embeddedworld
Exhibition&Conference

Open Source Software in Safety-Critical Applications: Challenges and Collaborative Solutions

Philipp Ahmann, Etas GmbH (Bosch)

With work from: Nicole Pappler, Kate Stewart, Gabriele Paoloni, Alessandro Carminati, and other OSS community members.

ELISA
Enabling Linux in Safety Applications

Aerospace · Automotive · Linux Features
Medical Devices · OS Engineering Process
Safety Architecture · Space Grade Linux · Systems · Tools

whoami – Philipp Ahmann

ETAS	Sr. OSS Community Manager
ELISA Enabling Linux in Safety Applications	Chair of the Technical Steering Committee Lead of the Systems Working Group
THE LINUX FOUNDATION | Europe	Member of the Inaugural Advisory Board
	OSS enthusiast and promoter

getting back to this agenda page

Agenda *(& navigation notes)*

clickable links to get to different sections
(more links in presentation)

- Projects and approaches
  - Additional examples

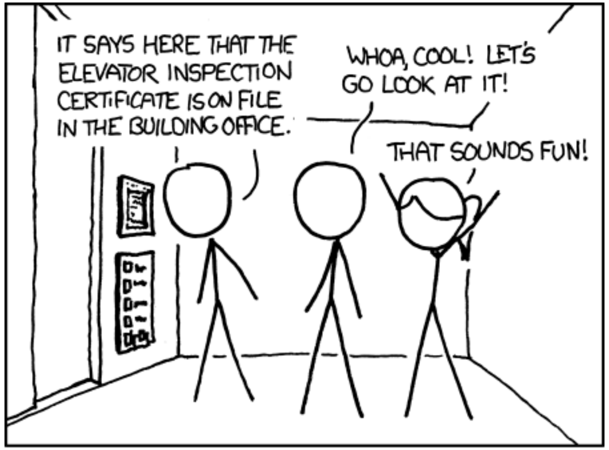
- Requirements tools & implementation
- Requirements within the Linux Kernel
- SPDX Safety profile

- ELISA project brief intro
  - Systems WG / Best practices standard

- Summary
- (Getting Involved)

handout slide

enjoy with your favorite drink


IT SAYS HERE THAT THE ELEVATOR INSPECTION CERTIFICATE IS ON FILE IN THE BUILDING OFFICE.
WHOA, COOL! LET'S GO LOOK AT IT!
THAT SOUNDS FUN!

INDUSTRY TIP: BUILDING OWNERS KNOW THIS NEVER HAPPENS. THOSE SIGNS MARK ELEVATORS WHICH HAVE NEVER BEEN INSPECTED.

https://xkcd.com/897/

Some Collaborative OSS Projects Addressing Functional Safety Gaps and Concerns

Linux: ELISA Enabling Linux in Safety Applications

RTOS: Zephyr™

Virtualization/Hypervisor: Xen Project

The Linux Foundation

Project Members

Platinum Members

ANALOG DEVICES · antmicro · CARIAD A VOLKSWAGEN GROUP COMPANY
Google · intel · Meta
NORDIC SEMICONDUCTOR · NXP · oticon life-changing technology
Qualcomm Innovation Center · ZEISS

Silver Members

ac6 · ALIF SEMICONDUCTOR · ARDUINO · arm
avsystem Zephyr Member · Baumer · Baylibre Zephyr Member · blues
DOULOS · ezurio · golioth · HONDA The Power of Dreams
hubble NETWORK · iar · infineon · inovex
irnas · Linaro · Memfault · MICROCHIP
MICROEJ · percepio · Qt Group · RENESAS
SILICON LABS · ST life.augmented · SYNOPSYS · TEXAS INSTRUMENTS
WNDRVR

Zephyr™

PROJECT MEMBERS

AMD XILINX · arm · aws · epam
VATES · XenServer · HONDA The Power of Dreams

Xen Project

Premier Members

BOEING · Red Hat

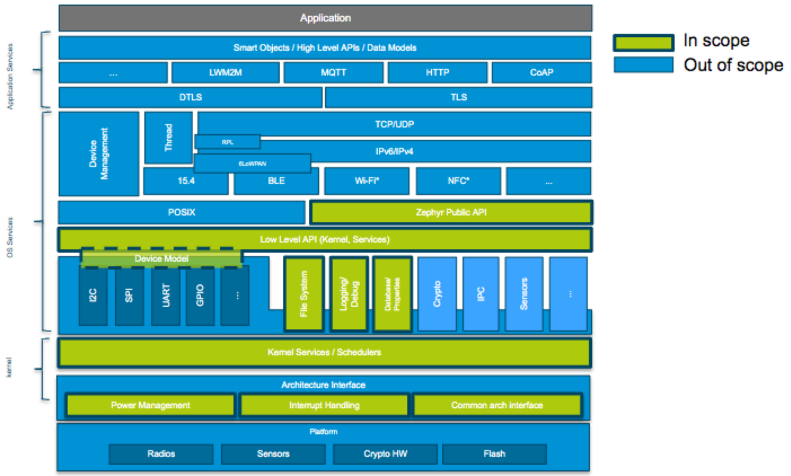
General Members

AISIN · arm · BOSCH · Canonical
Codethink · Elektrobit · EMQ · HONDA The Power of Dreams
HUAWEI · LINUTRONIX · LYNX · NISSAN MOTOR CORPORATION
上汽集团 SAIC MOTOR · WNDRVR

ELISA Enabling Linux in Safety Applications

handout slide

Zephyr

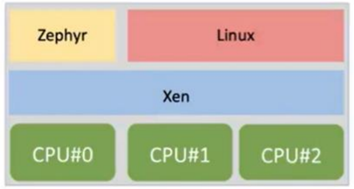


https://www.zephyrproject.org/introduction-of-coding-guidelines-for-zephyr-rtos/

- Targeting safety certification from the beginning of the project
- Certification artifacts and safety manual for premium members only
- Safety working group meets regularly
- Naturally, safety awareness in community is limited due to heavy “non-safety” use cases and many unrelated modules.
- Rich ecosystem with strong support for various HW and certain benefits on Linux.
- Posix compatible

Xen

- Since Xen for embedded security working group was started in parallel (in 2010)
- Security & isolation are project's top priority
- Real-time scheduling.
- Rigorous Quality Process. Full commit traceability.
- Commits are tested with 2 CI loops.
- Widely adopted in critical production environment: (Data center, Desktop & Embedded)
- AMD works on making Xen safety-certifiable
- Continuous certification in mind.
- Phase 1: Certification Concept Approval
- Phase 2: Final Assessment.

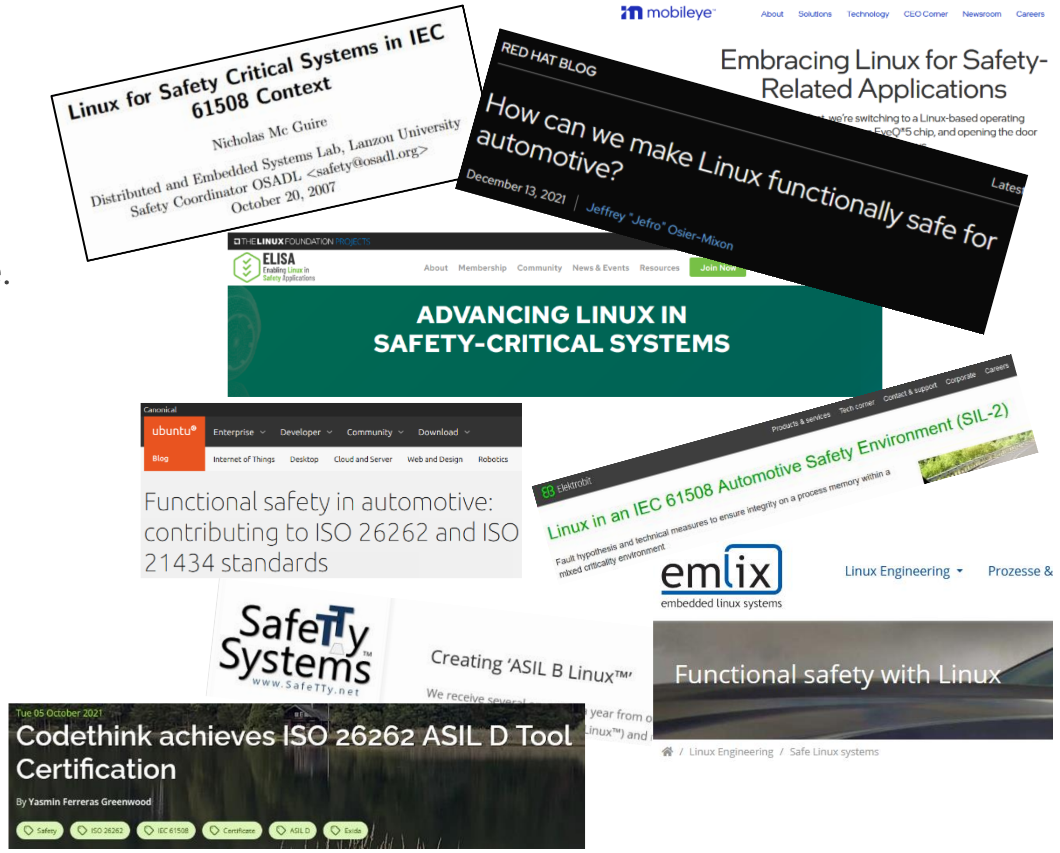


handout slide

Linux

- Open source software superlative.
- Largest community, largest source base.
- Made for flexibility and wide use cases.
- Spread over whole world and in space.
- Several attempts with certification path.
- Gains again momentum for high performance products (e.g. SDV*)
- Prominent open space examples: SIL2LinuxMP and ELISA

*SDV: Software-Defined-Vehicle



Linux for Safety Critical Systems in IEC 61508 Context

Nicholas Mc Guire

Distributed and Embedded Systems Lab, Lanzou University

Safety Coordinator OSADL <safety@osadl.org>

October 20, 2007

Embracing Linux for Safety-Related Applications

…we're switching to a Linux-based operating … EyeQ®5 chip, and opening the door

How can we make Linux functionally safe for automotive?

