Building Dependable Systems with Open Source

Kate Stewart, The Linux Foundation
Modern products are more than just hardware and software
“Ingredients” for a Modern Car

• Hardware
  – Traditional BOM, but with more CPUs, MCUs & GPUs incorporated

• Software
  – Managing interaction between sensors, actuators, humans & environment
  – Managing trained AI/ML models that assist in the safe & efficient operation of the vehicle

• Training Data Sets
  – Data used to train, test & validate the AI/ML models in use the system

• Communication to Remote Services
  – External environment awareness for navigation support
  – Updates to the software, firmware & models
We need to leverage a System Engineering approach to manage risk from the interactions of all these ingredients.
17 fatalities, 736 crashes: The shocking toll of Tesla’s Autopilot

Washington Post - 2023

San Francisco — The school bus was displaying its stop sign and flashing red warning lights, a police report said, when Tillman Mitchell, 17, stepped off one afternoon in March. Then a Tesla Model Y approached on North Carolina Highway 561.

The car — allegedly in Autopilot mode — never slowed down.

It struck Mitchell at 45 mph. The teenager was thrown into the windshield, flew into the air and landed.

Show Details on Incident #550

---

Tesla’s “Full Self-Driving” sees pedestrian, chooses not to slow down

Arstechnica - 2023

Tesla released a new version of its controversial “Full Self-Driving Beta” software last month. Among the updates in version 11.4 are new algorithms determining the car’s behavior around pedestrians. But alarmingly, a video posted to Twitter over the weekend shows that although the Tesla system can see pedestrians crossing the road, a Tesla can choose not to stop or even slow down as it drives past.

Show Details on Incident #540

---

Auto-Safety Regulators Investigate Cruise’s Self-Driving Cars Over Pedestrian Risks

WSJ - 2023

General Motors’ driverless-car unit Cruise is confronting a new safety investigation by federal regulators, after reports of its autonomous vehicles exhibiting risky behavior around pedestrians.

The National Highway Traffic Safety Administration said in a Tuesday filing that it had opened a safety-defect probe into nearly 600 driverless cars operated by Cruise, adding that they might not be exercised at all.

Show Details on Incident #596

---

Tesla Recalls 362,758 Vehicles Due to FSD Crash Risk

ExtremeTech - 2023

Tesla is recalling 362,758 of its vehicles due to crash risks associated with its autonomous driving software, referred to as Full Self Driving (FSD) Beta. The recall was announced via the National Highway Traffic Safety Administration (NHTSA) website Thursday. According to Tesla’s notice, some 2016-2023 Model S, Model X, 2017-2023 Model 3, and 2020-2023 Model Y vehicles with FSD Beta installed are included.

Show Details on Incident #478

Source: https://incidentdatabase.ai/
17 fatalities, 736 crashes: The shocking toll of Tesla’s Autopilot

Tesla’s driver-assistance system, known as Autopilot, has been involved in far more crashes than previously reported

By Fakir Siddiqui and Jeremy B. Mentill
June 10, 2023 at 7:00 am EDT

source: https://www.washingtonpost.com/technology/2023/06/10/tesla-autopilot-crashes-elon-musk/
SAN FRANCISCO — The school bus was displaying its stop sign and flashing red warning lights, a police report said, when Tillman Mitchell, 17, stepped off one afternoon in March. Then a Tesla Model Y approached on North Carolina Highway 561.

The car — allegedly in Autopilot mode — never slowed down.
SAN FRANCISCO — The school bus was displaying its stop sign and flashing red warning lights, a police report said, when Tillman Mitchell, 17, stepped off one afternoon in March. Then a Tesla Model Y approached on North Carolina Highway 561.

The car — allegedly in Autopilot mode — never slowed down.

Tesla’s 17 fatal crashes reveal distinct patterns, The Post found: Four involved a motorcycle. Another involved an emergency vehicle. Meanwhile, some of Musk’s decisions — such as widely expanding the availability of the features and stripping the vehicles of radar sensors — appear to have contributed to the reported uptick in incidents, according to experts who spoke with The Post.

source: https://www.washingtonpost.com/technology/2023/06/10/tesla-autopilot-crashes-elon-musk/
17 fatalities, 736 crashes: The shocking toll of Tesla’s Autopilot

SAN FRANCISCO — The school bus was displaying its stop sign and flashing red warning lights, a police report said, when Tillman Mitchell, 17, stepped off one afternoon in March. Then a Tesla Model Y approached on North Carolina Highway 561.

