ELISA
Enabling Linux in Safety Applications

WORKSHOP

Continuous Compliance in Safety-Critical Open-Source Projects

Rinat Shagisultanov
VP, Modern Dev & App Services
InfoMagnus
ELISA Ambassador

November 18-20, 2025
Co-hosted with Red Hat

The Open-Source Safety Paradox

Modern systems: 70-90% open-source components (LF, 2022)

Safety processes: Designed for waterfall, closed development

Result: Innovation bottlenecks that slow both compliance AND development

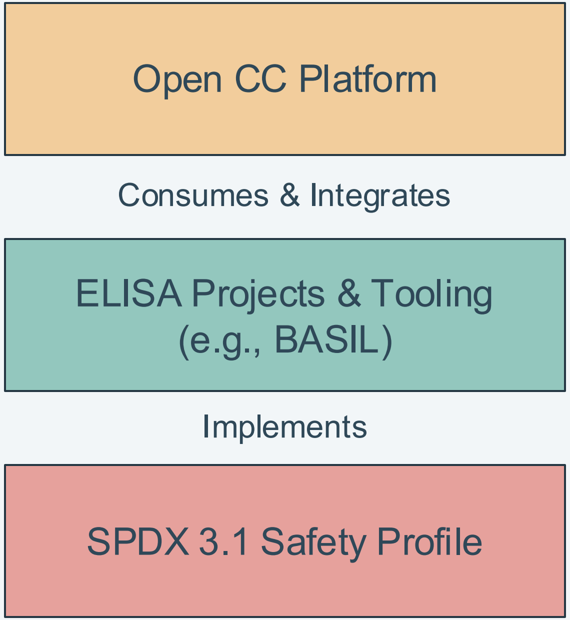
Industry Reality Check

Open Source is Everywhere

- **Automotive**: Tesla runs Linux; Android Automotive is OEM-ready
- **Aerospace**: SpaceX Dragon runs Linux for critical functions
- **Medical**: 50% of device manufacturers considering Linux transition
- **Industrial**: 70% of IoT devices use open source

Standards don't disappear: ISO 26262, IEC 61508, etc. still apply

Where Our Platform Fits in the ELISA Ecosystem



Open CC Platform

Compliance automation, dashboards, & reporting

Consumes & Integrates

ELISA Projects & Tooling (e.g., BASIL)

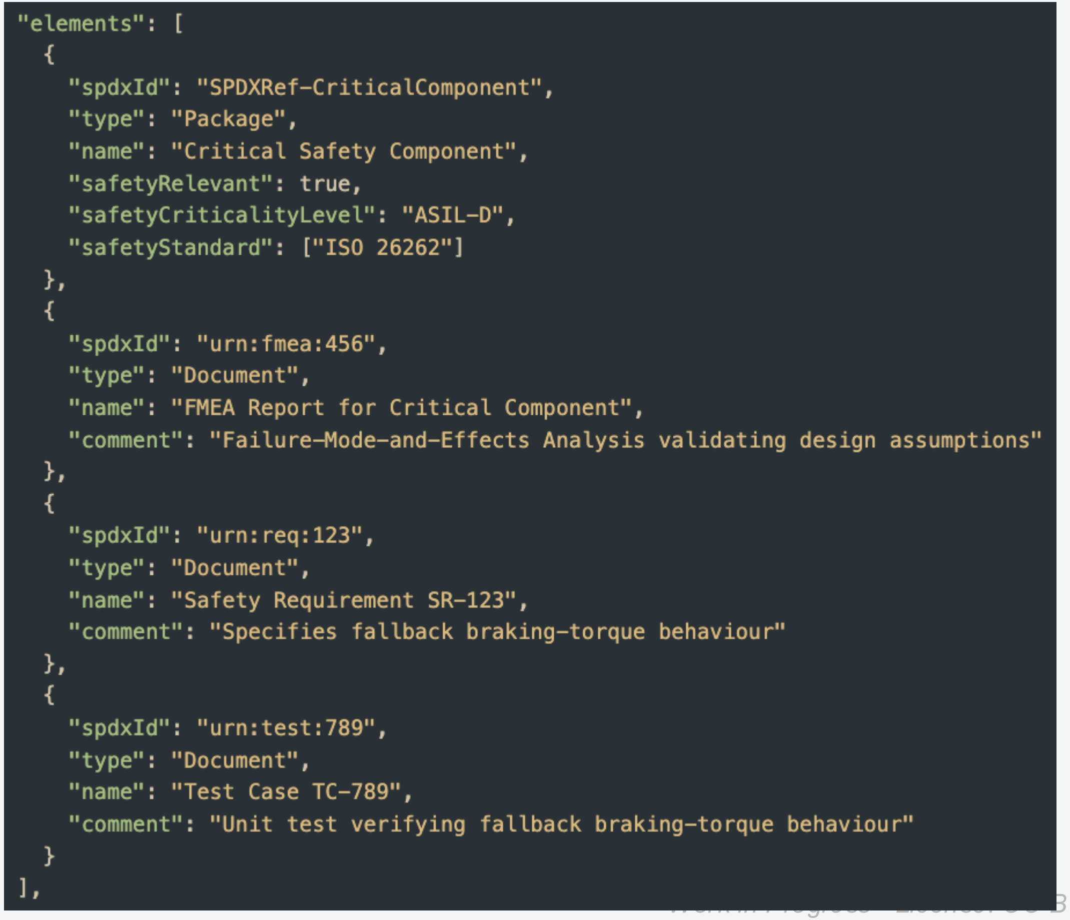
FuSa traceability & safety-annotated SBOMS