December 13, 2021 | Jeffrey "Jefro" Osier-Mixon

ADVANCING LINUX IN SAFETY-CRITICAL SYSTEMS

Functional safety in automotive: contributing to ISO 26262 and ISO 21434 standards

Linux in an IEC 61508 Automotive Safety Environment (SIL-2)

Fault hypothesis and technical measures to ensure integrity on a process memory within a mixed criticality environment

Creating 'ASIL B Linux™'

We receive several … year from o… Linux™) and …

Functional safety with Linux

/ Linux Engineering / Safe Linux systems

Tue 05 October 2021

Codethink achieves ISO 26262 ASIL D Tool Certification

By Yasmin Ferreras Greenwood

Community Challenges For All Projects

- Bring the argument of „OSS development is not organized like commercial software“
- Less influence on maintainers
  (positive & negative – no traditional supplier management)
- Harder to train/direct developers
  (but some Xen community members got Misra-C trainings and Zepyhr members IEC 61508 trainings)
- Liability of a community?
  (but commercial provider may be liable – insurance)
- Development process: Requirements, traceability, v-model,…
  mapping safety integrity standards

Fully Open vs. Pretty Open

Started safety-WG in 2023 for better collaboration.

New life to activity due to openness.
Example: requirements tool

Some results remain “behind the scenes” for premium members

Zephyr™

Discussions are open.

Misra-C, documentation and other parts are open source and upstream.

Safety manual and other safety artifacts will be made commercially available via AMD/Xilinx

Xen Project

Completely open to everyone.

Focus is on tools, processes, kernel improvements, and documentation.

Outcome enables other integrators to build their products around Linux.

Certification Financing

Platinum members

AMD/Xilinx

Integrators
(like RedHat or
Codethink)

Trainings

Provide(d) IEC 61508 training by TÜV SÜD for project members (some contributors/maintainers have official safety training)

The safety committee (and safety working group) mainly consist of experienced safety experts.

Misra-C trainings for project contributors via Bugseng sponsored by AMD.

Mainly 1 safety expert, many engineers with safety in mind and practical product experience

Special topic webinars within ELISA.

No direct ISO26262 or IEC61508 trainings for ELISA members.

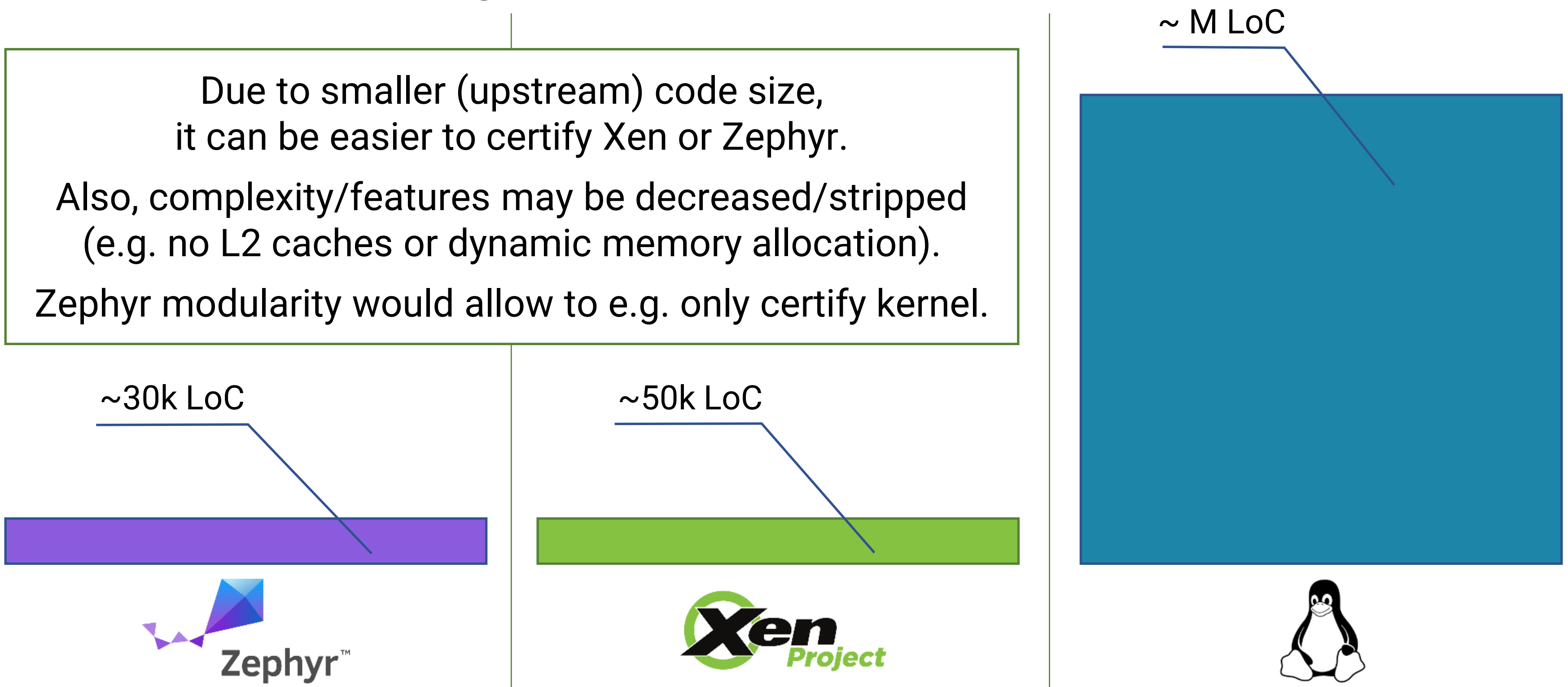
Many experienced safety experts within ELISA project.

Code Complexity/Size

Due to smaller (upstream) code size,
it can be easier to certify Xen or Zephyr.

Also, complexity/features may be decreased/stripped
(e.g. no L2 caches or dynamic memory allocation).

Zephyr modularity would allow to e.g. only certify kernel.



ELISA
Enabling Linux in Safety Applications

Additional Project Examples

Other Certification Projects

Hypervisor: L4Re by Kernkonzept
https://l4re.org/overview.html

- The L4Re Hypervisor and L4Re Micro Hypervisor form the base for virtualization platform for hosting workloads of general-purpose, real-time, security and safety kinds.
- It consists of a small kernel, a microkernel, and a user-level infrastructure that includes basic services such as program loading and memory management up to virtual machine management.

RTOS: ThreadX at Eclipse (Microsoft)
https://threadx.io/

- This RTOS is designed for deeply embedded applications. It provides advanced scheduling facilities, message passing, interrupt management, and messaging services.
- Eclipse ThreadX RTOS has many advanced features, including picokernel architecture, preemption threshold, event chaining, and a rich set of system services.
- **Certified before gone OSS**

ELISA
Enabling Linux in Safety Applications

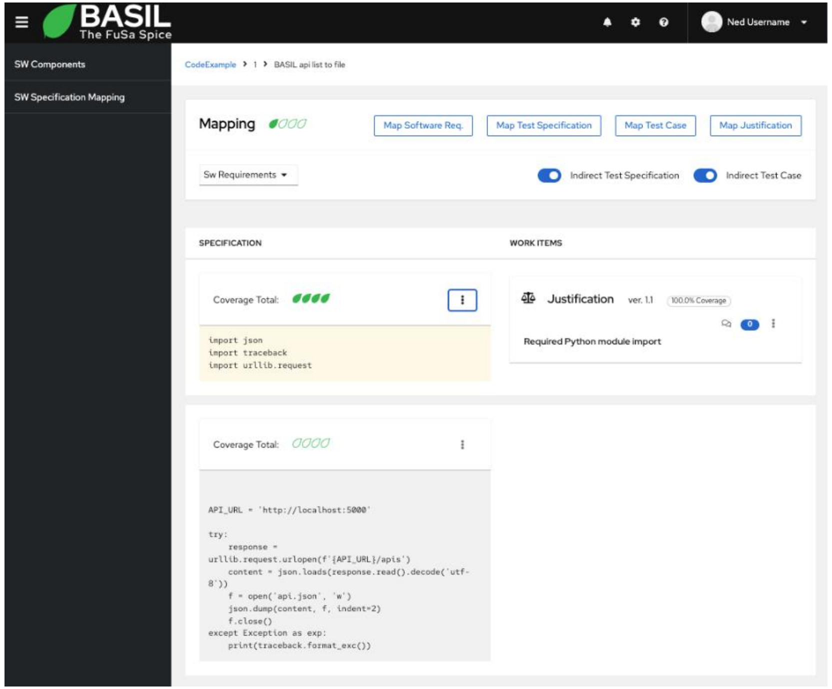
OSS Requirements: Tools & Implementation

Examples of OSS Requirement Tools

BASIL

- https://basil-the-fusa-spice.readthedocs.io/
- A tool developed to support Software Specification analysis, Software Requirements definition and Test Case mapping against source code or Software Specification.
- BASIL is a web application that enable collaboration within multiple users and provide a simplified work item relationships view. It comes also with a REST web api to simplify the integration in other toolchains.

Used e.g. by: ELISA project



```
import json
import traceback
import urllib.request
```

Required Python module import

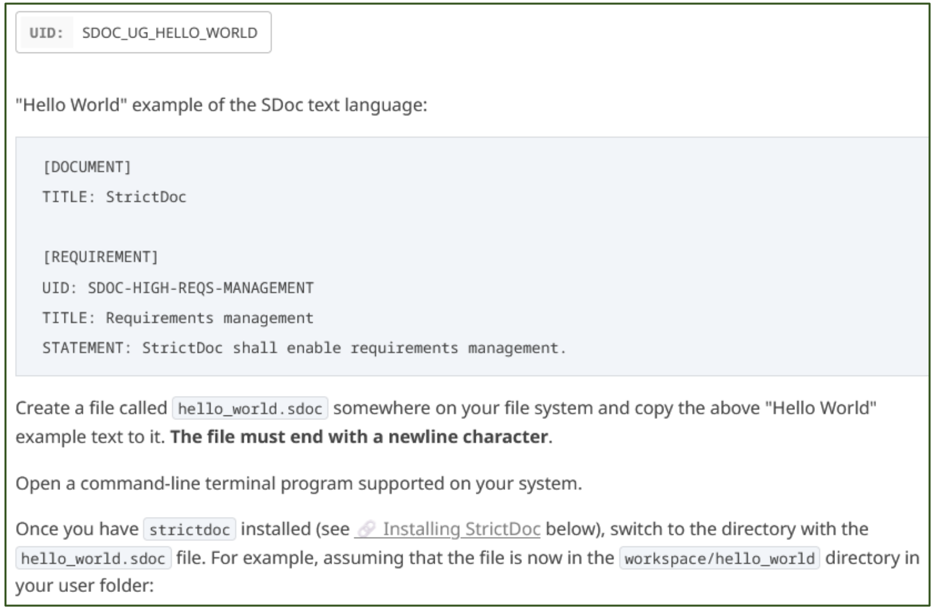
```
API_URL = 'http://localhost:5000'

try:
    response =
urllib.request.urlopen(f'{API_URL}/apis')
    content = json.loads(response.read().decode('utf-
8'))
    f = open('api.json', 'w')
    json.dump(content, f, indent=2)
    f.close()
except Exception as exp:
    print(traceback.format_exc())
```

Examples of OSS Requirement Tools

StrictDoc

- https://strictdoc.readthedocs.io
- StrictDoc efficiently manages requirements and specifications using a human-readable DSL (SDoc), generating output in multiple formats (HTML, PDF, etc.) via a web UI. Key features include traceability, customizable fields (e.g., ASIL, priority), and fast, incremental generation. See limitations for details.
- Developed with "safety in mind"

Used e.g. by: Zepyhr project



UID: SDOC_UG_HELLO_WORLD

"Hello World" example of the SDoc text language:

```
[DOCUMENT]
TITLE: StrictDoc

[REQUIREMENT]
UID: SDOC-HIGH-REQS-MANAGEMENT
TITLE: Requirements management
STATEMENT: StrictDoc shall enable requirements management.
```

Create a file called `hello_world.sdoc` somewhere on your file system and copy the above "Hello World" example text to it. **The file must end with a newline character**.