The car — allegedly in Autopilot mode — never slowed down.

Autopilot, which Tesla introduced in 2014, is a suite of features enabling the car to maneuver itself from highway on-ramp to off-ramp, maintaining speed and distance behind other vehicles and following lane lines. Tesla offers it as a standard feature on its vehicles, of which more than 800,000 are equipped with Autopilot on U.S. roads, though advanced iterations come at a cost.

Tesla’s 17 fatal crashes reveal distinct patterns, The Post found: Four involved a motorcycle. Another involved an emergency vehicle. Meanwhile, some of Musk’s decisions — such as widely expanding the availability of the features and stripping the vehicles of radar sensors — appear to have contributed to the reported uptick in incidents, according to experts who spoke with The Post.

source: https://www.washingtonpost.com/technology/2023/06/10/tesla-autopilot-crashes-elon-musk/
SAN FRANCISCO — The school bus was displaying its stop sign and flashing red warning lights, a police report said, when Tillman Mitchell, 17, stepped off one afternoon in March. Then a Tesla Model Y approached on North Carolina Highway 561.

The car — allegedly in Autopilot mode — never slowed down.

Autopilot, which Tesla introduced in 2014, is a suite of features enabling the car to maneuver itself from highway on-ramp to off-ramp, maintaining speed and distance behind other vehicles and following lane lines. Tesla offers it as a standard feature on its vehicles, of which more than 800,000 are equipped with Autopilot on U.S. roads, though advanced iterations come at a cost.

Tesla’s 17 fatal crashes reveal distinct patterns, The Post found: Four involved a motorcycle. Another involved an emergency vehicle. Meanwhile, some of Musk’s decisions — such as widely expanding the availability of the features and stripping the vehicles of radar sensors — appear to have contributed to the reported uptick in incidents, according to experts who spoke with The Post.

In February, Tesla issued a recall of more than 360,000 vehicles equipped with Full Self-Driving over concerns that the software prompted its vehicles to disobey traffic lights, stop signs and speed limits.

source: https://www.washingtonpost.com/technology/2023/06/10/tesla-autopilot-crashes-elon-musk/
17 fatalities, 736 crashes: The shocking toll of Tesla’s Autopilot

Tesla’s driver-assistance system, known as Autopilot, has been involved in far more crashes than previously reported.

By Felix Sallieul and Jeremy Benner
June 10, 2023 at 7:00 am

The uptick in crashes coincides with Tesla’s aggressive rollout of Full Self-Driving, which has expanded from about 12,000 users to nearly 400,000 in a little more than a year. Nearly two-thirds of all driver-assistance crashes that Tesla has reported to NHTSA occurred in the past year.

source: https://www.washingtonpost.com/technology/2023/06/10/tesla-autopilot-crashes-elon-musk/
17 fatalities, 736 crashes: The shocking toll of Tesla’s Autopilot

Tesla’s driver-assistance system, known as Autopilot, has been involved in far more crashes than previously reported.

By Faisal Siddiqi and Jeremy Berke
June 10, 2023 at 7:05 am

The uptick in crashes coincides with Tesla’s aggressive rollout of Full Self-Driving, which has expanded from about 12,000 users to nearly 400,000 in a little more than a year. Nearly two-thirds of all driver-assistance crashes that Tesla has reported to NHTSA occurred in the past year.

Source: National Highway Traffic Safety Administration

source: https://www.washingtonpost.com/technology/2023/06/10/tesla-autopilot-crashes-elon-musk/
# 17 fatalities, 736 crashes: The shocking toll of Tesla’s Autopilot

Tesla’s driver-assistance system, known as Autopilot, has been involved in far more crashes than previously reported.

By Faiz Siddiqui and Jeremy Scahill
June 10, 2023 at 7:05 am

## Crashes involving Tesla’s driver assistance system have grown

Tesla’s “Full Self-Driving” and Autopilot systems have been involved in far more incidents than driver-assistance systems from all other manufacturers combined.

