The Way for Safe Systems Constructed by Open Source Software

Philipp Ahmann, Etas GmbH (Bosch)

With work from: Nicole Pappler, Kate Stewart, Gabriele Paoloni, Alessandro Carminati, Stefano Stabellini, 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

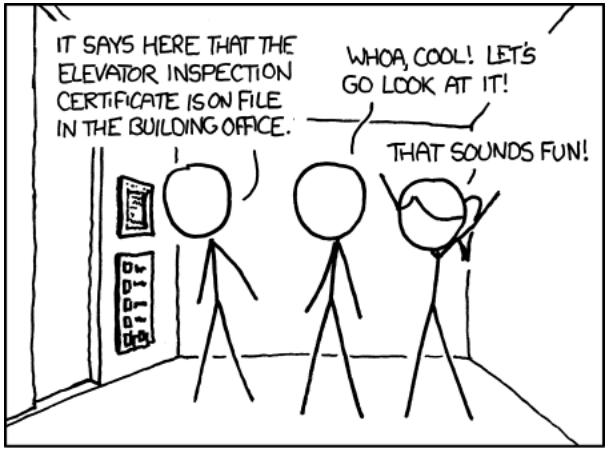
ELISA Enabling Linux in Safety Applications — Chair of the Technical Steering Committee
Lead of the Systems Working Group

THE LINUX FOUNDATION | Europe — Member of the Inaugural Advisory Board

OSS enthusiast and promoter

Agenda

- Intro & Motivation
- Example Open Source (OSS) Approaches
- Open Source Requirement Tools
- Requirements Within The Linux Kernel?
- Summary & Call to Action



https://xkcd.com/897/

ELISA
Enabling Linux in Safety Applications

Intro & Motivation

Introduction & Motivation

- Safety integrity standards need to adopt to increasing complexity of products
- Safety requires a robust fundament based on processes, technical measures and statistical analysis
- Growing industry interest in open source for safety-certified applications
- Current challenges in integrating open-source solutions with safety standards

(China is already making heavy use of Open Source in Automotive systems)

The Fundamental Challenge

- Traditional development processes / v-model vs. code centric open source development
- Standard checklist-based approaches vs. collaborative development
- The need for (formal) traceability and documentation in safety-critical systems

Route to Safety Certification

- IEC 61508 Route 3S for pre-existing software
- ISO 26262-8 clause 12 approach for automotive applications
- ISO PAS 8926 as a bridge for complex software
- Challenges increase with increased system complexity (like Linux systems)

Procedural Requirements for Safety

- Structured documentation of requirements
- Test-to-requirement traceability
- Keeping documentation synchronized with code
- Achieving maintainability over decades

Community Challenges For All Projects

- Argument of „OSS development is not organized like commercial software“
- Less influence on maintainers -> more on maintainers later
  (positive & negative – no traditional supplier management)
- Harder to train/direct developers -> see what trainings have been done later
- Liability of a community?
  (but commercial provider may be liable – insurance)
- Development process: Requirements, traceability, v-model,…
  mapping safety integrity standards

ELISA
Enabling Linux in Safety Applications

Example OSS Approaches

Some Collaborative OSS Projects Addressing Functional Safety Gaps and Concerns

Linux:

RTOS:

Virtualization/Hypervisor:

Project Members

Platinum Members

ANALOG DEVICES · antmicro · CARIAD · Google · intel. · Meta · NORDIC SEMICONDUCTOR · NXP · oticon · Qualcomm Innovation Center · RENESAS · ZEISS

Silver Members

ac6 · ALIF SEMICONDUCTOR · ARDUINO · arm · Avaystem · Baumer · Baylibre · Blecon · blues · DOULOS · ezurio · golioth · HONDA · hubble · iar · Infineon · inovex · irnas · Linaro · Memfault · MICROCHIP · MICROEJ · percepio · Qt Group · SILICON LABS · ST · SYNOPSYS · TEXAS INSTRUMENTS · WNDRVR

Zephyr™

PROJECT MEMBERS

AMD XILINX · arm · aws · epam · VATES · XenServer · HONDA · RENESAS

Xen Project

Premier Members

BOEING · Red Hat

General Members

AISIN · arm · BOSCH · Canonical · CodeThink · Elektrobit · EMQ · HONDA · HUAWEI · LINUTRONIX · LYNX · NISSAN MOTOR CORPORATION · SAIC MOTOR · WNDRVR