Open a command-line terminal program supported on your system.

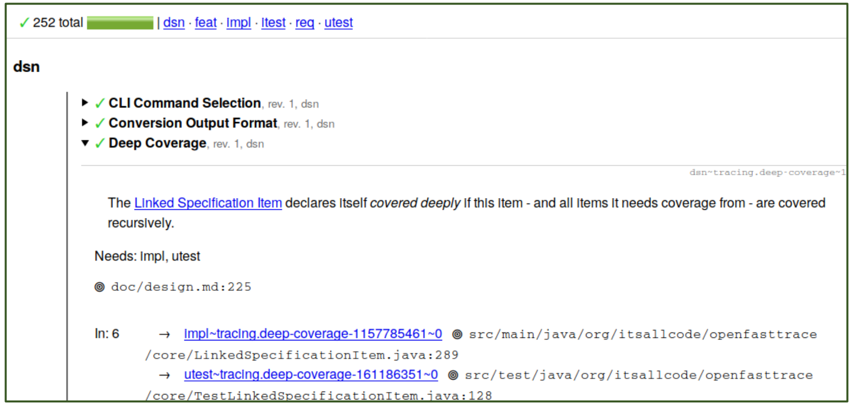
Once you have `strictdoc` installed (see Installing StrictDoc below), switch to the directory with the `hello_world.sdoc` file. For example, assuming that the file is now in the `workspace/hello_world` directory in your user folder:

Examples of OSS Requirement Tools

OpenFastTrace

- https://github.com/itsallcode/openfasttrace
- OpenFastTrace (short OFT) is a requirement tracing suite. Requirement tracing keeps track of whether you actually implemented everything you planned to in your specifications. It also identifies obsolete parts of your product and helps you to get rid of them.

Used e.g. by: Xen project, Eclipse SDV uProtocol



✓ 252 total | dsn · feat · impl · itest · req · utest

dsn

- ▸ ✓ **CLI Command Selection**, rev. 1, dsn
- ▸ ✓ **Conversion Output Format**, rev. 1, dsn
- ▾ ✓ **Deep Coverage**, rev. 1, dsn

`dsn~tracing.deep-coverage~1`

The Linked Specification Item declares itself *covered deeply* if this item - and all items it needs coverage from - are covered recursively.

Needs: impl, utest

```
◎ doc/design.md:225
```

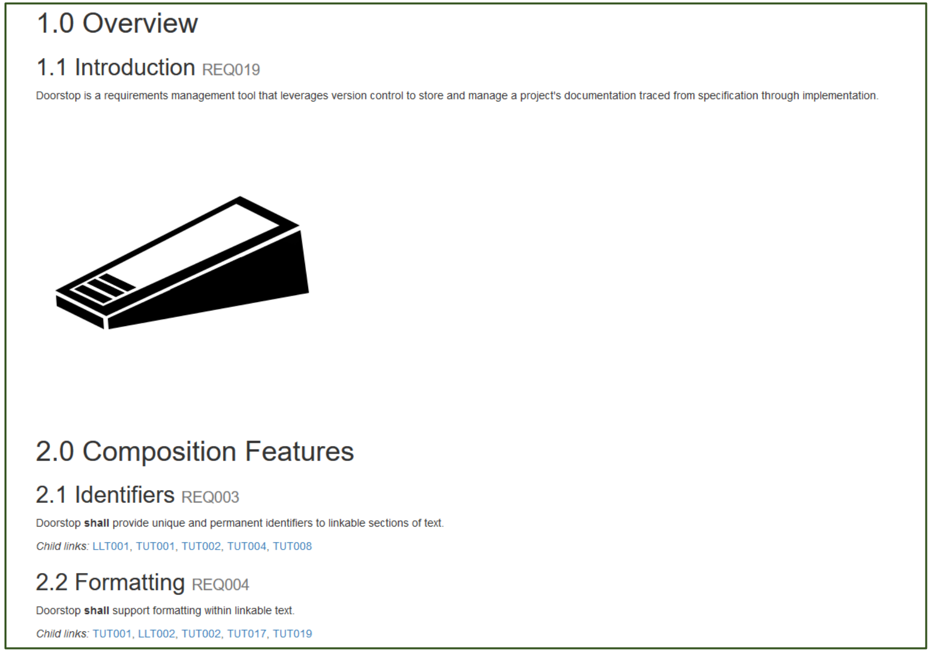
In: 6 → impl~tracing.deep-coverage-1157785461~0 ◎ `src/main/java/org/itsallcode/openfasttrace/core/LinkedSpecificationItem.java:289`
→ utest~tracing.deep-coverage-161186351~0 ◎ `src/test/java/org/itsallcode/openfasttrace/core/TestLinkedSpecificationItem.java:128`

Examples of OSS Requirement Tools

Doorstop

- https://doorstop.readthedocs.io
- Doorstop is a requirements management tool that facilitates the storage of textual requirements alongside source code in version control.
- When a project leverages this tool, each linkable item (requirement, test case, etc.) is stored as a YAML file in a designated directory. The items in each directory form a document. The relationship between documents forms a tree hierarchy.

Used e.g. by: Trustable Software



1.0 Overview

1.1 Introduction REQ019

Doorstop is a requirements management tool that leverages version control to store and manage a project's documentation traced from specification through implementation.

2.0 Composition Features

2.1 Identifiers REQ003

Doorstop **shall** provide unique and permanent identifiers to linkable sections of text.

Child links: LLT001, TUT001, TUT002, TUT004, TUT008

2.2 Formatting REQ004

Doorstop **shall** support formatting within linkable text.

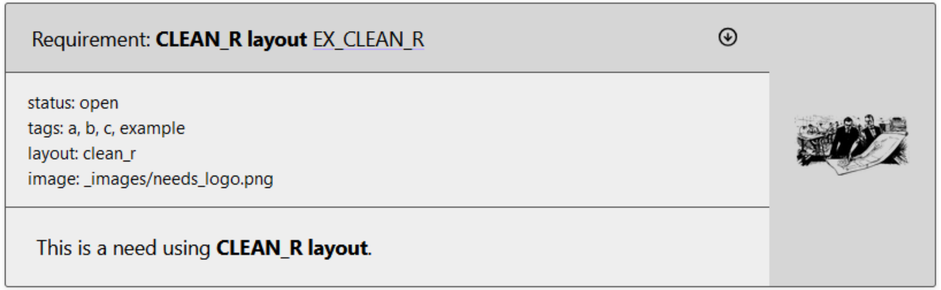
Child links: TUT001, LLT002, TUT002, TUT017, TUT019

Examples of OSS Requirement Tools

Sphinx-needs

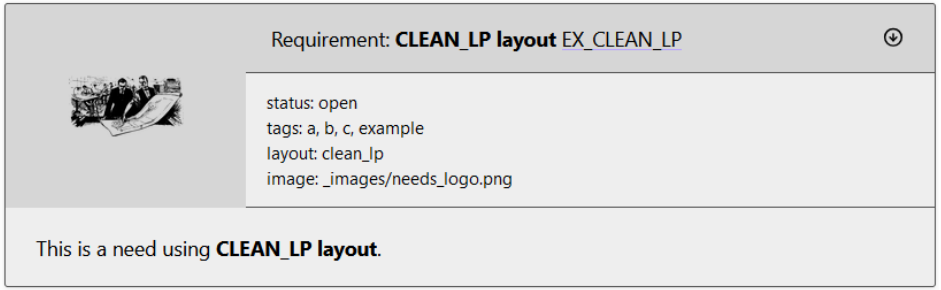
- https://sphinx-needs.readthedocs.io
- Combine Docs-as-Code with Application Lifecycle Management, to track requirements, specifications, test cases, and other engineering objects in your documentation

Used e.g. by: Eclipse S-Core project


Requirement: **CLEAN_R layout** EX_CLEAN_R

status: open
tags: a, b, c, example
layout: clean_r
image: _images/needs_logo.png

This is a need using **CLEAN_R layout**.


Requirement: **CLEAN_LP layout** EX_CLEAN_LP

status: open
tags: a, b, c, example
layout: clean_lp
image: _images/needs_logo.png

This is a need using **CLEAN_LP layout**.

Examples of OSS Requirement Tools

There are more OSS requirements tools out there than presented!

This is just a selection used in safety critical (OSS) projects!

(It does not mean that others are less suitable. It is your choice!)

Foto von Sigmund auf Unsplash

Quick hits: https://github.com/osrmt/osrmt | https://goeb.github.io/reqflow/ | https://github.com/topics/requirements-tracing

OSS Requirement Tools - Summary

- Docs-as-code is a common theme
- Requirements are handled as code
- Different types of „code“: yaml, rst, md, …

- Trace to source code and tests in mind
- High degree of automation in mind
- Further processing with APIs, CLIs, scripts,…
- CI/CD friendly tools

- By engineers for engineers

Photo by Jay Wennington on Unsplash

ELISA
Enabling Linux in Safety Applications

Requirements Within The Linux Kernel?