<table>
<thead>
<tr>
<th>Year</th>
<th>Tesla</th>
<th>Other Makes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021Q1</td>
<td>30</td>
<td>50</td>
</tr>
<tr>
<td>2021Q2</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>2021Q3</td>
<td>50</td>
<td>70</td>
</tr>
<tr>
<td>2021Q4</td>
<td>60</td>
<td>80</td>
</tr>
<tr>
<td>2022Q1</td>
<td>70</td>
<td>90</td>
</tr>
<tr>
<td>2022Q2</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>2022Q3</td>
<td>90</td>
<td>110</td>
</tr>
<tr>
<td>2022Q4</td>
<td>100</td>
<td>120</td>
</tr>
<tr>
<td>2023Q1</td>
<td>110</td>
<td>130</td>
</tr>
</tbody>
</table>

Complete data for 2023Q2 is not yet available. A small number of incidents from 2019 and 2020 are not included.

Source: National Highway Traffic Safety Administration

The uptick in crashes coincides with Tesla’s aggressive rollout of Full Self-Driving, which has expanded from about 12,000 users to nearly 400,000 in a little more than a year. Nearly two-thirds of all driver-assistance crashes that Tesla has reported to NHTSA occurred in the past year.

While Tesla has constantly tweaked its driver-assistance software, it also took the unprecedented step of eliminating radar sensors from new cars and disabling them from vehicles already on the road — depriving them of a critical sensor as Musk pushed a simpler hardware set amid the global computer chip shortage. Musk said last year, “Only very high resolution radar is relevant.”

The company has recently taken steps to reintroduce radar sensors, according to government filings first reported by Electrek.

More Ingredients ⇒ More Ways Can Go Wrong

• Software Vulnerabilities
  – Interaction between proprietary and open source components in system
  – Assessment if a mitigation needs to be applied to an incorporated image or not.

• Hazards from AI/ML model
  – Biases in training data sets
  – Interaction issues after update of model and with other software on system

• Training Data Sets
  – Data used to train, test & validate the AI/ML models in use the system

• Remote Services
  – External Environment awareness for navigation support
  – Software & model updates
We need to expand from **Software BOM ⇒ System BOM** in tracking dependencies between the “ingredients” especially when there are **safety elements**.
Standardized Metadata is Needed from the Supply Chains

All supply chains contributing “ingredients” (hardware, software, data sets, services) need to provide metadata in a standard format, so risk can be accurately assessed and managed.

• What software component versions are executing on which specific hardware devices (and/or models, and/or simulators/FPGAs)?
• What software components direct and transitive dependencies should be monitored for vulnerabilities?
• What is the provenance of how a model was trained? What datasets were used for testing and validation?
• How were the datasets used for training created? Are there known biases?
• How were the software components and models integrated and tested?
• What APIs are used to manage updates through remote services?
• What remote services does the running software and trained models depend on? What happens when the service is not available?
• How tracking updates to software, model, data sets in a product line, so current picture at any point in time?
Standardized Metadata Needs to be Accurate

From all supply chains (hardware, software, datasets, services) the standard format should:

- **Capture the data when it is created** in the product’s lifecycle
  - Design - system requirements, plans, processes
  - Source - source files, make scripts, build processes, test files, …
  - Build - built applications, libraries, firmware, build configuration, …
  - Deploy - application configuration information, installed dependencies, validation,…
  - Runtime - system configuration information, …

- **Assemble the facts into knowledge** about the system and it’s intended behavior
  - Use relationships to link between facts about each component
  - Create knowledge graph to represent product line at any point in time including requirements, sources, tests, and evidence that the requirement are satisfied.
Essential for Critical Infrastructure to have information, too!

Critical Infrastructure

Since 2005, the ‘Cybersecurity Policy for Critical Infrastructure Protection’ has been set as a common action plan shared between the government, which bears responsibility for promoting independent measures by CI operators relating to CI cybersecurity and implementing other necessary measures, and CI operators which independently carry out relevant protective measures, and the new edition was published in 2022.

This document identifies the 14 sectors as critical infrastructure and it expects stakeholders to undertake the five measures as below.