Implements

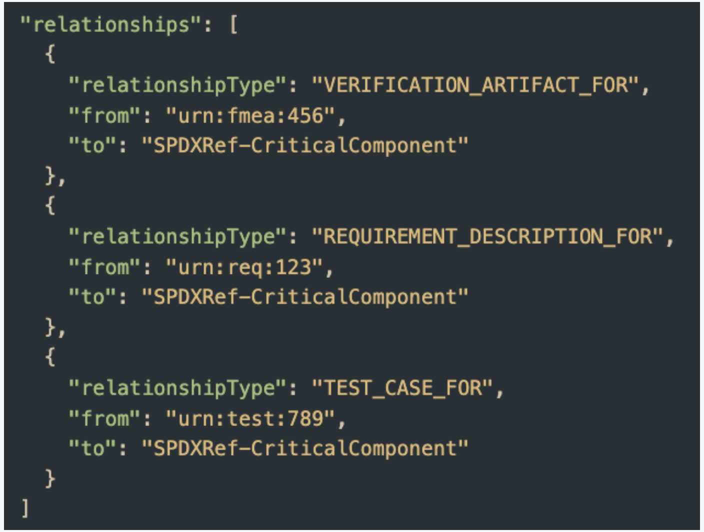
SPDX 3.1 Safety Profile

The data standard

Safety-Annotated SBOMs



```
"elements": [
  {
    "spdxId": "SPDXRef-CriticalComponent",
    "type": "Package",
    "name": "Critical Safety Component",
    "safetyRelevant": true,
    "safetyCriticalityLevel": "ASIL-D",
    "safetyStandard": ["ISO 26262"]
  },
  {
    "spdxId": "urn:fmea:456",
    "type": "Document",
    "name": "FMEA Report for Critical Component",
    "comment": "Failure-Mode-and-Effects Analysis validating design assumptions"
  },
  {
    "spdxId": "urn:req:123",
    "type": "Document",
    "name": "Safety Requirement SR-123",
    "comment": "Specifies fallback braking-torque behaviour"
  },
  {
    "spdxId": "urn:test:789",
    "type": "Document",
    "name": "Test Case TC-789",
    "comment": "Unit test verifying fallback braking-torque behaviour"
  }
],
```



```
"relationships": [
  {
    "relationshipType": "VERIFICATION_ARTIFACT_FOR",
    "from": "urn:fmea:456",
    "to": "SPDXRef-CriticalComponent"
  },
  {
    "relationshipType": "REQUIREMENT_DESCRIPTION_FOR",
    "from": "urn:req:123",
    "to": "SPDXRef-CriticalComponent"
  },
  {
    "relationshipType": "TEST_CASE_FOR",
    "from": "urn:test:789",
    "to": "SPDXRef-CriticalComponent"
  }
]
```