Linux Kernel Requirements Proposal

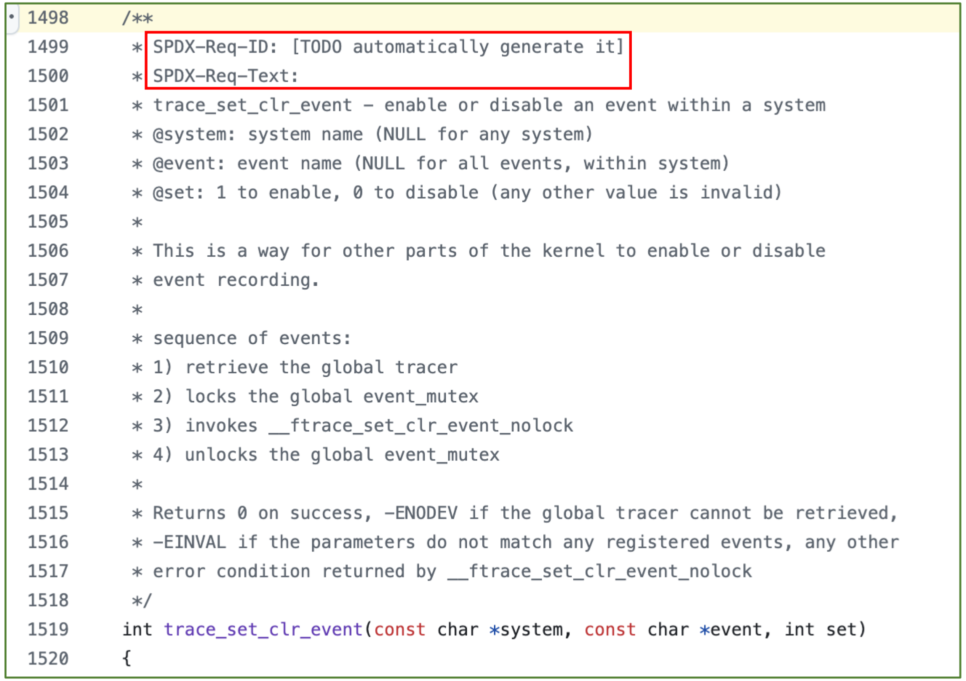
Show me the code!
(Finalizing the initial requirements framework, automation and examples)

- Prototyping Linux Kernel Requirements
- Prototyping the automation to check patch sets against requirements & vice-versa
- Finalizing and publish a Linux Kernel Requirements white paper

Considered parameter

- `SPDX-Req-ID`
- `SPDX-Req-End`
- `SPDX-Req-Ref`
- `SPDX-Req-HKey`
- `SPDX-Req-Child`
- `SPDX-Req-Sys`
- `SPDX-Req-Text`
- `SPDX-Req-Note`

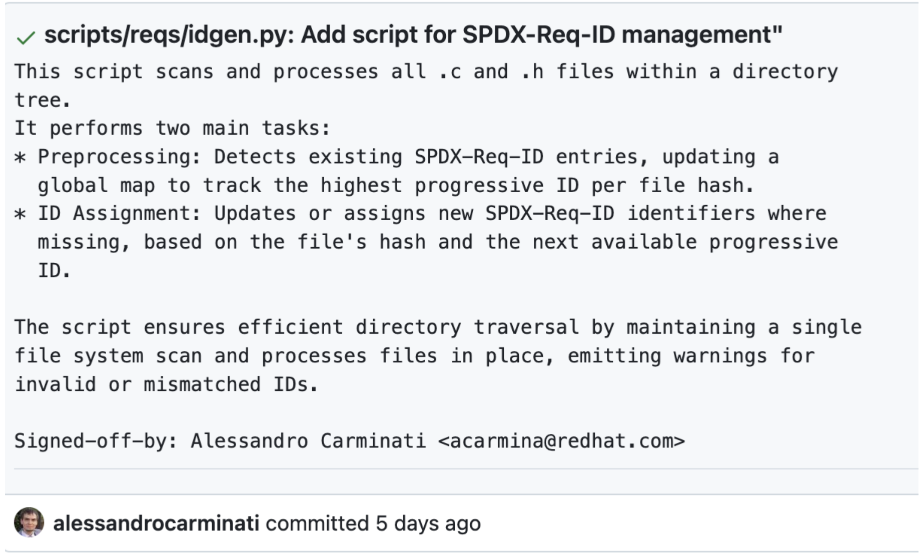
Prototyping Linux Kernel Requirements



```
/**
 * SPDX-Req-ID: [TODO automatically generate it]
 * SPDX-Req-Text:
 * trace_set_clr_event - enable or disable an event within a system
 * @system: system name (NULL for any system)
 * @event: event name (NULL for all events, within system)
 * @set: 1 to enable, 0 to disable (any other value is invalid)
 *
 * This is a way for other parts of the kernel to enable or disable
 * event recording.
 *
 * sequence of events:
 * 1) retrieve the global tracer
 * 2) locks the global event_mutex
 * 3) invokes __ftrace_set_clr_event_nolock
 * 4) unlocks the global event_mutex
 *
 * Returns 0 on success, -ENODEV if the global tracer cannot be retrieved,
 * -EINVAL if the parameters do not match any registered events, any other
 * error condition returned by __ftrace_set_clr_event_nolock
 */
int trace_set_clr_event(const char *system, const char *event, int set)
{
```

- Currently prototyping requirements for some functions in tracing subsystem.
- Requirements shall be:
  - Testable
  - Maintainable inline within the source code
  - Compatible with pre-existing Kernel Doc.
  - Hierarchically traceable
- The main challenge is identifying the main design elements to be documented starting from the pre-existing code

Prototyping the Automation to Check Patch Sets



✓ **scripts/reqs/idgen.py: Add script for SPDX-Req-ID management"**

```
This script scans and processes all .c and .h files within a directory
tree.
It performs two main tasks:
* Preprocessing: Detects existing SPDX-Req-ID entries, updating a
  global map to track the highest progressive ID per file hash.
* ID Assignment: Updates or assigns new SPDX-Req-ID identifiers where
  missing, based on the file's hash and the next available progressive
  ID.

The script ensures efficient directory traversal by maintaining a single
file system scan and processes files in place, emitting warnings for
invalid or mismatched IDs.

Signed-off-by: Alessandro Carminati <acarmina@redhat.com>
```

alessandrocarminati committed 5 days ago

A script to automate the generation of Requirements' IDs (**`SPDX-Req-ID`**) is in progress.

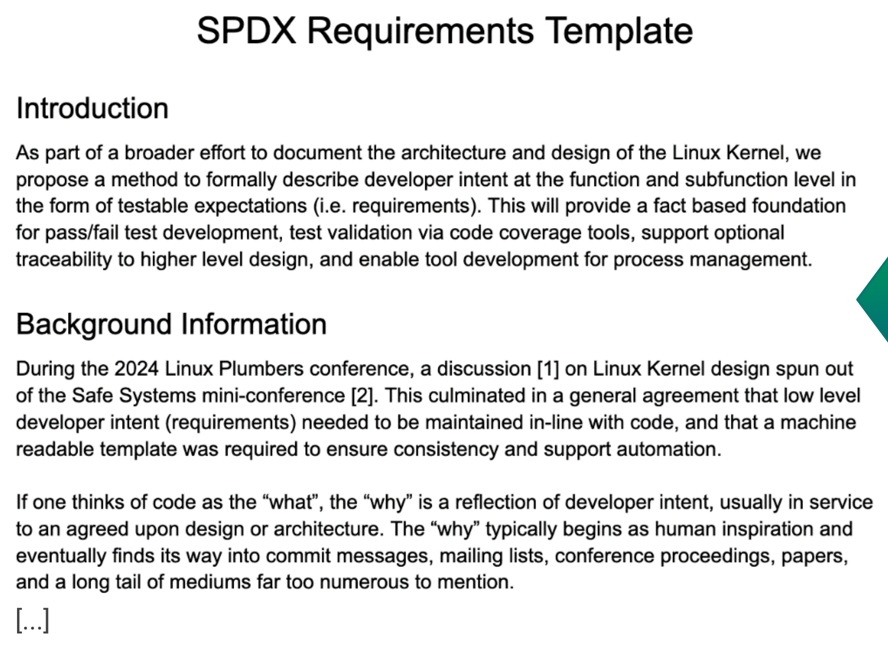
The goal is to generate a unique one that cannot change along the life of the requirements

"**`SPDX-Req-HKey`**" will instead be used to flag if, code or requirement's text changes, so that the requirement will be reviewed against the code (and vice versa).
"**`SPDX-Req-HKey`**" hashes are produced based on the following criteria:

● PROJECT:	The name of the project (e.g. linux)
● FILE_PATH:	The file the code resides in, relative to the root of the project repository.
● INSTANCE:	The requirement template instance, minus tags with hash strings.
● CODE:	The code that the SPDX-Req applies to.

"**`SPDX-Req-ID`**" is the very first "**`SPDX-Req-HKey`**" generated

Finalizing a Linux Kernel Requirements White Paper



SPDX Requirements Template

Introduction

As part of a broader effort to document the architecture and design of the Linux Kernel, we propose a method to formally describe developer intent at the function and subfunction level in the form of testable expectations (i.e. requirements). This will provide a fact based foundation for pass/fail test development, test validation via code coverage tools, support optional traceability to higher level design, and enable tool development for process management.

Background Information

During the 2024 Linux Plumbers conference, a discussion [1] on Linux Kernel design spun out of the Safe Systems mini-conference [2]. This culminated in a general agreement that low level developer intent (requirements) needed to be maintained in-line with code, and that a machine readable template was required to ensure consistency and support automation.

If one thinks of code as the "what", the "why" is a reflection of developer intent, usually in service to an agreed upon design or architecture. The "why" typically begins as human inspiration and eventually finds its way into commit messages, mailing lists, conference proceedings, papers, and a long tail of mediums far too numerous to mention.

[...]

- The ELISA Architecture working group is collaborating on a work in progress draft here.
- Following the finalization of the initial requirements' framework and examples, the draft will be refined, and a whitepaper should be published to engage with the community of Linux developers.

Stay tuned & wait for the publication!