1. Enhancement of Incident Response Capability
2. Maintenance and Promotion of the Safety Principles
3. Enhancement of Information Sharing System
4. Utilization of Risk Management
5. Enhancement of the Basis for CIP

The Cybersecurity Policy for Critical Infrastructure Protection

- [Full Text](https://www.nisc.go.jp/eng/index.html#sec4)
- [Risk Assessment Guide Based on the Concept of Mission Assurance in Critical Infrastructure (1st Edition) (Revised on May 2019)](#)
Connecting a Product’s Supply Chain MetaData

Database containing all product line component metadata, the relationships between components, requirements and evidence.
Evolving **SPDX profiles** to provide the framework for connecting metadata about components, processes, requirements and evidence to support **product line management**.
SPDX Evolution

**SPDX 2.2+** *(ISO/IEC 5962:2021)* supports exchanging metadata between systems

- Software BOM metadata and relationships between components.
- Supports traceability between requirements, code, tests & evidence

**SPDX 3.0** to support the databases more efficiently

- Introduces profiles to capture domain specific metadata about components and their interactions at points in time
- Extends beyond software to capture AI/ML model and dataset provenance
- Supports product lifecycle metadata and incorporation of updates to remediate vulnerabilities
- Import from suppliers and export to customers current state at point in time

**SPDX 3.1** extend beyond software to support safety profile needs for “all ingredients”

- Work already in progress on Hardware, Services and Safety Profiles
SPDX 3.0 Profiles

- **SECURITY**: Security information - vulnerability details related to software
- **BUILD**: Build related information - provenance and reproducible builds
- **AI**: Information about AI models - ethical, security, and model data
- **DATA**: Information about datasets - AI and other data use cases
- **LITE**: Minimal subset to support industry supply chain workflows
- **LICENSING**: Information about copyrights and licenses - supports compliance
- **SOFTWARE**: Information specific to software
- **CORE**: Information used across all profiles
Support generating SBOMs when the facts are known

- Source SBOM
- Design SBOM
- Build SBOM
- Deployed SBOM
- Runtime SBOM
## Align with the SBOM Types from CISA

<table>
<thead>
<tr>
<th>SBOM TYPE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>SBOM of intended, planned software project or product with included components (some of which may not yet exist) for a new software artifact.</td>
</tr>
<tr>
<td>Source</td>
<td>SBOM created directly from the development environment, source files, and included dependencies used to build an product artifact.</td>
</tr>
<tr>
<td>Build</td>
<td>SBOM generated as part of the process of building the software to create a releasable artifact (e.g., executable or package) from data such as source files, dependencies, built components, build process ephemeral data, and other SBOMs.</td>
</tr>
<tr>
<td>Deployed</td>
<td>SBOM provides an inventory of software that is present on a system. This may be an assembly of other SBOMs that combines analysis of configuration options, and examination of execution behavior in a (potentially simulated) deployment environment.</td>
</tr>
<tr>
<td>Runtime</td>
<td>BOM generated through instrumenting the system running the software, to capture only components present in the system, as well as external call-outs or dynamically loaded components. In some contexts, this may also be referred to as an “Instrumented” or “Dynamic” SBOM.</td>
</tr>
<tr>
<td>Analyzed</td>
<td>SBOM generated through analysis of artifacts (e.g., executables, packages, containers, and virtual machine images) after its build. Such analysis generally requires a variety of heuristics. In some contexts, this may also be referred to as a “3rd party” SBOM.</td>
</tr>
</tbody>
</table>

Source: [Types of Software Bills of Materials (SBOM) published by CISA](https://www.cisa.gov/types-software-bills-materials) on 2023/4/21
# SPDX 2.3 Relationships to Clarify Dependencies

For more details see: [https://spdx.github.io/spdx-spec/v2.3/relationships-between-SPDX-elements/](https://spdx.github.io/spdx-spec/v2.3/relationships-between-SPDX-elements/)

