Turning the Ignition on Safety: Zephyr RTOS in Automotive Compliance

Saravanan Sekar

whoami

Saravanan Sekar, Linumiz

- Embedded Software Engineer at Linumiz
- Embedded Linux: BSP, U-Boot, Linux kernel, Yocto Project, Buildroot
- Zephyr: SoC and board support, drivers

www.linumiz.com

About Linumiz

Linumiz Embedded software engineering and consulting for Linux and Zephyr ranges from board bring-up, board support package (BSP), device drivers, Over the air software updates (OTA) primarily in but not limited to ARM

Embedded Linux

- Board support package
- Driver Development
- Board Bring up
- Custom Linux build
- Application Development

Zephyr RTOS

- Board porting
- Firmware Development
- Driver development
- Update management
- Upstream activities

Consulting

- HW recommendation
- Architecture design
- System development
- Support for specific issues
- Support Services

Device security

- Static analysis
- Dynamic analysis
- Device hardening
- Root of Trust (RoT)
- Security recommendations

Linumiz
open-source powered

Agenda

- Zephyr meets automotive
- Why safety & ISO 26262
- Route 3S, SEooC, and scope
- Where Linumiz fits
- CAN example: TPMS to tell-tale
- Live demo: traceability dashboard

Zephyr already speaks automotive

Determinism and bounded latency are baseline kernel properties, not opt-in extras.

✓ **Deterministic timing**	→ Tickless kernel
✓ **Bounded interrupt latency**	→ Zero-Latency Interrupts (ZLI)
✓ **Spatial isolation**	→ MMU / MPU memory domains
✓ **Predictable scheduling**	→ Cooperative + preemptive, bounded latency
✓ **Disciplined C**	→ MISRA C:2012 (no UB-prone patterns)

Ten years upstream. Same kernel ships in ~1500 boards today.

Benchmarks already in the tree

tests/benchmarks/ ships with Zephyr — runs on any supported board, no extra dependencies.

latency_measure

ISR → thread latency · Thread → thread latency

sched_userspace

Scheduler overhead with user-mode threads

timing_info

High-resolution timing primitives

app_kernel

Kernel API throughput

NEW in 4.1 · thread_metric ported upstream

Standard cross-RTOS benchmark. Zephyr 4.1: ≈ Eclipse ThreadX · > FreeRTOS in most cases.

Drop into CI. Regress against a baseline. Fail the build when latency creeps up.

Why Safety in Automotive RTOS?

Modern Vehicle: 100+ ECUs

- Braking System
- Steering Control
- Airbag Controller
- ADAS Sensors
- Body Control
- Infotainment