ELISA
Enabling Linux in Safety Applications

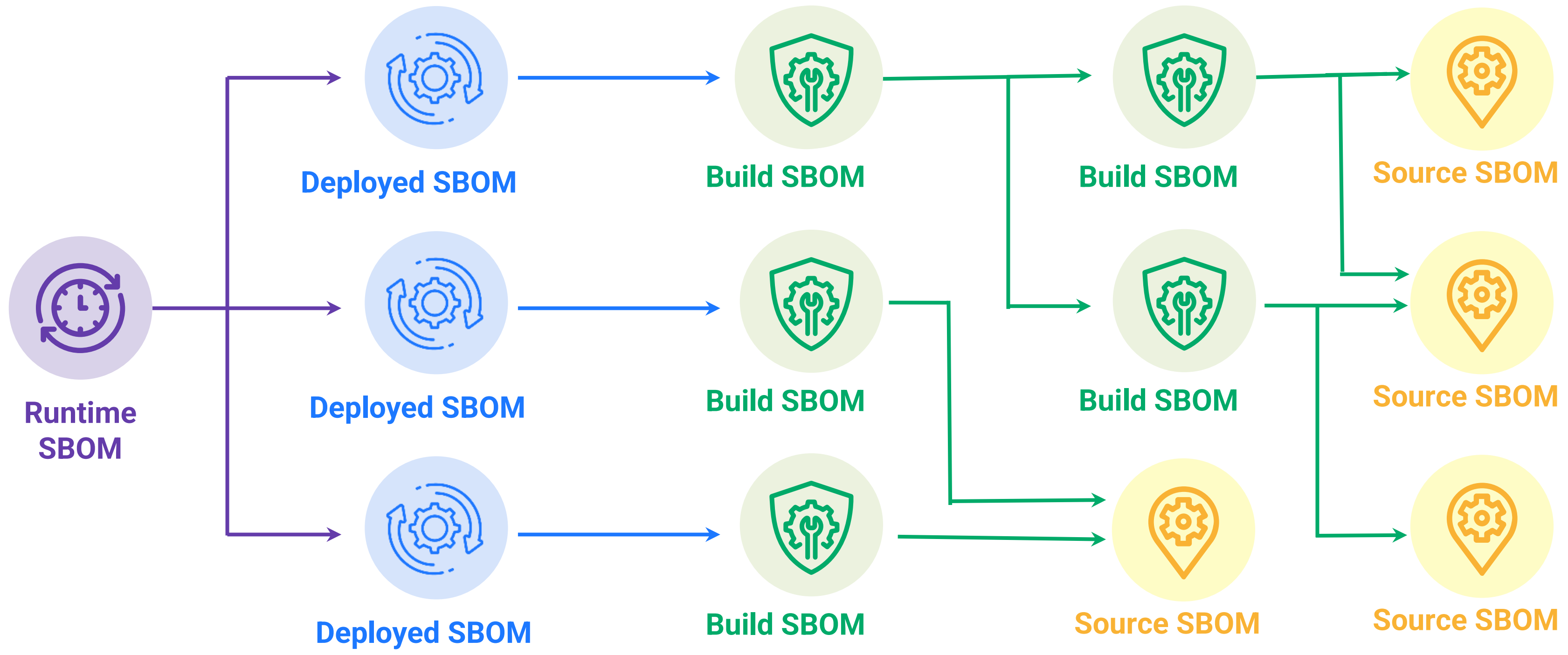
SPDX Safety Profile

SPDX Safety Profile

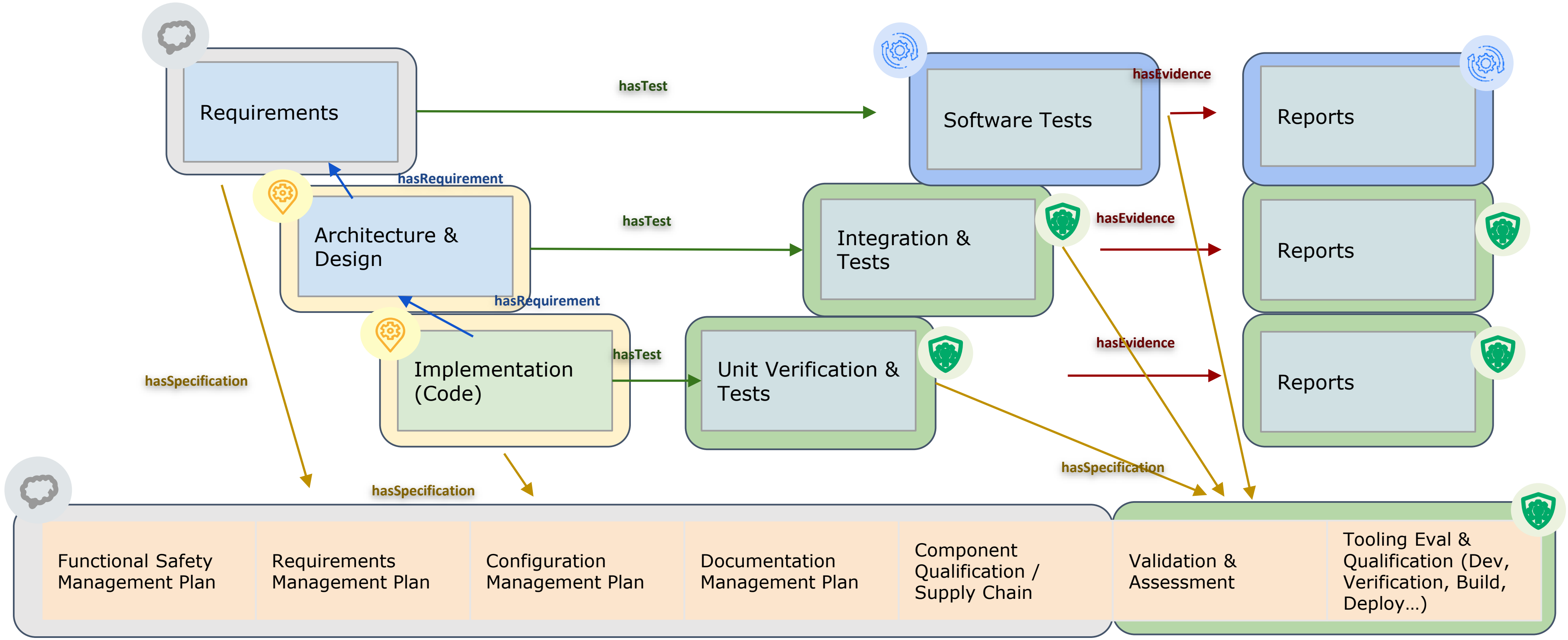
SPDX SAFETY

- Profile team formed in August 2022
- Mailing list: https://lists.spdx.org/g/spdx-fusa
- Scope:
  - Provide a complete model of dependencies in a safety related project
  - Support effective impact analysis methodologies (input information for FMEA, Ishikawa Analysis, GSN/SACM etc.)
  - Provide reproducible results in both impact analysis and evidence generation
  - Formal way to demonstrate completeness after project tailoring and for different scopes