V-Model Traceability with SPDX Relationships

V-Model Element	SPDX 3.x Relationship	Explanation
Requirement → Code	REQUIREMENT_DESCRIPTION_FOR	Links code to the requirement it fulfills
Requirement → Test Case	TEST_CASE_FOR	Links test to the requirement it verifies
Code → Test Case	TEST_CASE_FOR	Maps tests directly to implementation
Test Result → Test Case	VERIFICATION_ARTIFACT_OF	Links test results to test cases
Tool Output → Code	VERIFICATION_ARTIFACT_OF	Links verification tool results to test cases

BASIL
The FuSa Spice

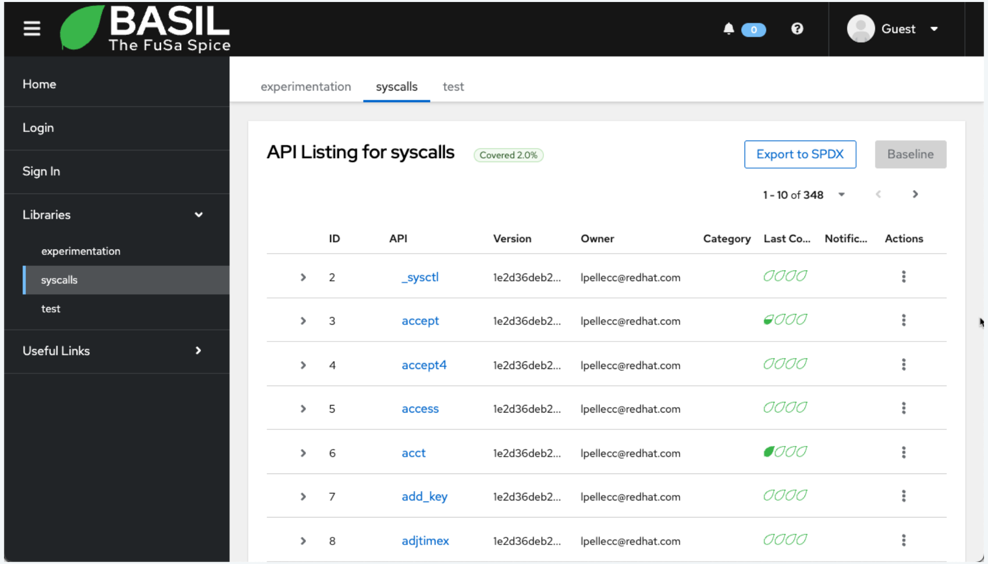
Key capabilities

Single source of traceability

Built-in gap & completeness analytics

Test-execution awareness

SPDX Model 3 export



API Listing for syscalls Covered 2.0%

1 - 10 of 348

ID	API	Version	Owner	Category	Last Co...	Notific...	Actions
2	_sysctl	1e2d36deb2...	lpellecc@redhat.com				⋮
3	accept	1e2d36deb2...	lpellecc@redhat.com				⋮
4	accept4	1e2d36deb2...	lpellecc@redhat.com				⋮
5	access	1e2d36deb2...	lpellecc@redhat.com				⋮
6	acct	1e2d36deb2...	lpellecc@redhat.com				⋮
7	add_key	1e2d36deb2...	lpellecc@redhat.com				⋮
8	adjtimex	1e2d36deb2...	lpellecc@redhat.com				⋮

https://github.com/elisa-tech/BASIL

Luigi Pellecchia & Gabriele Paoloni, Red Hat

ELISA Enabling Linux in Safety Applications WORKSHOP

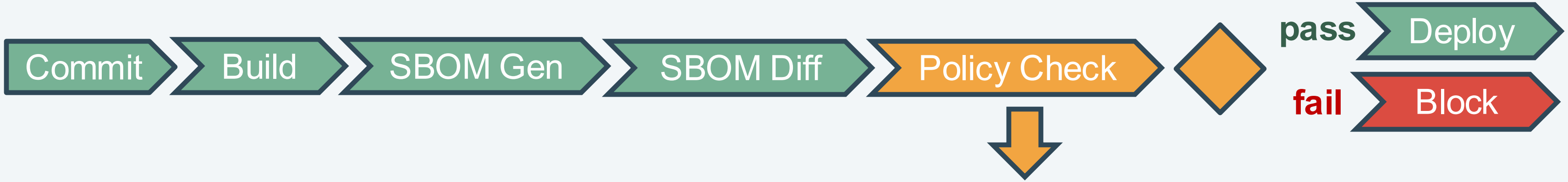
Mitigating Risk with Automated Impact Analysis

The Old Way (Manual):
- A component changes.
- Safety engineer spends 3 days tracing dependencies, reading docs, and flagging tests.

