make an efficient combination of verification methods** such that they both are complementary in each step and that what is produced in one step is supporting upcoming steps.

- The verification and argumentation structures, respectively, are closely related to the requirement structure modularized **by means of a contract structure** in four dimensions (abstraction, aggregation, allocation, and functional relation).

- The methodology also includes **how to apply general design principle**s (as for example separation of complexity and criticality) in the context of highly automated functionality and frequent updates.

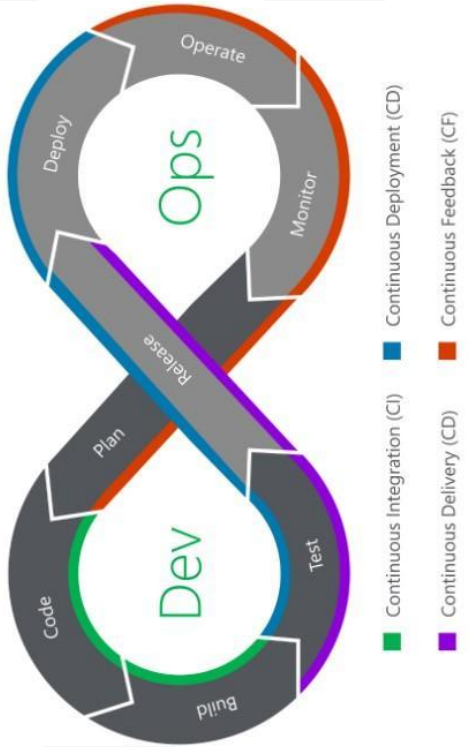
- Analyse and Identify **what is needed in terms of tool support** to generate valid ety cases for each deployment candidate.

Always Releasable!!!

Defining the DevOps, Run-time collected data used for Design-time (Code)

COLLECTION OF OPERATIONAL DATA

- This is the Ops part, ***essential*** to constitute a real DevOps, for Continuous Deployment.
- Carefully chosen collection of ***run-time data***, is fundamental in creating lessons learned used in coming versions of design-time updates of the ADS. The learning-loop from observing the field, **is not** by (too late) observing accidents and incidents, rather **collecting data identified as creating evidence in coming versions of the individual safety property.**
- Can be done by expressing a *triggering condition* to collect field data comparing how well (and hypothetical) a claimed absence of a pedestrian is consistent with a later observation of the related data volume.



Needs from a Safety context

- Support for general and stable safety tactics and patterns – e.g. ASIL B(D) for certain general failure modes (and also POSIX) but also allocatable by specific safety requirements
- Development needs – e.g. need to allow fast iterations without regression, additional merge conditions, re-basing – i.e. Continuous Integration and Deployment
- Allowing to add new features, i.e. changing the argumentation structure, architecture (i.e. static/dynamic) not only support for increased performance
- Verification and Validation needs – e.g. efficient and frequent generation of safety cases, certain activities take long time and must be accounted for.

Key Components of the CI/CD architecture

Rhoda ENGINEERING

1

Core Flow:

- **Code** (Git/GitLab) → **Build** (CMake/Yocto/ARM Toolchains) → **Test** (HIL/ISO 26262 Checks) → **Deploy** (OTA/ECU Flashing) → **Monitor** (ELK/Grafana)

2

Key Automotive Features:

- Hardware-in-Loop (HIL) testing with CANoe/dSPACE
- ISO 26262/MISRA compliance via Polyspace/LDRA
- Embedded toolchains (AUTOSAR, ARM GCC)
- Safety-critical artifact traceability

3

Infrastructure:

- Hybrid cloud/edge (Kubernetes + OpenStack)
- Data versioning (DVC) & streaming (Kafka)
- Infrastructure-as-Code (Ansible/Terraform)

4

Compliance:

- End-to-end audit trails
- TÜV-certified static analysis
- Secure OTA updates (Mender/RAUC)

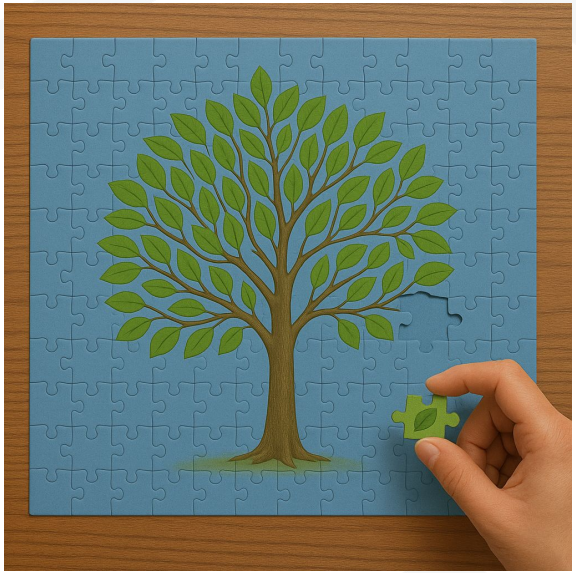
With courtesy of Rhode Engineering, © Teddy

Project in the Pipeline – Adressing the following…

- Enablers to make Rapid DevOps (special focus on Continuous Deployment) **Real**
- Methodology: Coordinated planning for a series of upcoming versions (candidates)
  - Product updates and new features.
  - Verification and Validation methods
  - Continuous Assurance Cases (Safety/Quality/Legal)
- Multi-stage Data-driven verification
  - Design for dedicated data-collection
  - **Future argumentation structures specifies data needed as evidence**
- Modular argumentation structure to enable automatization for CI/CD
  - Enable ***updatability*** with limited impact on argumentation – Safety Case
  - Explicitly showing still missing evidence (verification and data to be collected)

Final Thoughts on Linux/ELISA

- Linux introduction into the safety realm requires a technical and cultural transformation
- Can ELISA become a key enabler of this shift….
- Safe CD is essential for modern vehicles
- How to make Linux play a complementing part in the marriage for safety? Providing ASIL B(D) possibilities etc



THANK YOU!!!!

VOLVO