Understanding Safety Critical System: Traceability

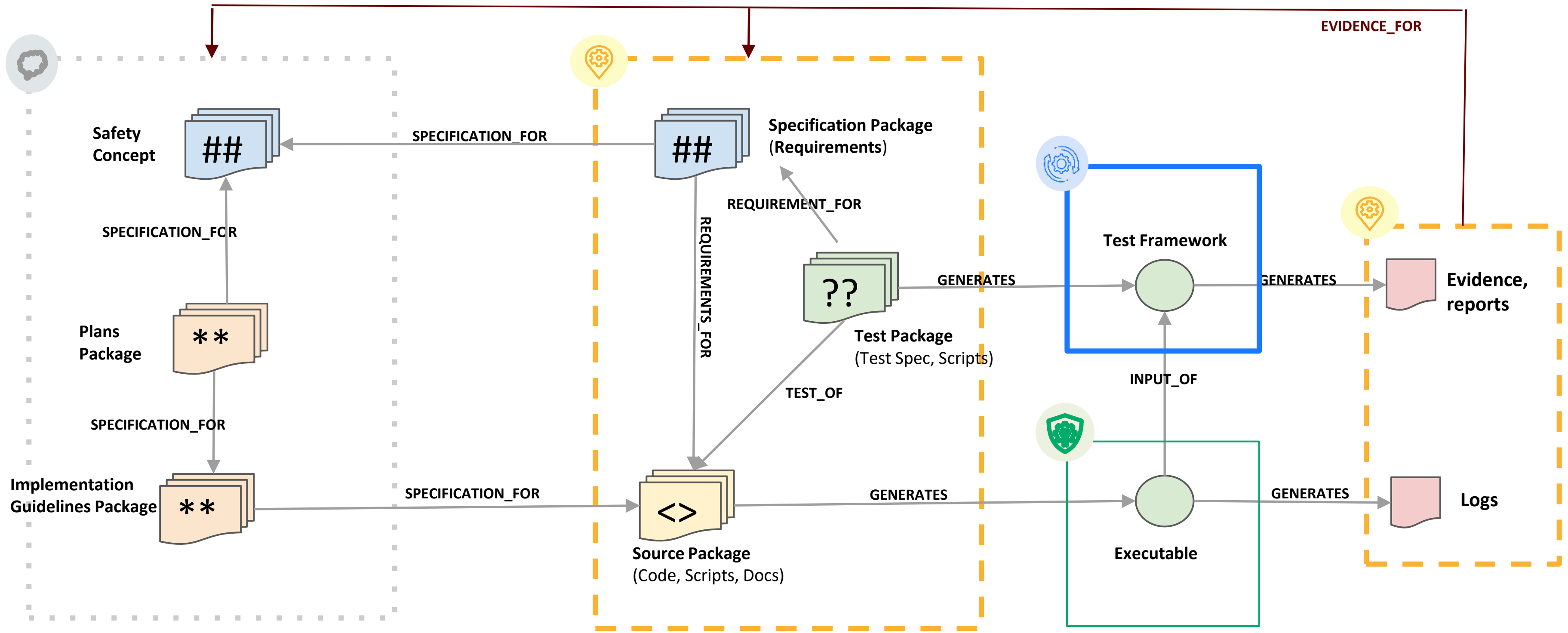


SPDX Safety Dependencies in a FuSa Project



back to agenda

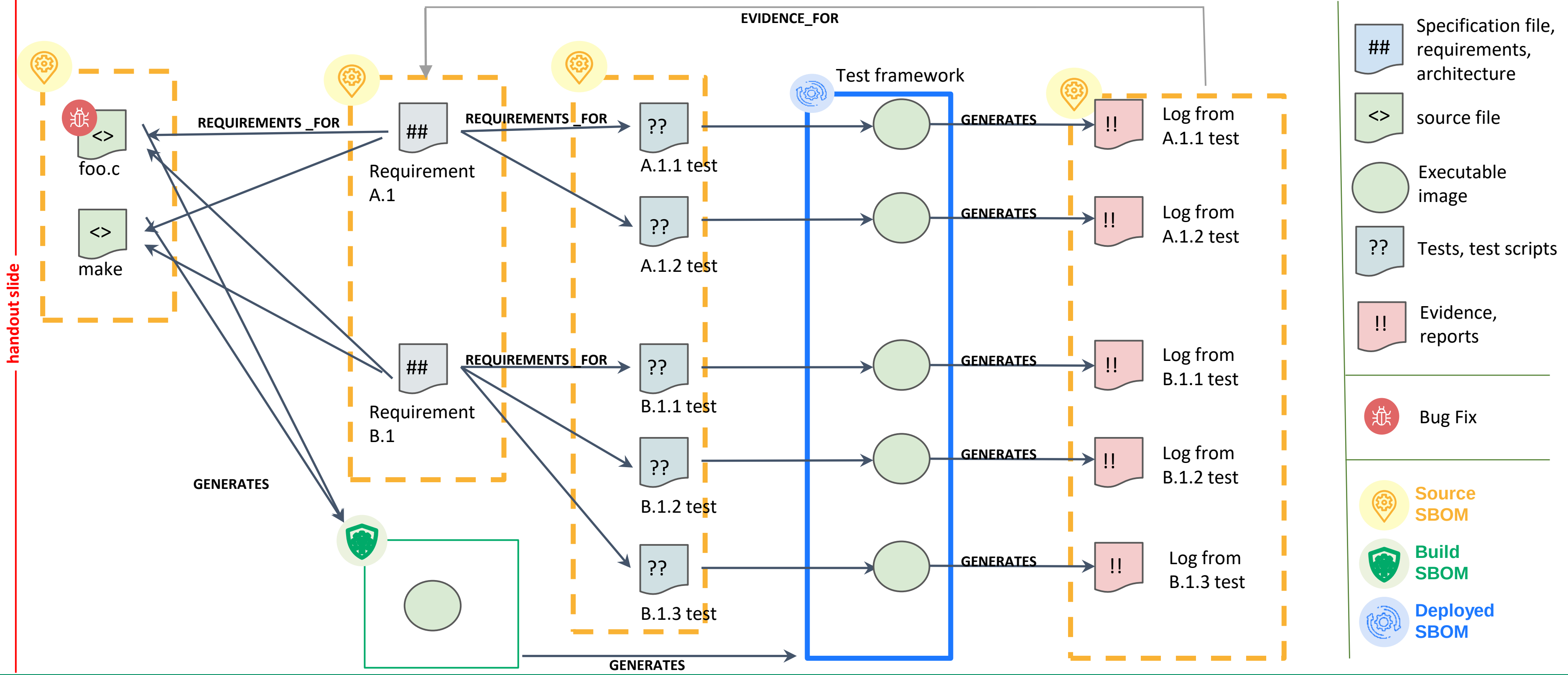
Supporting System Knowledge Graph Creation



SPDX enables component metadata modularity and relationships between components, allowing to create a knowledge graph inside a database for efficient safety, security, and change management analysis on updates.

Inside Component: Traceability of Source to Requirements

Code to Requirements to Tests to Evidence



ELISA
Enabling Linux in Safety Applications

ELISA project

ELISA Project

- Enabling **Safety-critical applications** with **Linux** (beyond Security)
- Increase **dependability & reliability** for whole Linux ecosystem
- **Various use cases**: Aerospace, Automotive, Medical & Industrial
- Supported by major **industrial grade Linux distributors** known for mission critical operation and various industries representatives
- Close community collaboration with **Xen, Zephyr, SPDX, Yocto & AGL** projects
- **Reproducible system** creation from specification to testing
- SW **elements**, engineering **processes**, development **tools**

ELISA : Architecture · Processes · Features · Tools · Systems

The Two Perspectives of …
Enabling Linux in Safety Applications

„Safe Linux“ is not „safe Linux“

Safety allocated to the system where Linux supports the safety application

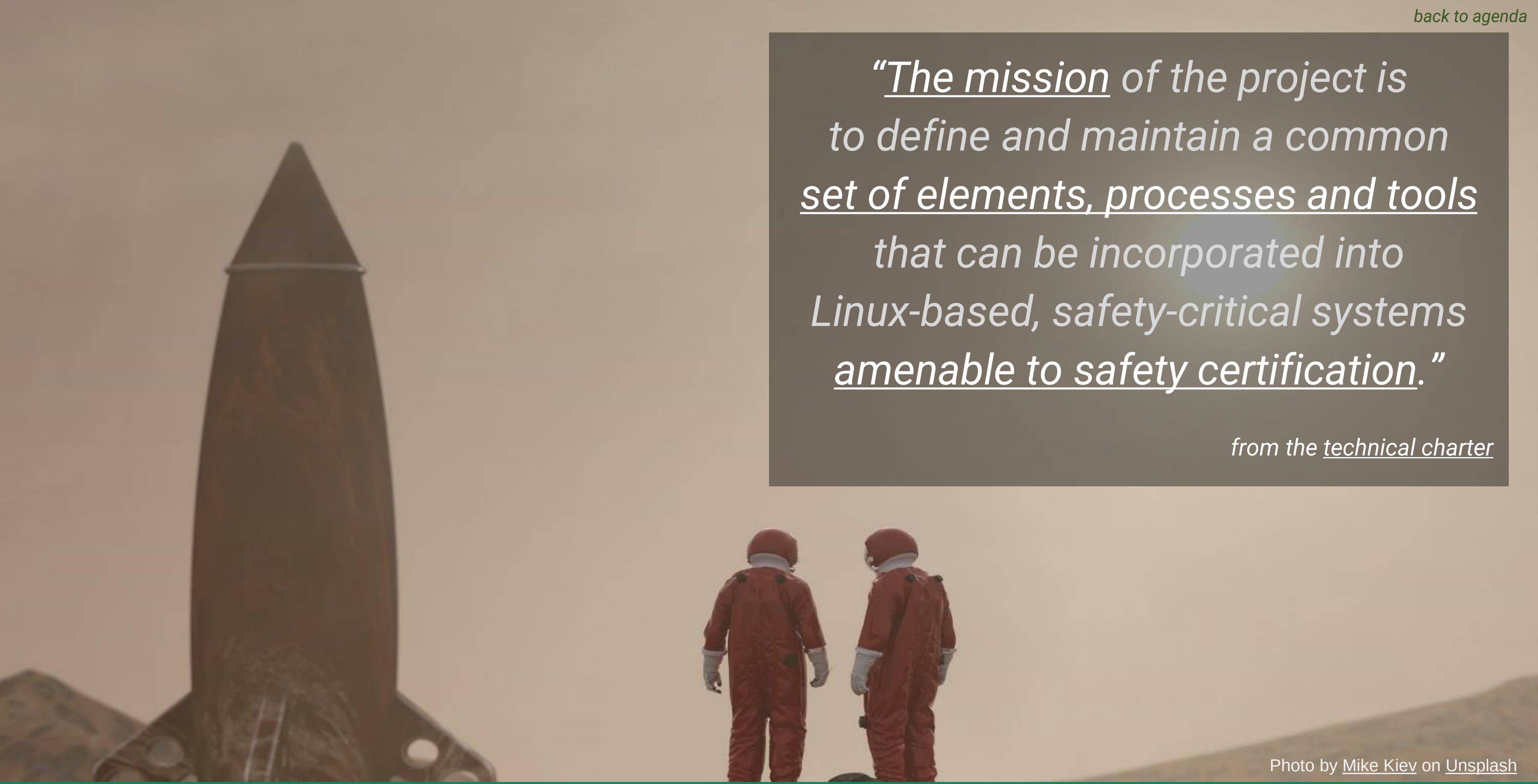
Safety allocated to Linux as safety-critical element

STOP - Limitations! The collaboration …

- *cannot* engineer your system to be safe.
- *cannot* ensure that you know how to apply the described process and methods.
- *cannot* create an out-of-tree Linux kernel for safety-critical applications. (continuous process improvement argument!)
- *cannot* relieve you from your responsibilities, legal obligations and liabilities.

But…

ELISA provides a <u>path forward</u> and peers to <u>collaborate</u> with!



"The mission of the project is to define and maintain a common set of elements, processes and tools that can be incorporated into Linux-based, safety-critical systems amenable to safety certification."

from the technical charter

Photo by Mike Kiev on Unsplash

Premier Members

Boeing · Red Hat

General Members

AISIN · arm · BOSCH · Canonical

Codethink · Elektrobit · EMQ · HONDA The Power of Dreams

HUAWEI · LINUTRONIX LINUX FOR INDUSTRY · LYNX SOFTWARE TECHNOLOGIES · NISSAN MOTOR CORPORATION

SAIC 上汽集团 SAIC MOTOR · WNDRVR

Associate Members

AUTOMOTIVE GRADE LINUX · IFST INSTITUT FÜR FLUGZEUG-SYSTEMTECHNIK

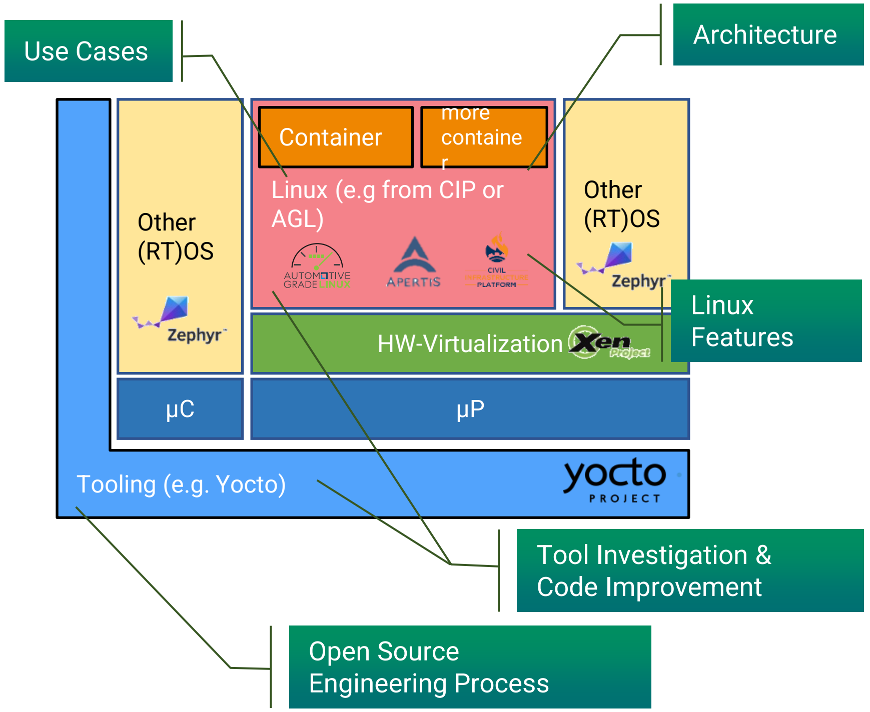
KernelCI · OTH OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

Industry Support

CIVIL INFRASTRUCTURE PLATFORM · OSADL Open Source Automation Development Lab eG · UL

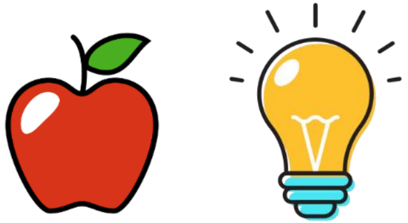
ELISA Working Groups - Fit in an Exemplary System

- **Linux Features**, **Architecture** and Code Improvements should be integrated into the reference system directly.
- **Tools** and **Engineering process** should serve the reproducible product creation.
- **Medical, Automotive, Aerospace** and future WG use cases should be able to strip down the reference system to their use case demands.