The New Way (Automated):
- **SBOM v1 vs. v2 diff** Identifies: `component-x` changed.
- **Platform Query** Determines: Component `is ASIL-D` and traces to `SafetyGoal-001`.
- **Result**: Automatically flag `FMEA-456` for re-check and trigger `TestSuite-123`.

Powered by: SBOM Versioning, Automated Diffing, and Multi-hop Traceability.

Enabling Continuous Compliance via SBOM Policies



ASIL-D components MUST NOT have any **CRITICAL vulnerabilities** unless linked **VEX** record says **“not affected.”**

Every component marked **safetyRelevant MUST** link to **at least one requirement document**

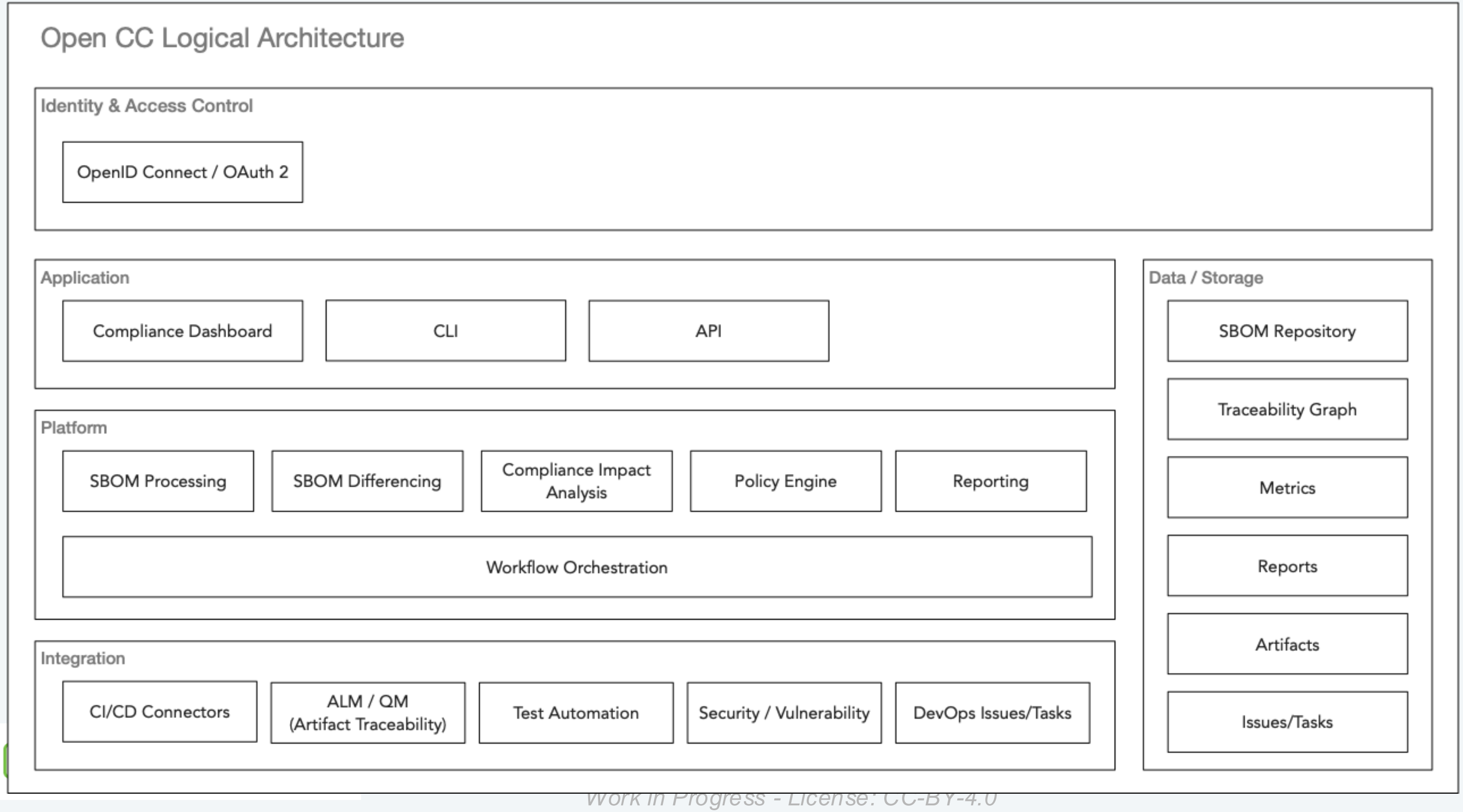
Components at **ASIL-B or higher MUST** link to **both** a **test case and** a **verification artifact**

Legacy components MUST NOT be reused in safety-critical scope unless an **Exception Justification** is linked

Every recorded **hazard MUST** trace to at least one **mitigation** in design or fault-detection docs.

Safety-relevant components MUST have a **complete SBOM**—no **NOASSERTION/UNKNOWN** for license, version, or supplier.

Foundations of a Continuous Compliance Platform



Open CC Logical Architecture

Identity & Access Control

- OpenID Connect / OAuth 2

Application

- Compliance Dashboard
- CLI
- API

Platform

- SBOM Processing
- SBOM Differencing
- Compliance Impact Analysis
- Policy Engine
- Reporting
- Workflow Orchestration

Integration

- CI/CD Connectors
- ALM / QM (Artifact Traceability)
- Test Automation
- Security / Vulnerability
- DevOps Issues/Tasks

Data / Storage

- SBOM Repository
- Traceability Graph
- Metrics
- Reports
- Artifacts
- Issues/Tasks

Current State: Alpha 1

spdx-diff: High-Performance Change Detection

- Semantic comparison of Software Bills of Materials (SBOMs)
- JSON-LD Context and IRI-aware comparison
- Supports all SPDX 3.0.1 profiles and extensible to support SPDX extensions

spdx-impact: Multi-Domain Compliance Analysis

- Analyzes Safety, Security, and Privacy impacts simultaneously
- Identifies which evidence must be re-verified
- Specifies which tests must be re-executed
- Traces transitive impacts through dependency chains
- YAML-based policy engine
- Highly configurable and extensible to support additional compliance domains

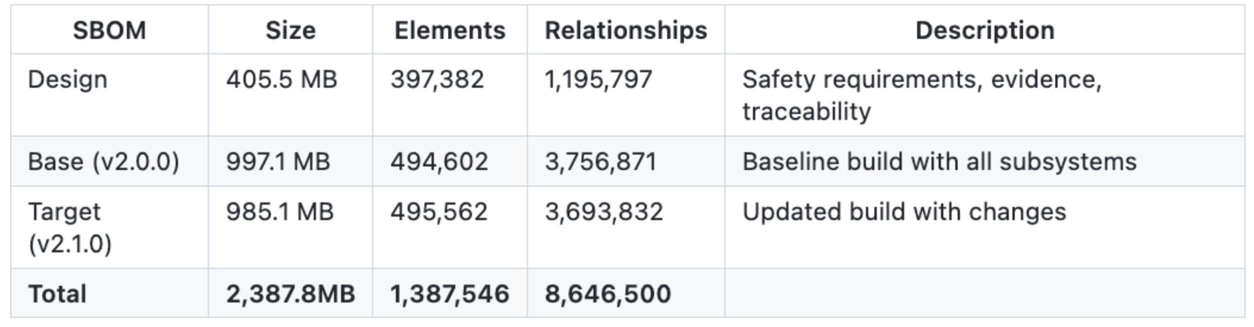
cc-audit: Regulator-Ready Audit Logging

- Blockchain-style cryptographic hash chains for tamper evidence
- Complete provenance tracking (inputs, versions, environment, checksums)
- Deterministic execution mode for reproducible builds
- SIEM integration for security monitoring

