

Responsible use of Open-Source Software



Safionyx

From code to compliance

from the Automotive SPICE & Cybersecurity
perspective

Alexander Much · Safionyx GmbH & Co. KG
2nd eLSA Symposium · Göttingen · 1 October 2026

Profile

Timeline:

- 1995-2004: Mathematics & computer sciences in Erlangen and Cambridge (UK)
- 2000-2003: SuSE Linux
- 2003-2025: Elektrobit Automotive GmbH
Director, Head of quality, functional safety, cybersecurity and software compliance
- 2026: Safionyx GmbH & Co. KG, Founder



Qualifications:

- Principal automotive SPICE assessor
- ECQA certified automotive quality manager
- ECQA certified functional safety manager
- EuroSPI certified cybersecurity manager

Committees:

- VDA Automotive SYS conference
- EuroSPI conference
- Intacs working group cybersecurity and AI
- Soqrates working group
- ... and others

A recurring moment in assessments

**“Who is your supplier for
this component?”**



Safionyx

From code to compliance

“It’s Linux.”

Everything the qualification machinery assumes is
missing — and every standard still applies.

Qualification assumes what open source does not provide

Same standards, same product liability — evidence of an entirely different shape

PROPRIETARY COMPONENT

- A named supplier, a contract, an agreed scope
- A requirements specification, including safety and security requirements
- A development process you may audit — or a supplier audit report you may read
- An escalation path with agreed reaction times
- A roadmap and an end-of-support commitment

OPEN-SOURCE COMPONENT

- A community — not a counterparty
- Requirements live implicitly in code, tests and discussion threads
- Process evidence exists, but never in the form the audit model expects
- No obligation towards you: you are not a customer
- Maintenance ends when the maintainers decide it ends

Ways to get this wrong

These are common, both are comfortable, both leave the product unprotected

1 False failure

Apply the model literally to the upstream project.
Ask a kernel maintainer for bidirectional traceability and a validated tool chain.

Result: everything is non-conformant, the assessment is quietly ignored, and the component ships anyway — now entirely unmanaged.

2 False comfort

An SBOM is generated, licenses are scanned, the checklist is green. The audit model has been satisfied.

Nobody has asked who reacts when a CVE lands on a Friday evening in year seven.

PART 1

What are we actually assessing?

The scope question decides everything that follows.



Safionyx
From code to compliance

You cannot assess a community, only your use of it

The assessed object is your integration, qualification and maintenance — not the upstream project's process.

1 Scope

The upstream project is not in your scope, and never was. Your selection, integration, verification and maintenance of the component are.

2 Evidence

What upstream provides is an input to your argument. The argument itself is yours to make — and yours to defend.

3 Accountability

It stays with the manufacturer. ISO 26262, ISO/SAE 21434, UNECE R155 and the CRA agree on this without exception.

The anchors already exist in the standards we use

No new process area is needed. What is needed is the discipline to apply these without a supplier.

Automotive SPICE

ACQ.4-style supplier monitoring — with no supplier to monitor

SUP.8 - SUP.10: upstream drops are configuration and change items

SWE.4 - SWE.6: the verification you run yourself, on your integration

MAN.3 / MAN.5 / (MAN.7): the maintenance effort and the risk both belong in the plan

Functional safety

ISO 26262-8 §12: qualification of software components

ISO 26262-8 §14: proven-in-use / confidence from use arguments

SEooC thinking: assumptions must be stated, then validated in your context

Freedom from interference remains your integration problem

Cybersecurity & CRA

ISO/SAE 21434 cl. 7: distributed activities, with no interface agreement to sign

ISO/SAE 21434 cl. 8: continual monitoring and vulnerability management

CRA Art. 13(5): due diligence when integrating FOSS components

CRA Annex I Part II: vulnerability handling across the support period

PART 2

Reading the temperature

The project is the temperature. The audit
model is only the thermometer.



Safionyx

From code to compliance

Six things you can actually measure in a project

Legitimate, observable evidence — the recalibrated thermometer

1 Release discipline

Is there a maintained stable or LTS line, or only HEAD? Are releases tagged, signed and reproducible?

2 Maintainership

Who may merge? Is review enforced or optional? What is the bus factor, and who pays those people?