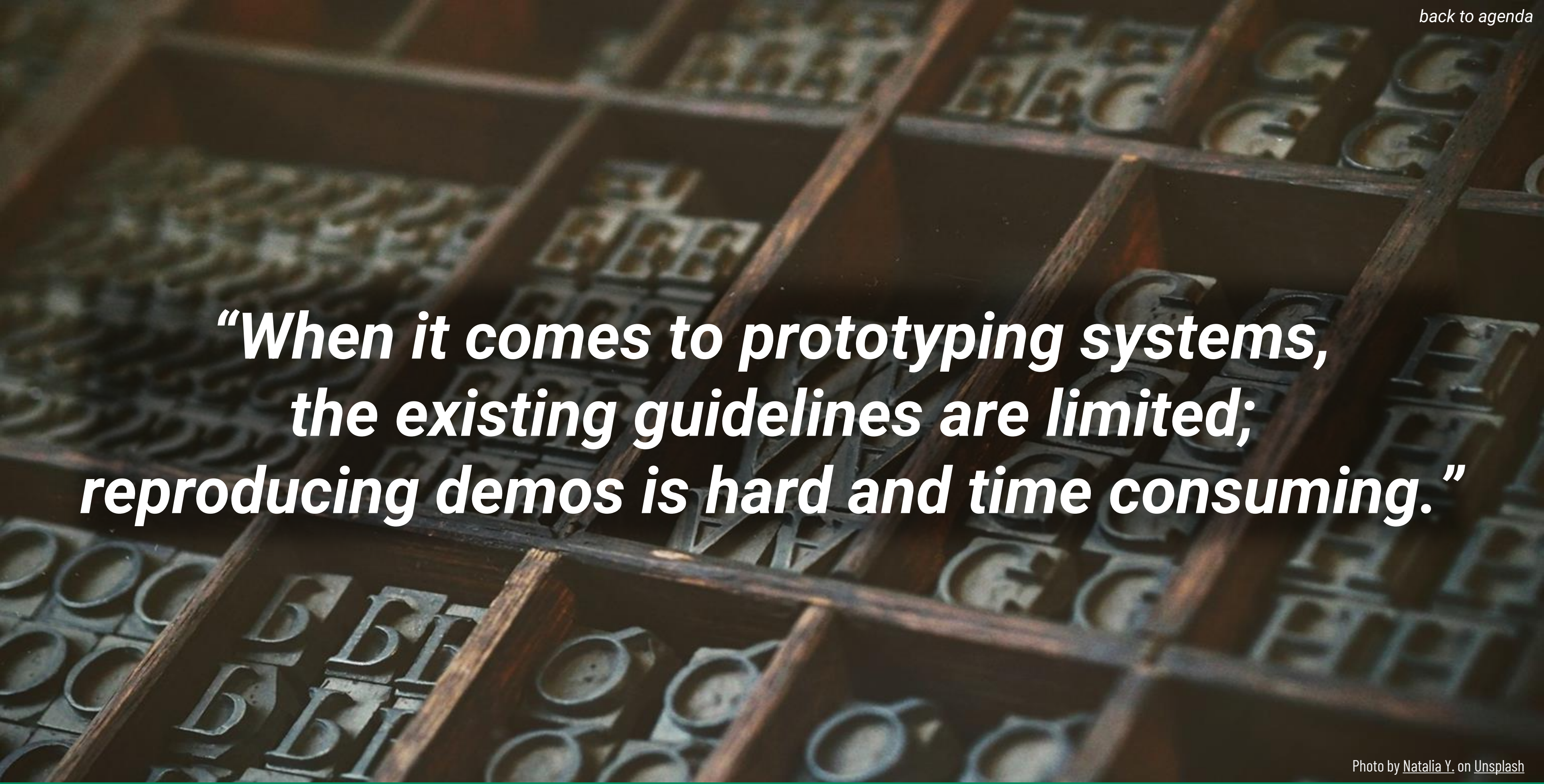
Interactions Between the Communities

- Open source projects focusing on safety-critical analysis
- Open source projects with safety-critical relevance and comparable system architecture considerations
- Further community interactions



"If you have an apple and I have an apple and we exchange these apples then you and I will still each have one apple
But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas

— George Bernard Shaw



"When it comes to prototyping systems, the existing guidelines are limited; reproducing demos is hard and time consuming."

Photo by Natalia Y. on Unsplash

ELISA
Enabling Linux in Safety Applications

Systems WG

Foto von Sajad Nori auf Unsplash

Linux in Safety Critical Systems

“Assessing whether a system is safe, requires understanding the system sufficiently.”

- Understand Linux within that system context and how Linux is used in that system.
- Select Linux components and features that can be evaluated for safety.
- Identify gaps that exist where more work is needed

  to evaluate safety sufficiently.

Example System - At Embedded World 2024

- Xilinx ZCU102 running Xen, Zepyhr, Linux
- Qemu version also exists

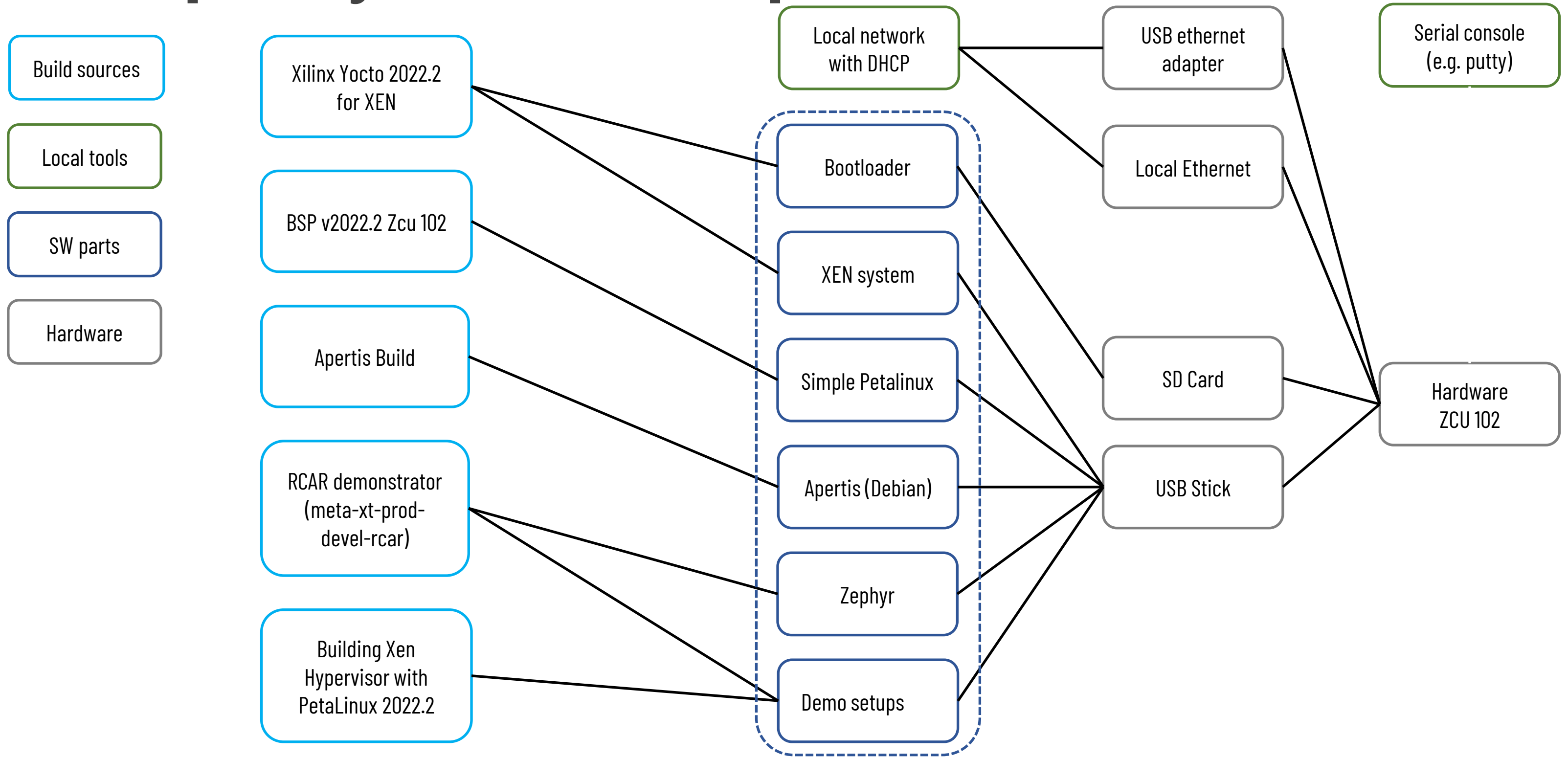
- Software built in ELISA CI
- Focus on reproducibility
- Examples provided as base for extension