ELISA Enabling Linux in Safety Applications

Other Certification Projects

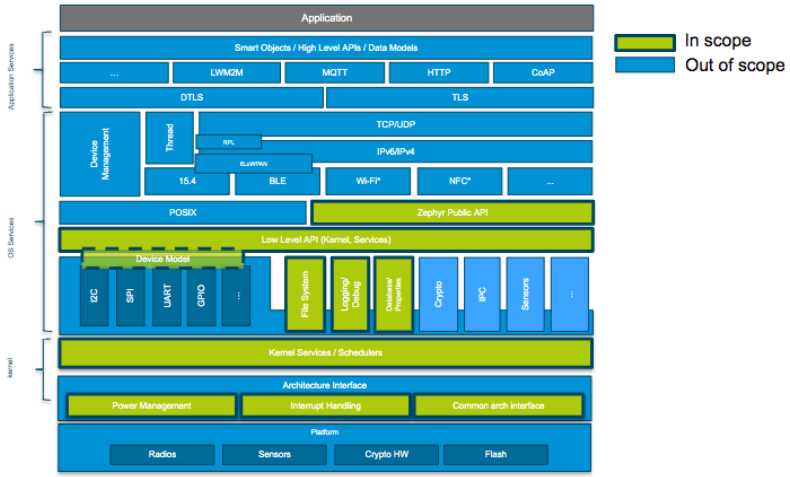
Hypervisor: L4Re by Kernkonzept
https://l4re.org/overview.html

- 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**

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.

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

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

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 path forward and peers to collaborate with!

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.

ELISA
Enabling Linux in Safety Applications

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.

Certification Financing

Platinum members

AMD/Xilinx

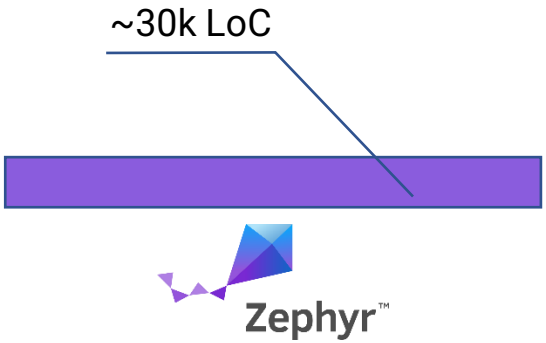
Integrators
(like RedHat or
Codethink)

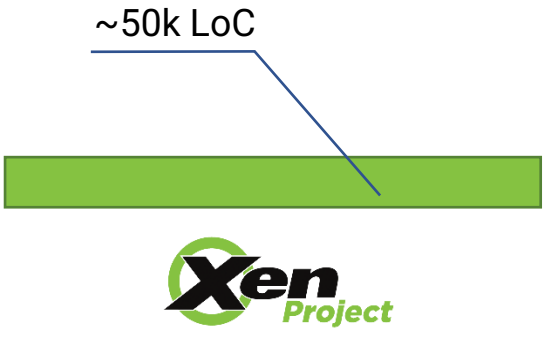
Code Complexity/Size

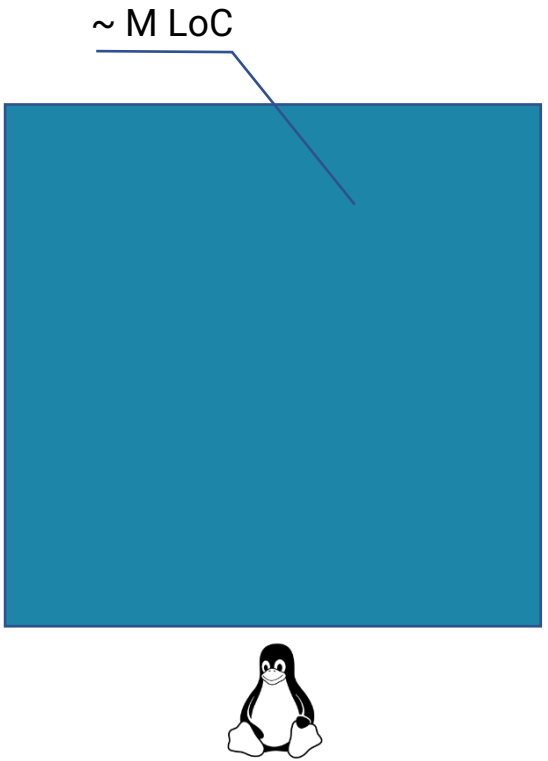
Due to smaller (upstream) code size,
it can be easier to certify Xen or Zephyr.