<table>
<thead>
<tr>
<th>RELATIONSHIP</th>
<th>DESCRIPTION</th>
<th>PREREQUISITE FOR</th>
<th>GENERATES</th>
<th>VARIANT_OF</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIBES</td>
<td>DEPENDENCY_OF</td>
<td>PREREQUISITE_FOR</td>
<td>GENERATES</td>
<td>VARIANT_OF</td>
</tr>
<tr>
<td>DESCRIBED_BY</td>
<td>RUNTIME_DEPENDENCY_OF</td>
<td>HAS_PREREQUISITE</td>
<td>TEST_OF</td>
<td>FILE_ADDED</td>
</tr>
<tr>
<td>CONTAINS</td>
<td>BUILD_DEPENDENCY_OF</td>
<td>ANCESTOR_OF</td>
<td>TEST_TOOL_OF</td>
<td>FILE_DELETED</td>
</tr>
<tr>
<td>CONTAINED_BY</td>
<td>DEV_DEPENDENCY_OF</td>
<td>DESCENDENT_OF</td>
<td>TEST_CASE_OF</td>
<td>FILE_MODIFIED</td>
</tr>
<tr>
<td>DYNAMIC_LINK</td>
<td>OPTIONAL_DEPENDENCY_OF</td>
<td>DOCUMENTATION_OF</td>
<td>EXAMPLE_OF</td>
<td>PATCH_FOR</td>
</tr>
<tr>
<td>STATIC_LINK</td>
<td>PROVIDED_DEPENDENCY_OF</td>
<td>BUILD_TOOL_OF</td>
<td>METAFILE_OF</td>
<td>PATCH_APPLIED</td>
</tr>
<tr>
<td>AMENDS</td>
<td>TEST_DEPENDENCY_OF</td>
<td>EXPANDED_FROM_ARCHIVE</td>
<td>PACKAGE_OF</td>
<td>REQUIREMENT_FOR</td>
</tr>
<tr>
<td>COPY_OF</td>
<td>OPTIONAL_COMPONENT_OF</td>
<td>DISTRIBUTION_ARTIFACT</td>
<td>DATA_FILE_OF</td>
<td>SPECIFICATION_FOR</td>
</tr>
<tr>
<td>DEPENDS_ON</td>
<td>DEPENDENCY_MANIFEST_OF</td>
<td>GENERATED_FROM</td>
<td>DEV_TOOL_OF</td>
<td>OTHER</td>
</tr>
</tbody>
</table>
SPDX component modularity and relationships between components, allows us to create the knowledge graph for accurate and efficient Safety & Security Analysis.
## Manage Safety Artifacts with SBOMs

<table>
<thead>
<tr>
<th>SBOM Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design SBOM</strong></td>
<td>Functional Safety Management (Plans) and Safety Concept</td>
</tr>
<tr>
<td><strong>Source SBOM</strong></td>
<td>Requirements, Design, Safety Analysis, Source Code, Test Cases</td>
</tr>
<tr>
<td><strong>Build SBOM</strong></td>
<td>Build Framework, Build configuration and environment data, Test Framework, Executable, Test Reports</td>
</tr>
<tr>
<td><strong>Deploy SBOM</strong></td>
<td>Deployed configuration and environment data, Hardware architecture specific information and data, deployment tests and reports</td>
</tr>
<tr>
<td><strong>Runtime SBOM</strong></td>
<td>Runtime relevant data (configuration data), training data, error logging data</td>
</tr>
</tbody>
</table>
Design SBOM to Source SBOM

- **Zephyr Safety Dev Plan**
- **Zephyr Requirements Management Plan**
- **Zephyr Verification Plan**
- **Zephyr Configuration & Change Management Plan**
- **Software Requirements Specifications**
- **Coding Guidelines**
- **Source Code**
- **Component Tests**

Specifications:
- Specification file, requirements, architecture
- Source file
- Tests, test scripts, verification
- Evidence, reports
- Plans, Guidelines, Process

Evidences:
- Code review (Static Analysis)
- Static analysis scan reports
- Component test reports

Planned:
- Requirement for:
  - Software Component Design Specifications
  - Component Tests
Source SBOM to Build SBOM
Dependency Identification between Components