- Detailed documentation available

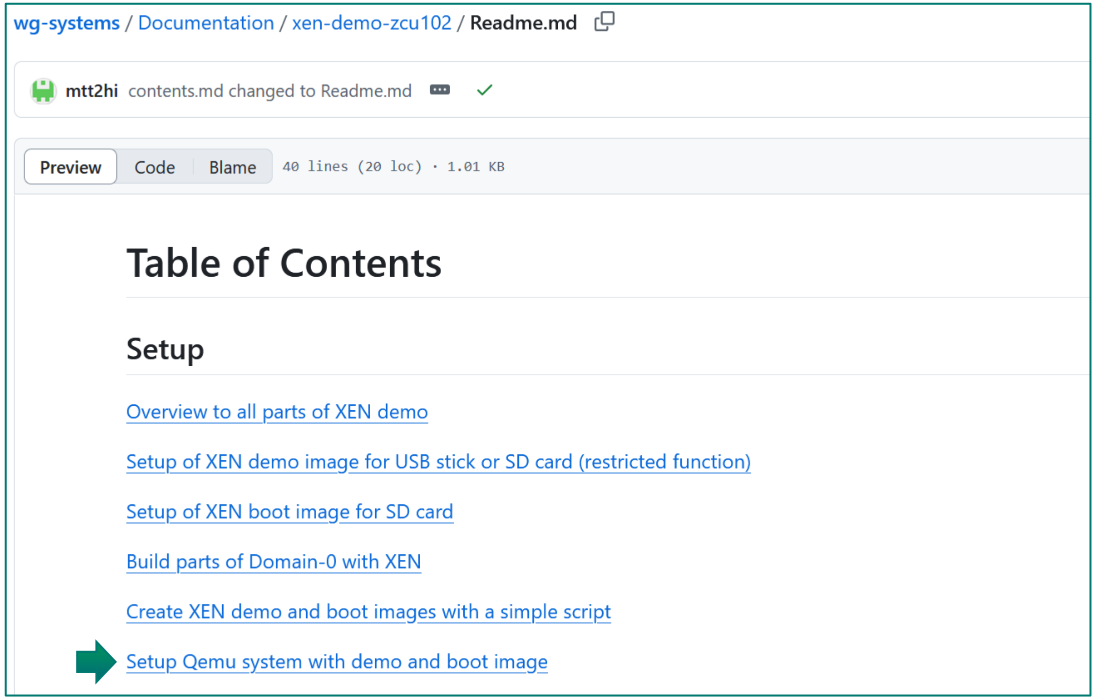
Looking for new hardware in 2025
Candidates: ARM-qemu, Xilinx Kria, Pi5, …
(open for your suggestions and contribution)

https://elisa.tech/blog/2024/04/09/elisa-project-at-embedded-world/

Example System - Composition



Example System - Reproducibility & Documentation



wg-systems / Documentation / xen-demo-zcu102 / Readme.md

mtt2hi contents.md changed to Readme.md

40 lines (20 loc) · 1.01 KB

Table of Contents

Setup

https://github.com/elisa-tech/wg-systems/blob/main/Documentation/xen-demo-zcu102/Readme.md

ELISA
Enabling Linux in Safety Applications

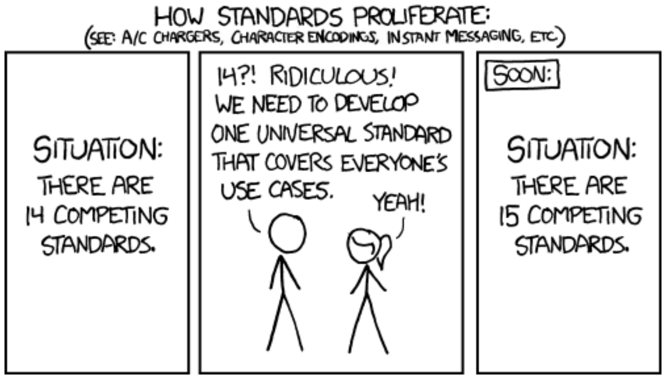
Best Practices Standard

Open Source Good Practices - Yet Another Standard

- Standards are based on v-model
- Nobody is strictly following v-model. 😱

- ASPICE or CMMI are main argumentation for quality management in Automotive
- Quality management can be entry point to safety standards

- No existing standard matches decent software development practices! (code-centric, CI driven and agile focus)


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

https://xkcd.com/927/

Open Source Good Practices - Goal

The goal of this project is to <u>evaluate and document</u> established <u>open source</u> development <u>best practices</u>

&

to provide an <u>assessment guide</u> for the user to <u>rate the quality</u> of open source projects.

Photo by Paul Skorupskas on Unsplash

Open Source Good Practices - Overview

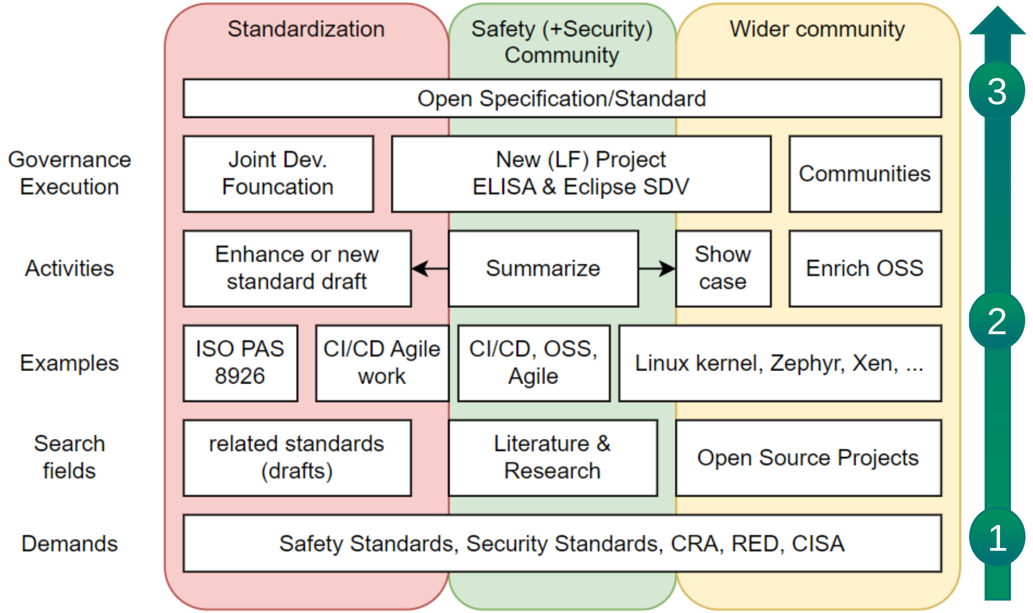
Phases
1. Determination of status quo
2. Definition of practices
3. Assessment of pilot projects

Contribution
- Academia, Public sector, OSS communities
- Industries: Medical, Robotics, Avionics, Automotive, Railway, Automation, … (SME to industry leaders)

Funding
- Funding to be clarified (PfP or by members?)

Reach out, if you are interested in this effort.

Press Release & Survey under preparation



ELISA
Enabling Linux in Safety Applications

Summary

Recommendations for New Contributors

- Just show up – All presented projects are open for the adaptation of new use cases, input, domain-specific working groups etc.

- Share Safety Best Practice: Functional and structural expectations of the component used in the context of the entire system

- Become an OSS evangelist: Open source can already be used in a variety of safety contexts. Knowledge of the actual structure and potential is very scarce in the field of assessors, notified bodies and related authorities.

JOIN THE COMMUNITY

Our infrastructure and tools are open by default, so jump in and introduce yourself, ask questions and share ideas. Please consider this your invitation to participate.

Subscribe to Mailing Lists

Join Community Meetings

Contribute to Tools and Docs on GitHub

Participate in Working Groups

Attend Events

Summary – In a Nutshell

General takes:

- Open Source Requirements tools exist!
- Tools support tracing to code and test.
- Safety SBOMs need to be considered.
- Automation is central element for continuous compliance.

- Certifying pre-existing software is harder (than integrating safety from the beginning)
- Safety & OSS can go together
- Two flavors of "safe Linux" are promoted.

ELISA related activities to remember:

- Requirements inside the kernel (&tools)
- Reproducible example System
- Good Practices in OSS standard
- SPDX Safety SBOM

Photo by Dave Avram on Unsplash

ELISA Enabling Linux in Safety Applications

embeddedworld Exhibition&Conference

Thank you!

https://elisa.tech

handout slide

Getting Involved

handout slide

Getting involved with ELISA

https://elisa.tech

https://github.com/elisa-tech

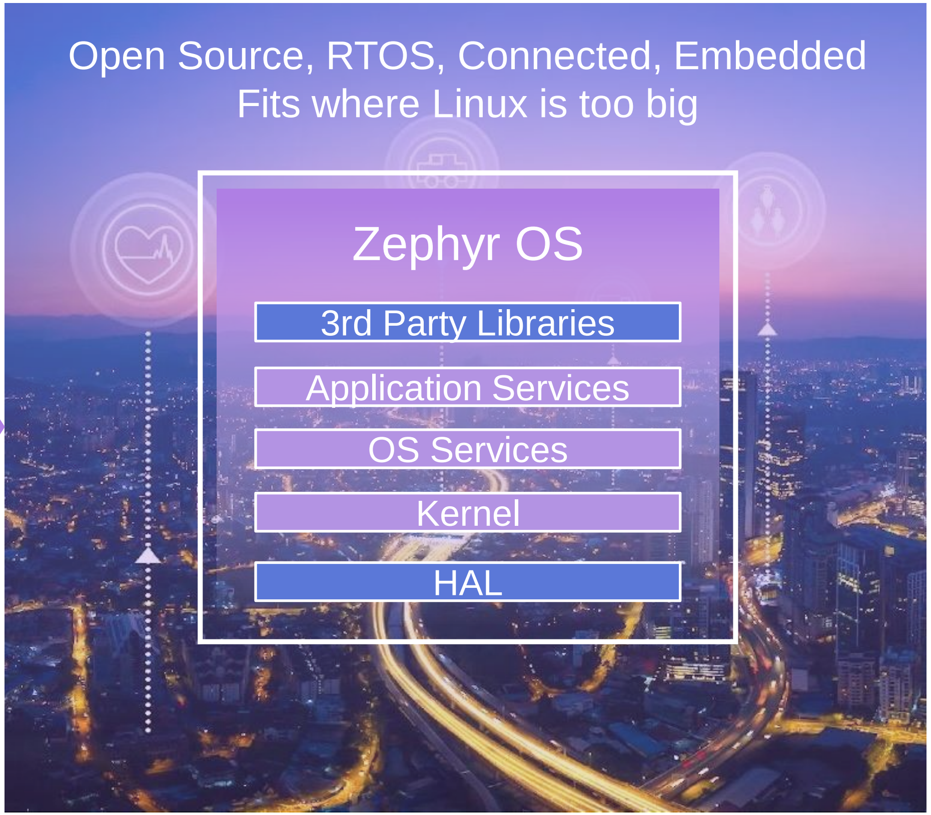
https://lists.elisa.tech

https://www.youtube.com/@elisaproject8453

handout slide

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



THE LINUX FOUNDATION PROJECTS

handout slide

Getting involved with Zephyr

https://www.zephyrproject.org

https://www.github.com/zephyrproject-rtos

https://lists.zephyrproject.org

https://chat.zephyrproject.org

handout slide

Getting involved with Xen

https://www.xenproject.org

https://github.com/xen-project

https://xenproject.org/help/mailing-list/

https://xenproject.org/help/matrix/