3 Test & CI evidence

Does a suite exist, does it run per change, are results public, and does anyone act on a red build?

4 Vulnerability handling

A security contact, a disclosure policy, and a CVE history you can measure response times against.

5 Change rationale

Issue → discussion → patch → commit.
Can you still reconstruct why a line of code exists?

6 Governance & provenance

Documented decision rights, license hygiene, contribution provenance & IP rights, machine-generated SBOM.

What these signals do not do

A healthy project used carelessly is still a finding

They describe the project, not your product

Your safety and security requirements, your integration verification and your validation in your operational context remain entirely yours.

They are indicators, not conformity evidence

A strong upstream reduces your risk and shortens your argument. It never replaces the argument, and it transfers no obligation.

They decay

Every one of these readings has a shelf life. A project healthy at selection can be abandoned three years later — measure again, on a defined cadence.

What the CRA adds: named roles

Regulation (EU) 2024/2847 — reporting obligations live since 11 September 2026, full application 11 December 2027

Manufacturer

Places the product with digital elements on the market. Annex I Part I and Part II, support period, Art. 14 reporting ladder. Full liability.

OSS steward (Art. 24)

A legal person sustaining an OSS project intended for commercial use without monetising it. Documented cybersecurity policy, cooperation with authorities. Exempt from fines.

Non-commercial contributor

Outside the scope of the regulation. The person who wrote the code you depend on may owe you, and the market, nothing at all.

Why this matters to an assessor

A steward is a named entity with a written, verifiable cybersecurity policy — the first thing in this ecosystem you can actually point at, ask for, and read. Art. 25 adds voluntary security attestation schemes on top.

But accountability does not transfer

The steward is not your supplier, and the CRA never says otherwise

WHAT A STEWARD IS NOT

- Not a contractual counterparty — no warranty, no liability, no agreed reaction time
- Not a route to delegate Annex I: the essential requirements remain the manufacturer's
- Not a guarantee of continuity — stewardship can end
- Not a substitute for your own due diligence under Art. 13(5)

WHAT YOU MUST OWN INTERNALLY

- A named maintainer per component — a person, not a department
- Budgeted upstreaming: a patch you upstream is maintained for you, a patch you keep is maintained by you
- Someone who watches the feeds, and someone who triages what they find
- An agreed internal reaction time, tested at least once before you need it

The long tail: your product outlives the upstream branch

15 to 25 years of field life against a stable branch maintained for two to six

STAY CLOSE TO UPSTREAM

Cheap over the lifetime, but it demands continuous discipline: regular rebases, upstreamed patches, a delta you can state in one sentence.

FREEZE AND OWN IT

Legitimate — but you have taken over the maintainer role, including security fixes, and you must staff and fund it for the full support period.

The finding is never the choice. The finding is doing neither, for years, and calling it a decision.

PART 3

What good and bad look like

From several years of assessing projects that use open source in earnest.



Safionyx

From code to compliance

What I find acceptable — and what I do not

ACCEPTABLE

- Upstream-first, with a documented delta that is shrinking
- Version chosen on a written rationale that includes the maintenance outlook
- A component qualification argument supported by your own tests, in your configuration
- A named maintainer, monitored feeds, and a CVE turnaround you can show with dates
- An SBOM produced by the build, per release

PROBLEMATIC

- “We took what was current at the time” — and it has not moved since
- An SBOM maintained by hand, in a spreadsheet, updated before audits
- Patches living only in a local fork, with no owner and no upstream intent
- Security monitoring that consists of a mailing list nobody reads
- “It is widely used” offered as the qualification argument

Five questions I ask in an assessment

- 1 Why this component, and why this version? Show me the decision, not the outcome.
- 2 Show me the delta to upstream, and tell me who owns it.
- 3 Take the last CVE that hit this component. Walk me through what happened, with dates.
- 4 Who merges a change into your build — and what do they check before they do?
- 5 What happens to this component in year twelve?

None of these can be answered by a document that was written for the assessment.

**The project is the
temperature.**

**The audit model is only
the thermometer.**



Safionyx

From code to compliance

Do not throw the thermometer away — recalibrate it.

And let us agree on a shared vocabulary between
assessors and open-source projects, so that projects
can be judged on what they genuinely do well.

alexander.much@safionyx.com · safionyx.com