REQUIREMENT_FOR
Specification file, requirements, architecture

SPECIFICATION_FOR
Zephyr Verification Plan

SPECIFICATION_FOR
Zephyr Configuration & Change Management Plan

SPECIFICATION_FOR
Software Component Design Specifications

SPECIFICATION_FOR
Coding Guidelines

TEST_OF
Source Code

TEST_OF
Component Tests

GENERATES
Executable image

SPECIFICATION_FOR
Software Build Chain Specification

SPECIFICATION_FOR
Integr. Test Framework Specification

TEST_OF
Software Tests

??
Software Tests

??
Executive image

??
Executable image

??
Source file

??
Tests, test scripts, verification

??
Evidence, reports

??
Plans, Guidelines, Process
Dependency Identification at Component Level

- REQUIREMENT_FOR
  - Source file
  - Tests, test scripts, verification
  - Evidence, reports
  - Plans, guidelines, process
  - Executable image

- SPECIFICATION_FOR
  - Software Requirements Specification
  - Software Build Chain Specification
  - Integr. Test Framework Specification

- GENERATES
  - Executable image

- TEST_OF
  - Component Tests
  - Code review (Static Analysis)
  - Source Code

- REQUIREMENT_FOR
  - Software Component Design Specifications
  - Zephyr Configuration & Change Management Plan
  - Zephyr Verification Plan

- LICENSED_UNDER
  - CC-BY-SA-3.0

- SPDX

- SAFETY
Dependency Identification at Component Level

** REQUIREMENT FOR **

- Specification file
- Requirements
- Architecture

** source file <-> Tests, test scripts, verification

** Evidence, reports

** Plans, Guidelines, Process

** Executable image

** SPECIFICATION FOR **

- Coding Guidelines
- Code review (Static Analysis)

** TEST OF **

- Source Code
- Component Tests
- Executable image

** GENERATES **

- Software Tests
- Executable image

** SPECIFICATION FOR **

- Software Build Chain Specification
- Integr. Test Framework Specification

** REQUIREMENT FOR **

- Zephyr Verification Plan
- Zephyr Configuration & Change Management Plan

** SOFTWARE REQUIREMENTS SPECIFICATION **

- (Software Requirements Specification)
Dependency Identification at Component Level

**SPECIFICATION**

- **Source file**
- **Tests, test scripts, verification**
- **Evidence, reports**
- **Plans, Guidelines, Process**
- **Executable image**

**TEST**

- **Software Build Chain Specification**
- **Test Framework Specification**
- **Executable image**

**GENERATES**

- **Software Tests**

**SPECIFICATION**

- **Software Requirements Specification**
- **Component Tests**

**REQUIREMENT**

- **Software Component Design Specifications**

**Coding Guidelines**

- **Code review (Static Analysis)**

**SPDX SAFETY**

Licensed under CC-BY-SA-3.0
Dependency Identification at Component Level

- **Zephyr Safety Dev Plan**
  - **Zephyr Requirements Management Plan**
  - **Zephyr Verification Plan**
  - **Zephyr Configuration & Change Management Plan**

- **Software Requirements Specifications**

- **Coding Guidelines**
  - **Code review**
  - **(Static Analysis)**
  - **Specification file, requirements, architecture**
  - **Tests, test scripts, verification**
  - **Evidence, reports**
  - **Plans, Guidelines, Process**

- **Source Code**

- **Component Tests**
  - **Component test reports**
  - **Static analysis scan reports**

- **Software Component Design Specifications**

- **Software Requirements Specifications**

- **Component Tests**

License: CC-BY-SA-3.0
Component Level Requirements Traceability
When needed: Traceability Inside Component

Requirement to Code to Tests to Evidence

- Requirement A.1
- foo.c
- make
- A.1.1 test
- A.1.2 test
- A.1.3 test

Specifications file, requirements, architecture

Source file

Tests, test scripts

Evidence, reports

Test framework

Log from A.1.1 test
Log from A.1.2 test
Log from A.1.3 test
When needed: Traceability Inside Component
 Requirement to Code to Tests to Evidence

- Requirement A.1
  - foo.c
  - make

- A.1.1 test
- A.1.2 test
- A.1.3 test

- Log from A.1.1 test
- Log from A.1.2 test
- Log from A.1.3 test

- Test framework
- Test framework
- Test framework

- Specification file, requirements, architecture
- source file
- Tests, test scripts
- Evidence, reports
- Bug Fix

Licensed under CC-BY-SA-3.0
Traceability Inside Component
New Requirement to Code to Tests to Evidence