Synthesized Autonomous Vehicle SBOMs

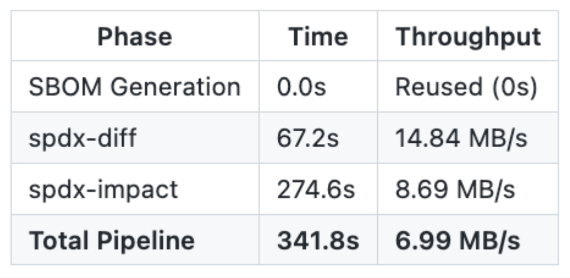
1. Perception - Camera, LiDAR, radar, object detection (ASIL-D)
2. Localization - GPS, IMU, SLAM (ASIL-D)
3. Planning - Route, behavioral, motion planning (ASIL-D)
4. Control - Steering, throttle, brake controllers (ASIL-D)
5. Safety - Health monitoring, fail-safe systems (ASIL-D)
6. Communication - CAN, Ethernet, V2X (ASIL-B)
7. Middleware - ROS2, DDS (ASIL-B)
8. Operating System - Linux kernel, drivers (ASIL-D)
9. ML/AI - Perception, prediction models (ASIL-B)
10. Security - Secure boot, encryption, IDS (ASIL-D)
11. Third-Party - Libraries and dependencies (QM)
12. Diagnostics - Logging, OTA updates (ASIL-A)

Performance Results



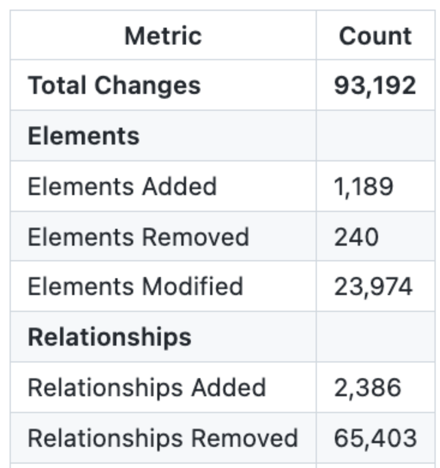
SBOM	Size	Elements	Relationships	Description
Design	405.5 MB	397,382	1,195,797	Safety requirements, evidence, traceability
Base (v2.0.0)	997.1 MB	494,602	3,756,871	Baseline build with all subsystems
Target (v2.1.0)	985.1 MB	495,562	3,693,832	Updated build with changes
Total	**2,387.8MB**	**1,387,546**	**8,646,500**	

Benchmarks



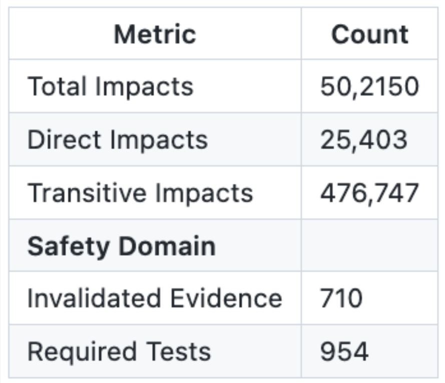
Phase	Time	Throughput
SBOM Generation	0.0s	Reused (0s)
spdx-diff	67.2s	14.84 MB/s
spdx-impact	274.6s	8.69 MB/s
Total Pipeline	**341.8s**	**6.99 MB/s**

spdx-diff



Metric	Count
Total Changes	**93,192**
Elements	
Elements Added	1,189
Elements Removed	240
Elements Modified	23,974
Relationships	
Relationships Added	2,386
Relationships Removed	65,403

spdx-impact



Metric	Count
Total Impacts	50,2150
Direct Impacts	25,403
Transitive Impacts	476,747
Safety Domain	
Invalidated Evidence	710
Required Tests	954

Immediate Next Steps

Use Case Buildout

Implement full end-to-end CI/CD pipeline for a representative project.

Solicit Community Feedback

Engage and gather feedback from key industry and community stakeholders. (THIS MEANS YOU)

Expand Policy Engine

Expand rule engine features and bundled policies.

SPDX 3.1 Alignment

Align to official Safety Profile and other SPDX 3.1 additions & enhancements.

Mid-Term Goals

Build Compliance Dashboard

Create a website front-end that supports configuration, reporting, and advanced analytics.

Add Integrations

Build-out integrations with ALM/QM, Security/Vulnerability, and DevOps platforms.

Expand Policies & Compliance Domains

Add and improve the compliance policy bundles.

The Frontier: AI-Enhanced Safety Compliance

Developer Copilot for Safety: Real-time, MISRA-aware code generation and compliance checks directly in the IDE and SWE-agents based on MCP/A2A

Predictive Risk Assessment: AI models that forecast component failures and compliance risks before they happen.

Automated Traceability Discovery: AI that reads documentation and code to automatically suggest and maintain traceability links.

Natural Language Policy Engine: Define and query complex safety policies using plain English, e.g., "Show me all ASIL-D components with failing tests."

Thank You

InfoMagnus

LinkedIn