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.







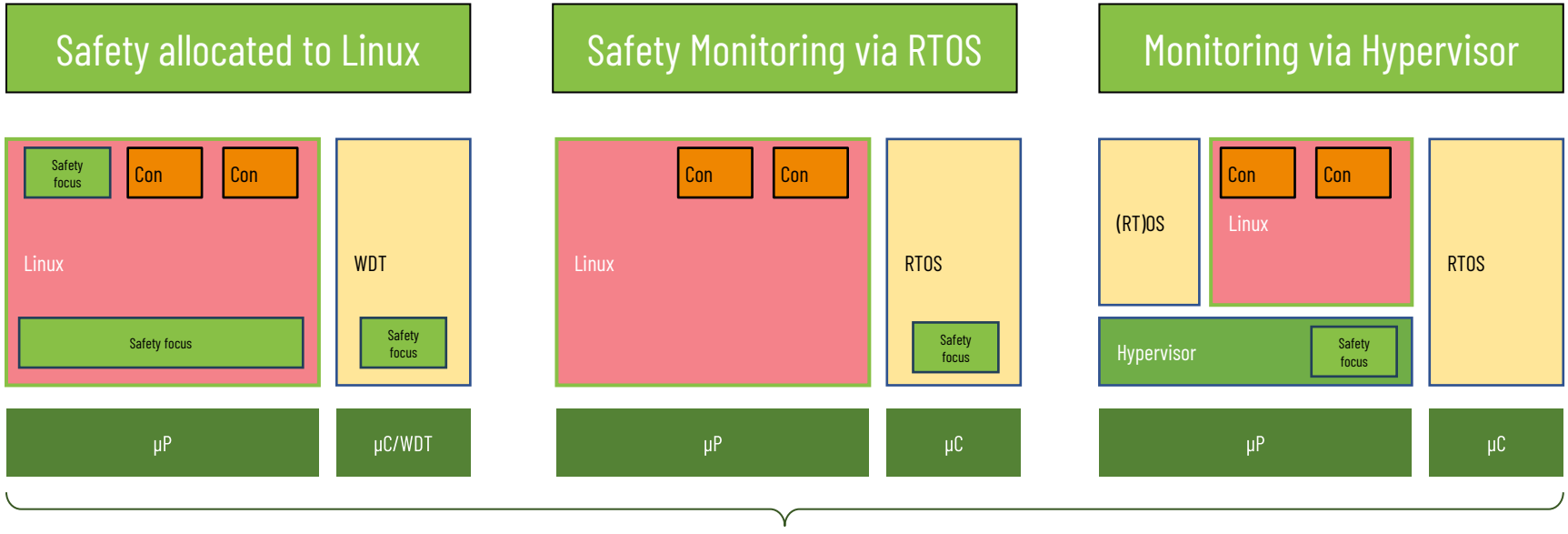
The Two Perspectives of …
Enabling Linux in Safety Applications

„Safe Linux“ is not „safety Linux“

Safety allocated to the system where Linux supports the safety application

Safety allocated to Linux as safety-critical element

Typical concepts and approaches



Watchdog is an essential element in various concepts

External Watchdog

- The challenge-response watchdog serves as the “safety net” for the safety-critical workload
- The concept is widely used in Automotive and other industrial applications
- It can be used as an iterative approach to assign more safety-critical functionality to Linux

With a proper system design the watchdog will never need to trigger the “safe state”.

Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units

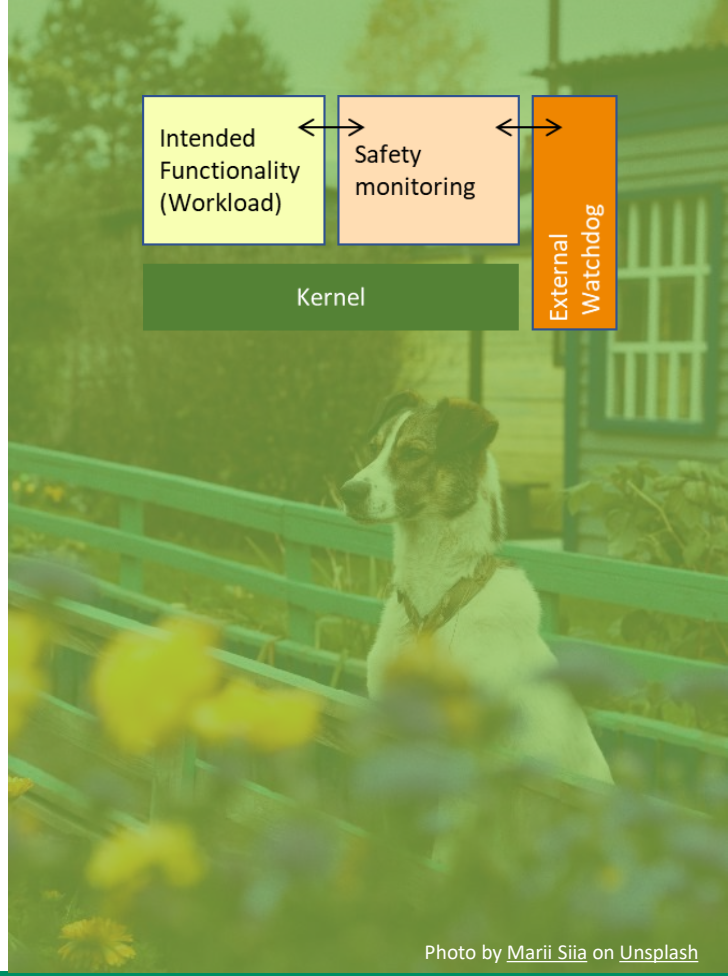
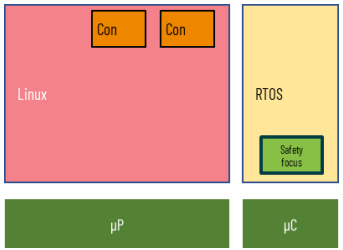


Photo by Marii Siia on Unsplash

Linux in (software-defined) cars beyond IVI

	In the wild & past	Under development	Future?
Conservative	• Rear View Camera • Tell tales (IC Warnings) • E-Mirror • Surround View	• Interior Monitoring • ADAS L2	• ADAS L4
Aggressive	• ADAS L2+ systems 	• ADAS L4	• ADAS L3? (cost driver)

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.

ELISA
Enabling Linux in Safety Applications

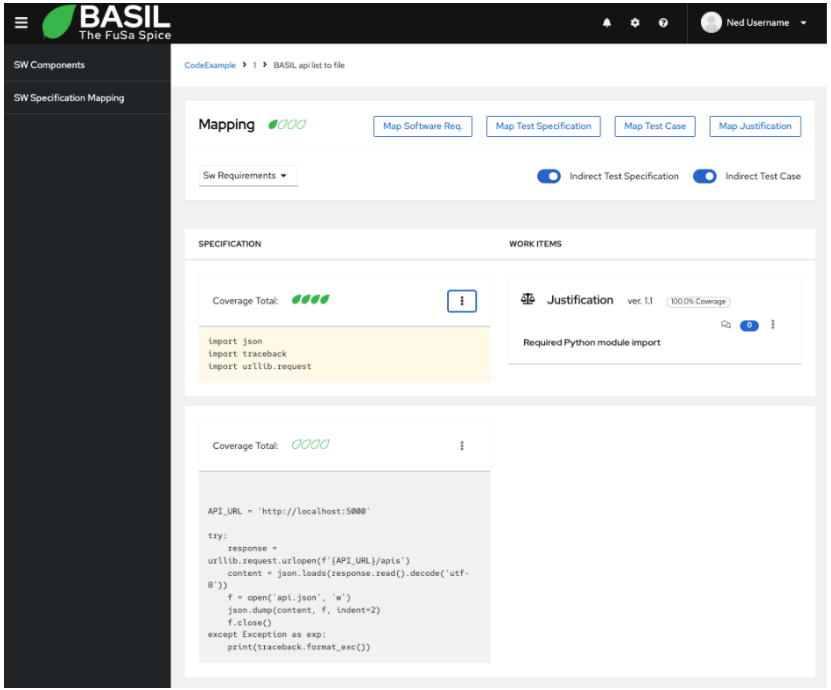
Open Source Requirement Tools

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.

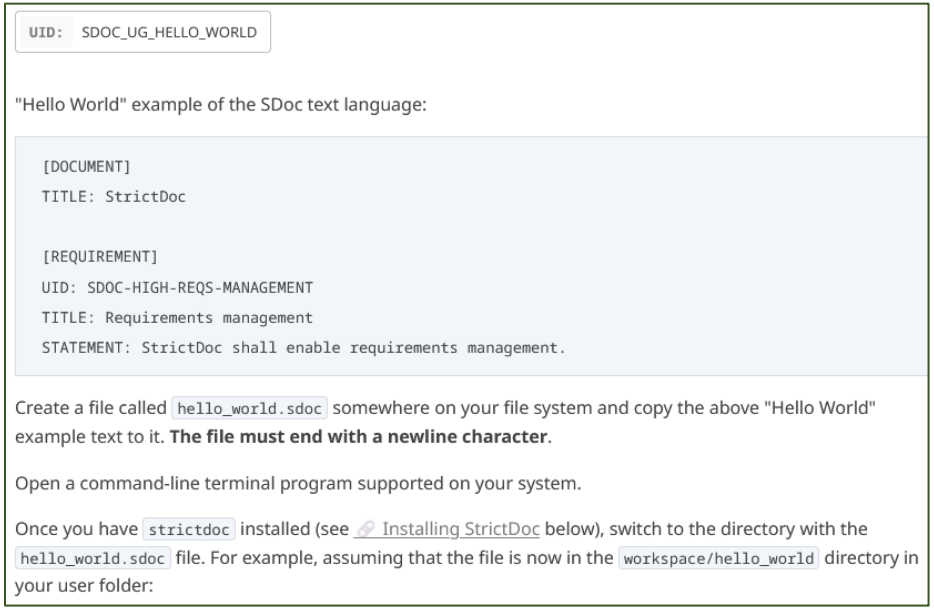
Used e.g. by: ELISA project



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”



`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:

Used e.g. by: Zepyhr project

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

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://doorstop.readthedocs.io | https://sphinx-needs.readthedocs.io | https://github.com/NASA-SW-VnV/fret
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

```
1498  /**
1499   * SPDX-Req-ID: [TODO automatically generate it]
1500   * SPDX-Req-Text:
1501   * trace_set_clr_event - enable or disable an event within a system
1502   * @system: system name (NULL for any system)
1503   * @event: event name (NULL for all events, within system)
1504   * @set: 1 to enable, 0 to disable (any other value is invalid)
1505   *
1506   * This is a way for other parts of the kernel to enable or disable
1507   * event recording.
1508   *
1509   * sequence of events:
1510   * 1) retrieve the global tracer
1511   * 2) locks the global event_mutex
1512   * 3) invokes __ftrace_set_clr_event_nolock
1513   * 4) unlocks the global event_mutex
1514   *
1515   * Returns 0 on success, -ENODEV if the global tracer cannot be retrieved,
1516   * -EINVAL if the parameters do not match any registered events, any other
1517   * error condition returned by __ftrace_set_clr_event_nolock
1518   */
1519  int trace_set_clr_event(const char *system, const char *event, int set)
1520  {
```

- 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 identifyin
  design elements to be docu
  starting from the pre-existin

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

ELISA
Enabling Linux in Safety Applications

Summary & Call to Action

Conclusion & Call to Action

- Open source is inevitable in safety-critical systems
- Companies must engage with communities now
- Bridging formal standards with open development
- Join initiatives like ELISA to shape the future
- Just show up – The projects are open!

ELISA
Enabling Linux in Safety Applications

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

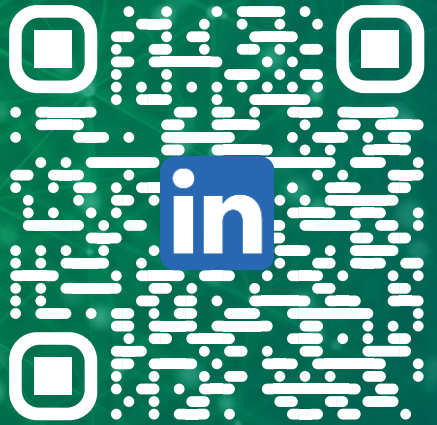
Contribute to Tools and Docs on GitHub

Participate in Working Groups

Attend Events

ELISA
Enabling Linux in Safety Applications

Thank you!



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

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/