Requirement A.1

New Requirement From Impact Analysis

foo.c

make

A.1.1 test

A.1.2 test

A.1.3 test

NR test

Test framework

GENERATES

EVIDENCE_FOR

Log from A.1.1 test

Log from A.1.2 test

Log from A.1.3 test

Log from NR test

Specification file, requirements, architecture

source file

Test scripts

Evidence, reports

Bug Fix

Requirement

New Requirement

Requirement

Requirement

Requirement
Inside Component: Traceability of Source to Requirements
Code to Requirements to Tests to Evidence

- **foo.c**: Source file
- **make**: Command
- **Requirement A.1**: Requirement
- **B.1**: Requirement
- **Test framework**: Log from A.1.1 test
- **Log from A.1.2 test**
- **Log from A.1.3 test**
- **Log from B.1.1 test**
- **Log from B.1.2 test**
- **Log from B.1.3 test**

**Specifications**:
- Specification file, requirements, architecture
- Source file
- Executable image
- Tests, test scripts
- Evidence, reports
- Bug Fix

**Traceability Map**:
- Code to Requirements
- Requirements to Tests
- Tests to Evidence

Licensed under CC-BY-SA-3.0
How can we establish “Requirements” for Open Source Components that System Engineering & Safety Analysis need?
Open Source Projects working to Support Functional Safety

Linux:
- ELISA

RTOS:
- Zephyr

Virtualization/Hypervisor:
- Xen Project

Reproducible Build Framework:
- Yocto Project
The Yocto Project:
It’s not an embedded Linux distribution, it creates a custom one for you!

The de facto industry standard “tool kit”
The de facto industry standard “tool kit” for building custom embedded Linux operating systems

Preferred platform for a variety of industry initiatives
AGL, RDK Set Top Boxes, TVs, Commercial Switches, Routers, Security Products, Embedded Devices, Medical Devices, and much more

The #1 platform to validate new SoC designs
(all architectures) and build BSPsSBOMs and Reproducible Builds

Maintained by a highly skilled, small team
We are always looking for contributors and members.
Yocto Support

Today:
- Reproducible binaries are supported
- Yocto generates SPDX SBOMs for the build toolchain & all components built by that toolchain, to source level today, by a single configuration change
- System view is done by a master index (for UUID) today.
- Participated in creation of SPDX Build profile to capture key data

Work in Progress:
- Product Line System BOM generation with SPDX
- Linkage PTEST results with some components: Lot of test data.

Any feature enabled by support in Yocto can scale throughout it’s ecosystem
ENABLING LINUX IN SAFETY APPLICATIONS
Project Goals

- Support safety certification of Linux-based systems with a set of elements, processes and tools.
- Enable companies to incorporate the output of the project into products.
- The work is accepted by the open source community, safety community, regulation authorities, standardization bodies and system developers.
- Focus the project activities using a Linux-based reference system to safety-integrity standards.
Systems

Goal(s):

“Enable other working groups within ELISA to put their safety claims towards Linux in a wider system context.”

Activities:

● Provide a reproducible reference system based on real world architectures.
● Reference system fully automated and fully based on Open-Source technologies.
● Interactions with other OSS projects with relevance to mixed-criticality system elements.

In practice:

● Working on systems to connect Linux with hypervisor and RTOS & explore implications of OSS projects interacting mixed criticality systems.
● First one shown during OSS NA - illustrating Linux, Xen & Zephyr interacting. Enhancement with AGL Linux in progress. SPDX prototyping.
Systems group integrates ELISA working groups

- **Linux Features, Architecture** and **Code Improvements** should be integrated into the reference system directly.

- **Tools** and **Engineering process** should fit the reproducible product creation.