ASIL Classification

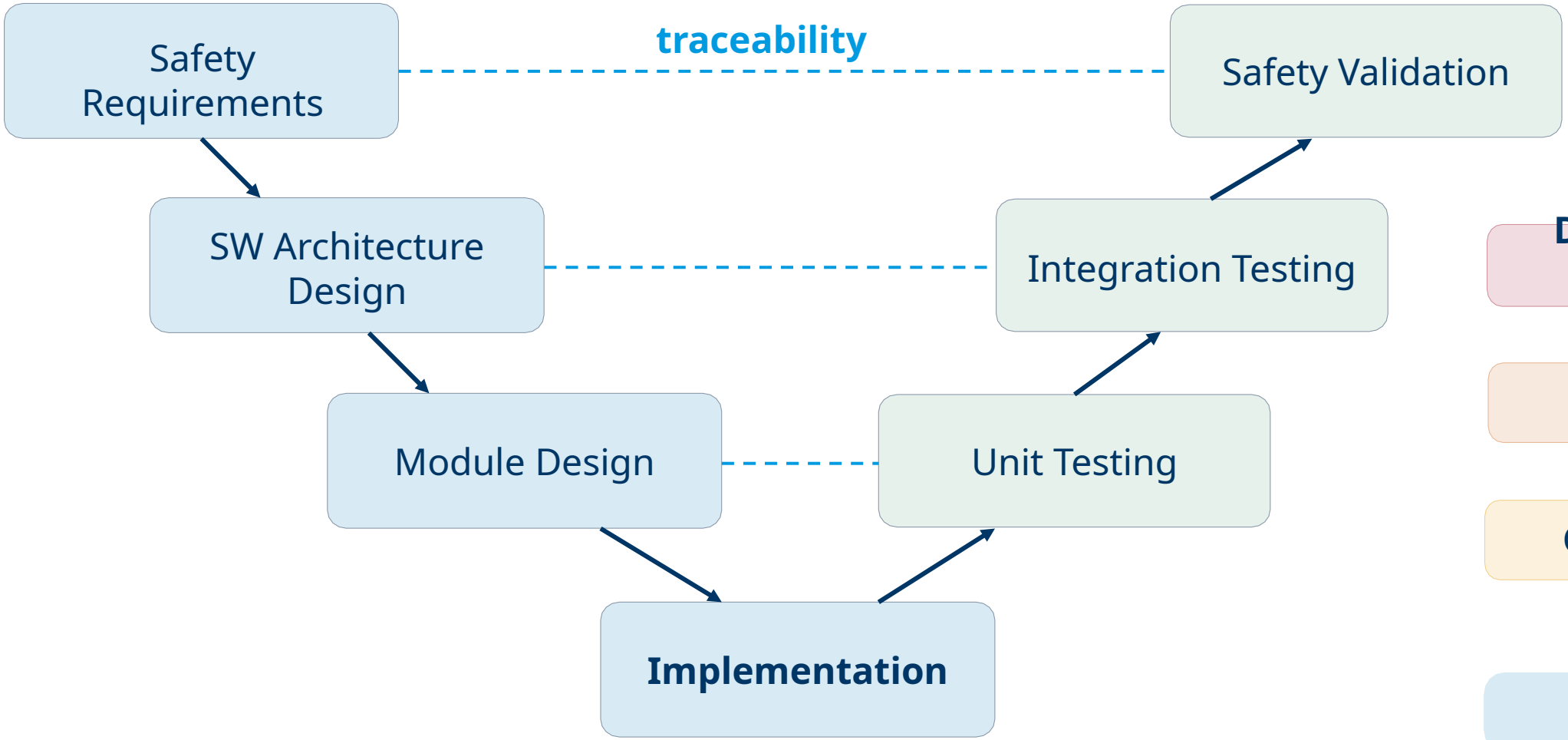
ASIL D	Steering, Braking
ASIL C	Airbag, ABS
ASIL B	Cruise Control
ASIL A	Rear Lights
QM	Infotainment

RISK LEVEL

ASIL D requires MC/DC coverage. Zephyr targets statement & branch only. Integrators must close this gap.

Linumiz
open-source powered

ISO 26262: Automotive Functional Safety



ASIL Determination

Severity

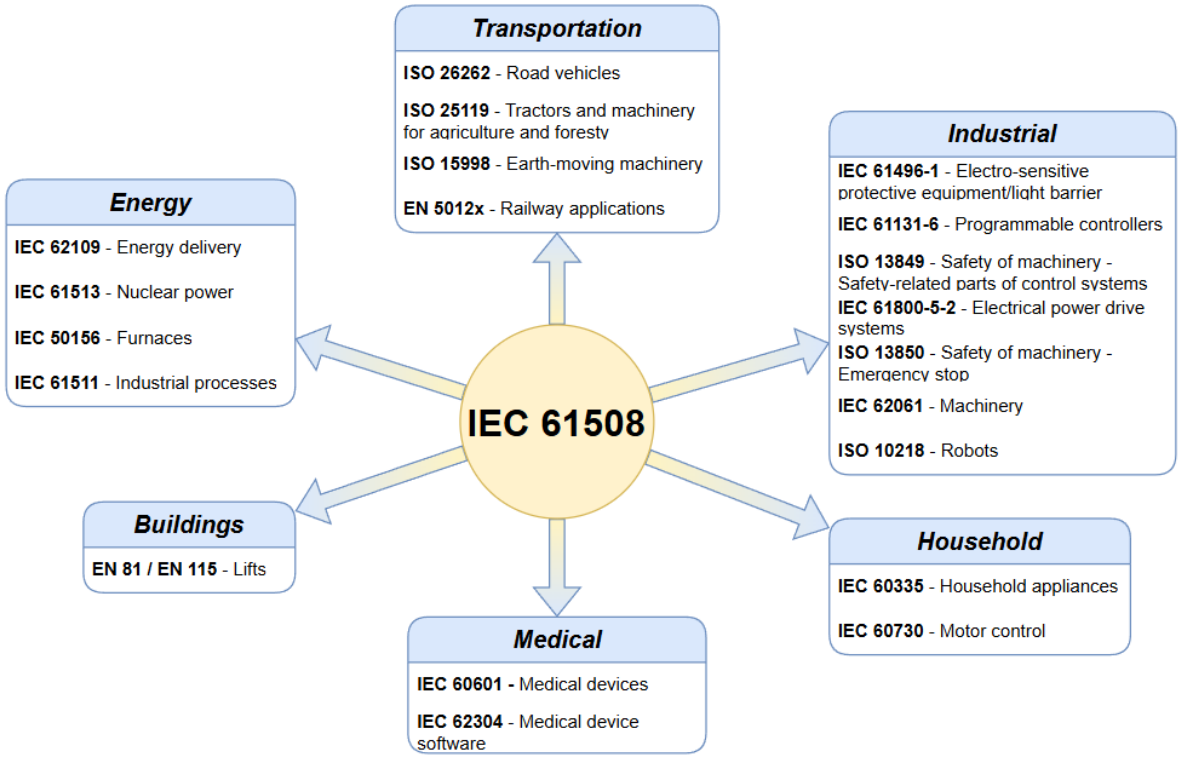
Exposure

Controllability

= ASIL A..D

The pathway: Route 3S meets SEooC

Two complementary pre-existing-software routes — one for the kernel, one for the driver layer.



Route 3S

IEC 61508-3 §7.4.2.12 — pre-existing-software pathway. Proven-in-use evidence + retrofit requirements + traceability + assessment.

Zephyr SWG's chosen path for the kernel.

SEooC — Safety Element out of Context

ISO 26262 Part 10 §9. Develop a safety element against assumed integration conditions; integrator validates those assumptions in-system.

IEC 61508 — the umbrella functional-safety standard. One certification maps to ISO 26262 (auto), IEC 62304 (medical), EN 5012x (rail), and others.

Same goal — reuse OSS code under a safety lifecycle. One pathway for the kernel, the other for what sits above it.

Linumiz
open-source powered

Scope: covered, not covered

✓ COVERED — Zephyr SWG (IEC 61508 SIL 3 / SC 3)

- Core kernel: scheduler, IPC primitives, memory management, time services
- A selected subset of subsystem APIs
- Documented configuration profile & assumptions

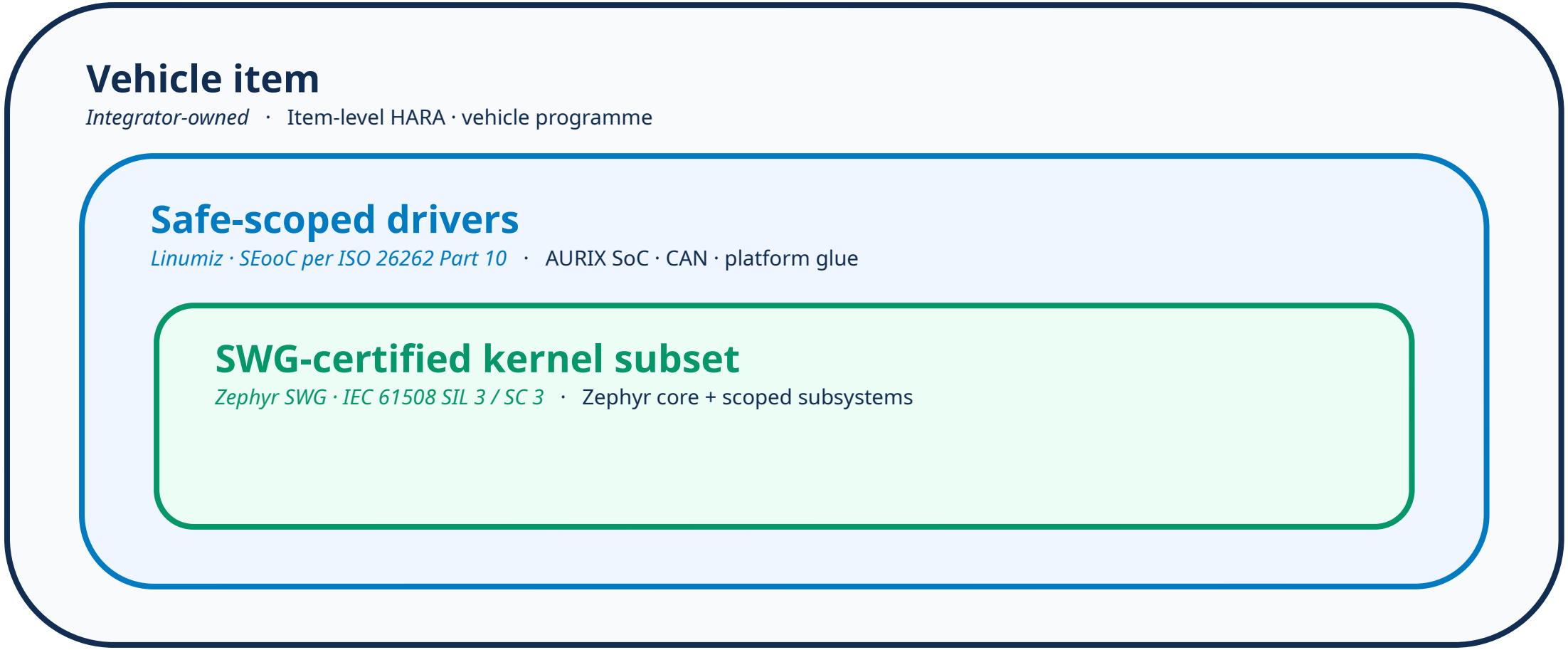
✗ NOT COVERED — integrator delivers

- Vendor-specific SoC drivers (TriCore, Cortex-R52, RH850, S32, …)
- Automotive protocol stacks (CAN-XL, J1939, ISO-TP, AUTOSAR adapters)
- Item-level HARA — always integrator-owned
- Toolchain qualification (HighTec-RT for AURIX, etc.)

Kernel cert covers ~70% of a safety case. The other 30% — drivers, glue, integration evidence — is what Linumiz delivers to ASIL B today.

Where Linumiz fits

The layered-scope view — three rings, three owners, three sets of evidence.



Linumiz fills the gap between the kernel certification and the vehicle item — driver layer, SEooC evidence, integration support.

Linumiz
open-source powered

OSS velocity vs audit cadence

Two clocks at different speeds — and the five levers that sync them.

⚠ THE FRICTION

Mainline Zephyr moves daily. The safety case freezes on the audit calendar.

- Bug fix on mainline doesn't auto-propagate to the safety branch.
- Backporting ⇒ re-review, re-trace, re-test.
- Mainline drift can invalidate consumed-assumption arguments.

✓ HOW LINUMIZ RECONCILES

- **Pin to LTS**
  Zephyr 4.3.0 LTS as base. Bug-fix-only uptake keeps the re-verification surface small.
- **SEooC framing**
  Drivers as Safety Elements out of Context, with explicit assumptions over the consumed Zephyr API.
- **Wrapper pattern**
  Thin Linumiz-owned safety wrapper on top of upstream backend — validation, mode-gate, query API.
- **Findings flow upstream**
  Error-code normalisation, atomic_t state introspection ⇒ back to mainline.
- **StrictDoc traceability**
  Every requirement → parent + code, machine-validated.

Two clocks. One pinned LTS plus a thin wrapper makes them sync.

Linumiz
open-source powered

What Linumiz is working on

Zephyr porting on a real automotive platform, with the safety case in parallel.

1

Infineon TriCore AURIX porting

Bringing Zephyr to a real automotive MCU family.

2

Automotive focus

Platform drivers · BSP · application enablement.

3

Safety case in parallel

ISO 26262 work-products grow alongside the engineering, not after it.

Linumiz
open-source powered

Worked example: TPMS to tell-tale

Tire-pressure data crosses a CAN bus and lights up a dashboard warning. The CAN driver is the bridge we safety-certify.

CAN bus @ 500 kbps

Tell-tale ECU

TPMS LOW

Driver alert

SEooC SCOPE

- CAN driver developed as a Safety Element out of Context.
- Assumptions: scoped Zephyr API · ASIL B target.
- Integrator validates assumptions in the vehicle context.

APIs IN SCOPE

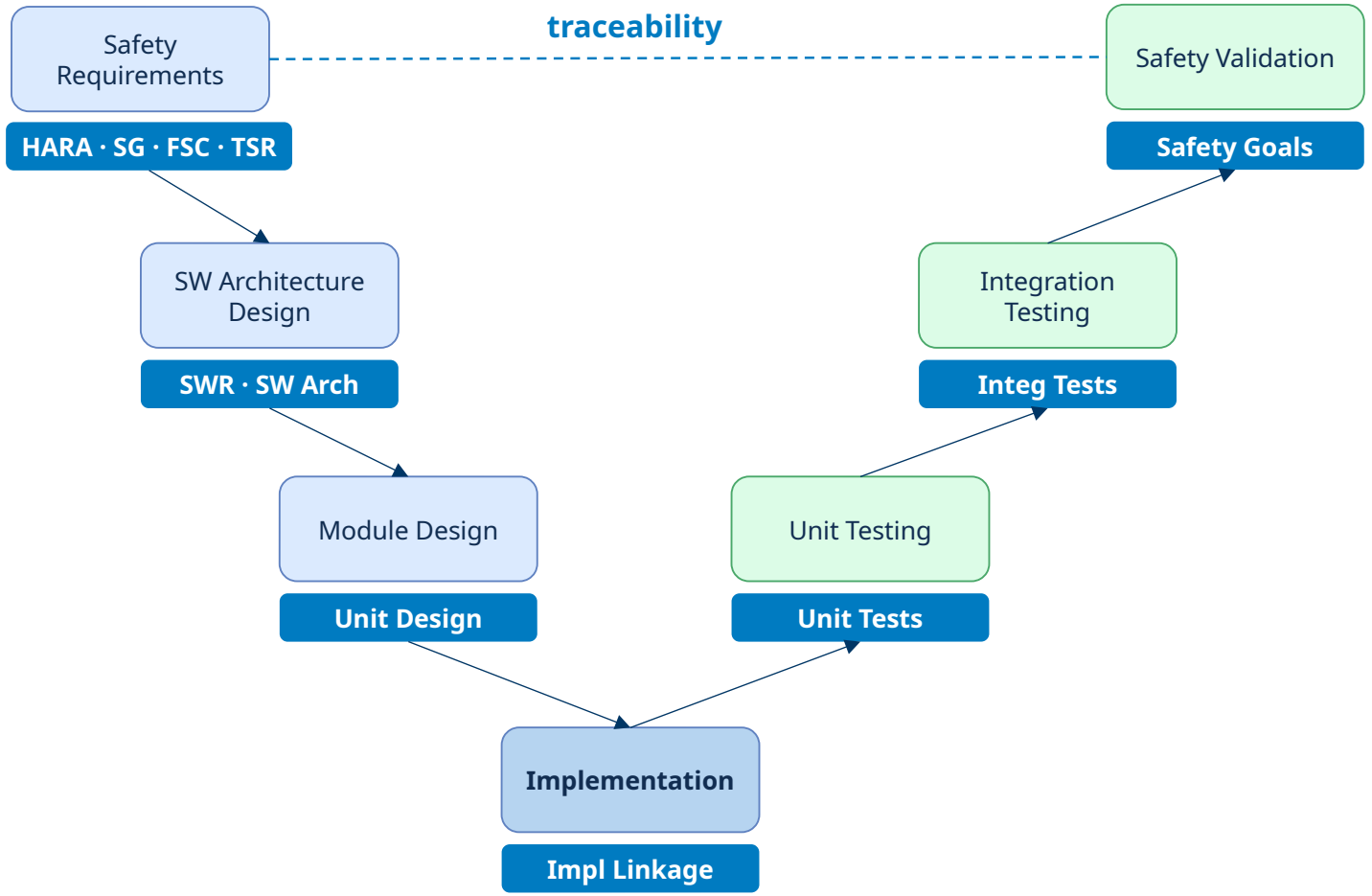
- **can_send()**
  Frame transmission
- **can_set_bitrate()**
  Bus configuration

One bus, two APIs — that's the whole certified surface.

Linumiz
open-source powered

V-model, but for CAN: every stage has an .sdoc

Same V-model from earlier — now annotated with the sdoc artifact we authored at each stage.



Our CAN dossier

Concept
- HARA
- Safety Goals
- Functional Safety Concept

System
- Technical Safety Concept

Software
- SW Safety Requirements
- SW Architecture & Unit Design

Verification
- Software Test Plan

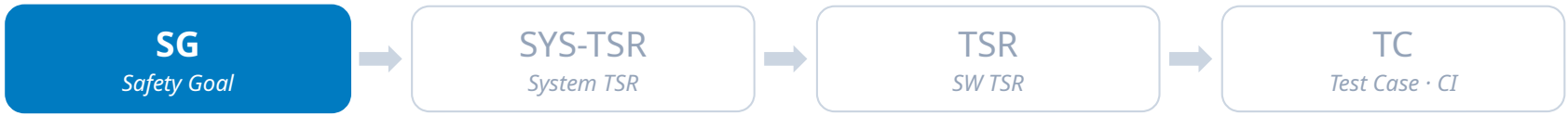
Code
- Implementation Linkage

SEooC
- Technical Context
- Consumed Assumptions
- Integrator Obligations

Every V-stage has a signed-off sdoc — that's the dossier.

Linumiz
open-source powered

Stage 1 · Safety Goals



ASIL B

SG-CAN-001

Prevent silent CAN frame loss

If the controller cannot transmit a frame, the driver returns an error within the configured timeout — never silent.

↑ *from HAZ-CAN-001*

concept/safety_goals.sdoc

SG-CAN-002

Prevent bitrate misconfiguration during active operation

Reject any attempt to change CAN bitrate unless the controller is in STOPPED state.

↑ *from HAZ-CAN-002*

concept/safety_goals.sdoc

SG-CAN-003

Detect and report CAN bus-off condition

When the controller enters bus-off, detect it and block further transmission attempts.

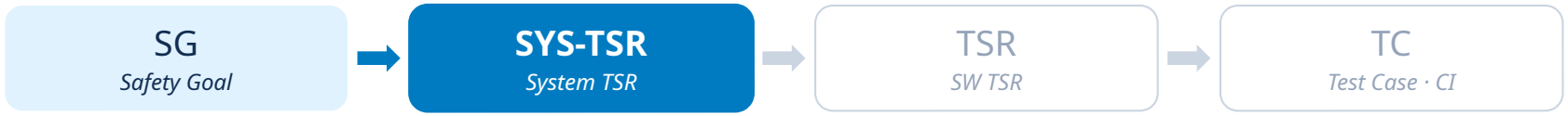
↑ *from HAZ-CAN-003*

concept/safety_goals.sdoc

Three safety goals — one per hazard. The whole chain hangs off these.

Linumiz
open-source powered

Stage 2 · System-level Technical Safety Requirements



ASIL B

SYS-TSR-001

Distinct error codes for each failure mode

Return a unique non-zero error per failure mode: -ENETDOWN, -EBUSY, -ENETUNREACH, -EALREADY, -EINVAL, -ENOTSUP, -EAGAIN.

↑ *derives from SG-CAN-001*

system/technical_safety_concept.sdoc

SYS-TSR-004

CAN driver shall protect shared resources from concurrent access

All shared resources in the CAN driver (timing registers, TX buffer pool, TX semaphore count) shall be protected by mutual exclusion mechanisms (mutex or semaphore) to prevent data corruption from concurrent multi-threaded access.

↑ *derives from SG-CAN-002*

system/technical_safety_concept.sdoc

SYS-TSR-005

CAN driver shall clean up resources on any failure path

When any internal operation fails (register write, MRAM access, semaphore timeout), the CAN driver software shall release all acquired resources (mutexes, semaphores) before returning the error code. No failure path shall leave a resource in a locked or acquired state.

↑ *derives from SG-CAN-003*

system/technical_safety_concept.sdoc

Each safety goal becomes a system requirement the driver has to satisfy.

Linumiz
open-source powered

Stage 3 · Software Technical Safety Requirements

SG *Safety Goal* → SYS-TSR *System TSR* → **TSR** *SW TSR* → TC *Test Case · CI*

ASIL B

TSR-BITRATE-002

Protect timing register writes with mutex

When can_set_bitrate() writes to the NBTP timing register, the write operation SHALL be protected by a mutex to prevent concurrent access from multiple threads.

`can_mcan.c : 1017-1018` ↑ *derives from SG-CAN-001 · SYS-TSR-001* *software/sw_safety_requirements.sdoc*

TSR-BITRATE-004

Report register write failure

If the write to the NBTP timing register fails, can_set_bitrate() SHALL return the error code from the register write operation and SHALL NOT leave the timing in a partially configured state.

`can_mcan.c : 221-222` ↑ *derives from SG-CAN-002 · SYS-TSR-002* *software/sw_safety_requirements.sdoc*

TSR-START-001

Reject start when already started

When the CAN controller is already in STARTED state, can_start() SHALL return -EALREADY and SHALL NOT modify the controller state.

`can_mcan.c : 1021-1027` ↑ *derives from SG-CAN-003 · SYS-TSR-003* *software/sw_safety_requirements.sdoc*

Each SYS-TSR pinned to a software requirement — and to a specific range of lines in the driver.

Linumiz open-source powered

Stage 4 · Test Cases · CI / CD

ASIL B

SG *Safety Goal* → SYS-TSR *System TSR* → TSR *SW TSR* → **TC** *Test Case · CI*

TC-BITRATE-003

Verify can_set_bitrate() is thread-safe (mutex protection)

ZTEST(can_bitrate, test_set_bitrate_thread_safety)

test_can_bitrate.c ↑ *verifies TSR-SEND-001* *software/test_plan.sdoc*

TC-BITRATE-006

Verify standard automotive bitrates are accepted

ZTEST(can_bitrate, test_set_bitrate_standard_automotive_rates)

test_can_bitrate.c ↑ *verifies TSR-BITRATE-001* *software/test_plan.sdoc*

TC-START-001

Verify can_start() returns -EALREADY when already started

ZTEST(can_start_suite, test_start_rejects_double_start)

test_can_send.c ↑ *verifies TSR-SEND-002* *software/test_plan.sdoc*

CI / CD

Test runner
west twister --integration

Build
west build · west test

Workflow
.github/workflows/ci.yml

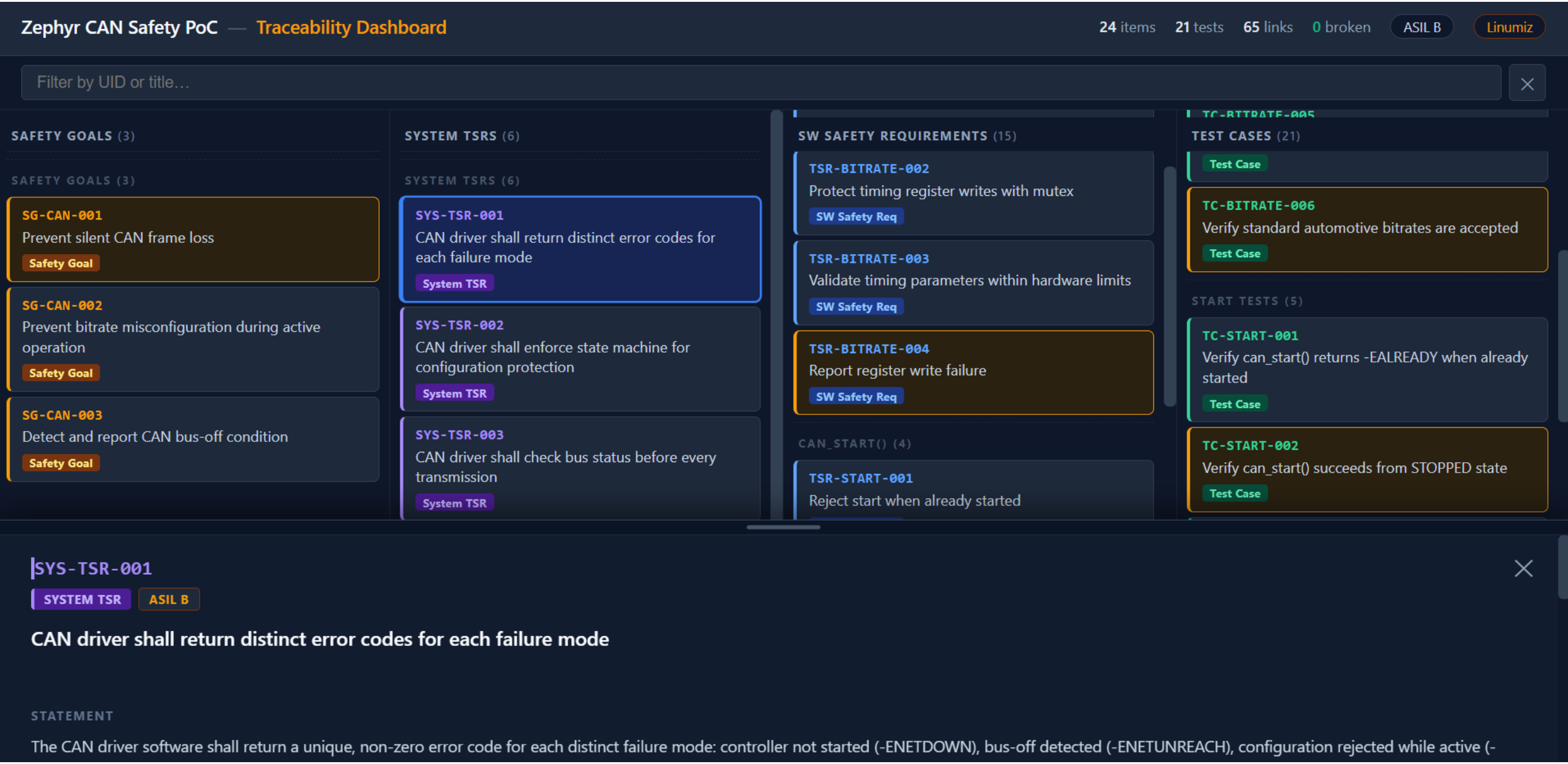
Trigger
PR · push · nightly

✓ **21 / 21 tests pass**

Every requirement has a test. Every test runs in CI on every change.

Linumiz open-source powered

Zephyr CAN Safety - Demo



Thank You

contact@linumiz.com

Linumiz
open-source powered