- **Medical, Automotive** and future WG use cases should be able to strip down the reference system to their use case demands.
New Requirements Tool: **BASIL** Open Sourced

Learn more at: [https://elisa.tech/blog/2023/11/30/basil-the-fusa-spice/](https://elisa.tech/blog/2023/11/30/basil-the-fusa-spice/)
Contribute to the code at: [https://github.com/elisa-tech/BASIL](https://github.com/elisa-tech/BASIL)
Zephyr Project

- **Open source** real time operating system
- **Developer friendly** with vibrant community participation
- Built with **safety and security** in mind
- **Broad SoC, board and sensor support.**
- **Vendor Neutral** governance
- **Permissively licensed** - Apache 2.0
- **Complete**, fully integrated, highly configurable, **modular** for **flexibility**
- **Product** development ready using LTS includes **security updates**
- **Certification** ready with Zephyr Auditable

Open Source, RTOS, Connected, Embedded Fits where Linux is too big

Zephyr OS
- 3rd Party Libraries
- Application Services
- OS Services
- Kernel
- HAL

THE LINUX FOUNDATION PROJECTS
Safety: Initial certification focus

- Start with a limited scope of kernel and interfaces
- Initial target is IEC 61508 SIL 3 / SC 3 (IEC 61508-3, 7.4.2.12, Route 3s)
- Option for 26262 certification has been included in contract with certification authority should there be sufficient member interest

Scope can be extended to include additional components with associated requirements and traceability as determined by the safety committee.
Current requirements work

- Used tooling: StrictDoc (https://github.com/strictdoc-project/strictdoc)

- Decision on UIDs for requirements (UID will be generated by StrictDoc)

- Hierarchical structure of requirements that works for the project

- Capturing the requirements in StrictDoc which is working towards import/export of SPDX
Mission Statement

THE MISSION OF THE XEN PROJECT IS TO ADVANCE VIRTUALIZATION TECHNOLOGY ACROSS A WIDE RANGE OF COMMERCIAL AND OPEN-SOURCE DOMAINS.

BY PROVIDING A POWERFUL AND VERSATILE HYPervisor, THE PROJECT AIMS TO ENABLE INNOVATION, SCALABILITY, SAFETY, AND SECURITY IN VIRTUALIZATION SOLUTIONS.
The Xen Project

- What is it?
  - Xen is a Type-1 hypervisor that plays a central role in providing isolation between different software components

- The history of Xen
  - The project started in 2003 from Cambridge University
  - Became a Linux Foundation project in 2013
  - It’s widely used for its safety and security first environments
  - The flexible architecture allows for diverse applications and service needs to coexist on the same hardware

- Open source project
  - Many subprojects: Hypervisor, Windows PV, XAPI, automotive etc
  - Intel and AMD x86 and ARM already supported
  - Diverse community of maintainers and contributors from Amazon, SUSE, XenServer (formerly Citrix) and more
Xen Support

Today:
- Xen is chosen for safety critical applications due to its maturity and robust security features
- Can be configured to provide real-time scheduling for VMs
- Allows critical tasks to run within predefined time constraints

Work in Progress:
- Improve Xen coding style with MISRA-C
- Implement features to improve real-time and reduce interference
- Project members working on getting Xen safety certified for 61508 & 26262
Next steps to continue the discussion?

Augmenting open source components:

**Wednesday**, December 6 · 15:05 - 15:45 · Conference Room 1

- **Linux**: join in [ELISA working groups](#)
- **Zephyr**: join in the [safety](#) working group
- **Xen**: join the [FuSa](#) special interest group
- **Yocto**: join the build & release communities

Framework for connecting “All the Ingredients”:

- **SPDX**: join the Functional Safety(FuSa) profile group [meetings](#) and/or [mailing list](#)
Integrating Open Source efficiently into System Engineering practices is overdue, community required.

Hint: don’t expect upstream project maintainers to take the lead here.
THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

JAPAN
Abstract: By looking at the press headlines, we've learned that open source is already being used in market segments (like space, automotive, industrial, medical, agricultural) applications that have safety considerations today. Details about the safety analysis performed are behind NDAs and are not available to developers in the open source projects being used in these products. To make the challenge even more interesting, the processes the safety standards are expecting are behind paywalls, and not readily accessible to the wider open source community maintainers and developers. Figuring out pragmatic steps to adopt in open source projects requires the safety assessor communities, the product creators, and open source developers to communicate openly. There are some tasks that can be done today that help, like knowing exactly what source is being included in a system and how it was configured and built. Automatic creation of accurate Software Bill of Materials (SBOMs), is one pragmatic step that has emerged as a best practice for security and safety analysis. This talk will overview some of the methods being applied in some open source projects (like Linux, Xen & Zephyr), as we try to establish other pragmatic steps when open source projects are used in safety critical: