
ELISA
Enabling Linux in
Safety Applications
WORKSHOP
ELISA Workshop
Munich, Germany
November 18-20, 2025
Co-hosted with Red Hat

# Exploring possibilities for integrating StrictDoc with ELISA's requirements template approach for the Linux kernel

or: "RE: Introducing SW Requirements in the Linux kernel development process"

Tobias Deiminger, Linutronix GmbH

Stanislav Pankevich, Reflex Aerospace GmbH

ELISA Enabling Linux in Safety Applications

WORKSHOP

# Introductions

**Tobias Deiminger**: Software Engineer, Linutronix GmbH

- Developing security features for and driving security certification of Linutronix IGLOS
- OSSW: Contributor to StrictDoc, Debian, misc projects for upstreaming patches

**Stanislav Pankevich**: Software Engineer, Reflex Aerospace GmbH

- Satellite software, software systems engineering
- OSSW: StrictDoc documentation tool, ReqIF Python lib, Mull mutation testing system
- WG: SPDX Functional Safety working group

# Agenda

- StrictDoc: Introduction to the project
- Traceability use case at Linutronix
- Linux kernel showcase and hands-on demo

# Motivation behind the StrictDoc project

"Every hard engineering problem can be solved with an infinite amount of cash."

How much cash is needed to **bring requirements to open source software**?

Ok, quite some cash but also: culture, methodology, and tools.

# Requirement statement/rationale examples

**STATEMENT example — WHAT**

*The `sem_wait()` function **shall lock the semaphore** referenced by sem by performing a semaphore lock operation on that semaphore. If the semaphore value is currently zero, then…*

https://pubs.opengroup.org/onlinepubs/9799919799/functions/sem_wait.html

Ideally, requirements should also have a **`UUID`** for easier linking to other elements.

**RATIONALE example — WHY**

*The `nanosleep()` function specifies that the system-wide clock `CLOCK_REALTIME` is used to measure the elapsed time for this time service. However, with the introduction of the monotonic clock `CLOCK_MONOTONIC` **a new relative sleep function is needed** to allow an application to take advantage of the special characteristics of this clock.*

https://pubs.opengroup.org/onlinepubs/9799919799/functions/clock_nanosleep.html

# Culture — Bridging requirements with source code

- Most documentation on GitHub — README How-tos and API reference
  - The **What** and **Why** are less obvious due to a focus on the How and implementation details.
- Design/development phase vs audit phase →
  - Requirements enable the safety and security assessments but equally important:
  - Make requirements really useful for developers **while** and **before** any code is written
    - **Not only after**!
- Culture of testing → Culture of requirements (extended Virtuous Cycle)
  - Simple test programs in projects from 1990-2000s, if existed, very pretty basic and chaotic.
  - Nowadays there are test methods and frameworks, everyone knows how to do testing.
  - Is the same happening with requirements, i.e., **REQ** → RED → **GREEN** → **REFACTOR**?

# Methodology — Traceability mechanics

- Working with the large numbers of requirements is not easy
  - Good structure based on functional analysis/partitioning.
- Linux's System-Subsystem-File → Integrate into user project specs:
  - Example: Satellite → Onboard Data Handling Subsystem → Operating System Component → Linux → Timers → `clock_nanosleep()/sys_clock_nanosleep()`.
- Distill the requirements/intent from the other elements:
  - Requirements — What-Why → Intent
  - API reference — Interface
  - Design and architecture — Implementation details
  - User manual — How-to instructions
  - Other: LICENSE, contributors, config, meta information…

**Exercise**:

Grab a marker and find:

Requirements vs other aspects

in your favourite Linux subsystem/module description.

# Tools — Integrating with the existing tools

- Technology gaps:
  - 'Big' OSSW players, such as, Sphinx, Doxygen do not support traceability out of the box.
  - Sphinx – Technical documentation websites. Building block: document/prose.
  - Doxygen – API documentation websites. Building block: Source file with comment markers.
  - Commercial SW, e.g., Confluence, does not trace to SW source and other artifacts.
- GitHub gist: 18 tools for requirements traceability of various maturity
  - At least 5 Sphinx extensions on GitHub to add traceability (+a few in-house)
  - Doxygen is adding requirements traceability but requirements have to come from another tool
- How to make these tools work together?

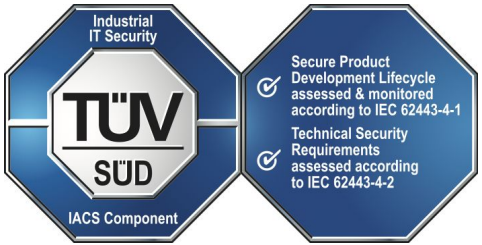
# StrictDoc tool

- Created in 2019, inspired by Doorstop
- Apache 2 license, 1.9K pull requests, 5K+ commits, 30K+ LOC
- In a nutshell:
  - Let's cut prose and code into atomic nodes, give them UUIDs, and link them together
- Key highlights:
  - Connecting docs, requirements, source and test code, test reports, coverage.
  - Web-based requirements editor.
  - SDoc format for storing requirements with metadata. Internal representation is a graph.
  - Other formats can be read or written. Native ReqIF bi-directional interface.
  - RST export for interfacing with Sphinx. Possible direction: sphinx-strictdoc plugin.
  - Work with the SPDX FuSa WG. Establishing the equivalence between SPDX and SDoc.

# Traceability use case at Linutronix

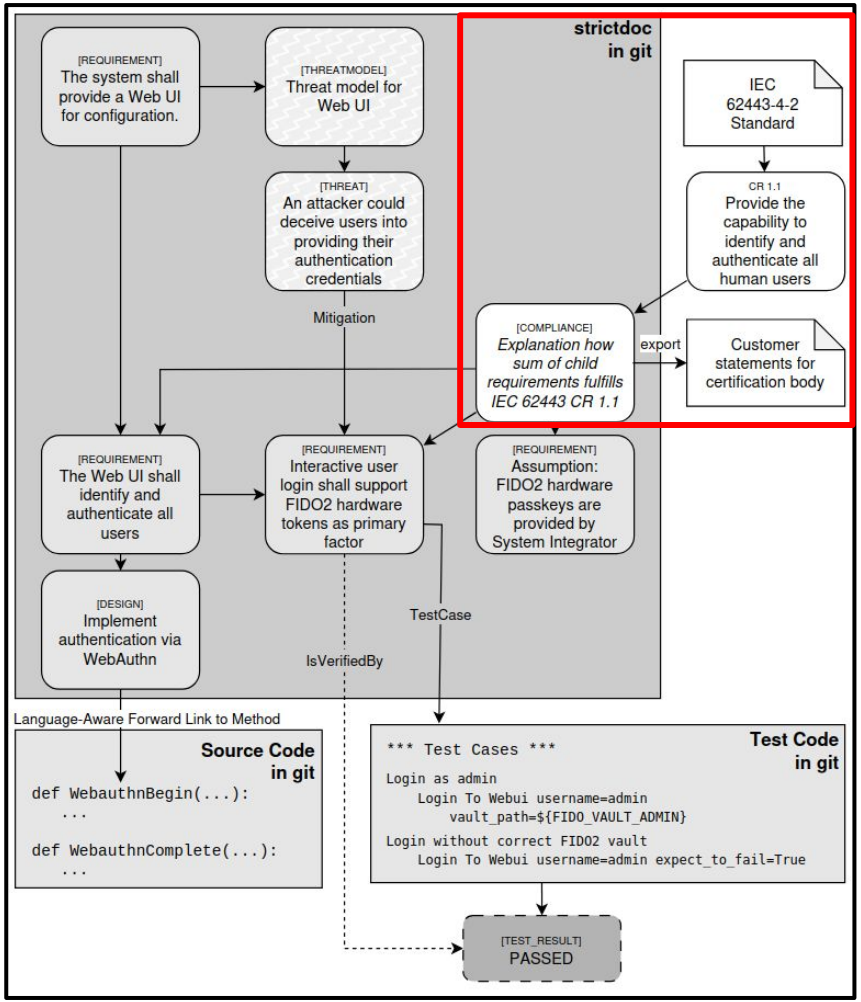

- Technical documentation for IGLOS, a secure Linux-based OS for industrial use
- Started using and contributing to StrictDoc in 2024
- Edited via Web UI or text editors, reviewed in GitLab MRs. HTML export deployed to an internal web server. A diff-UI supports requirement reviews.
- Certification according to IEC 62443-4-2 accomplished, EU CRA upcoming
- See blog post [1]

[1] https://www.linutronix.de/blog/From-Code-to-Compliance-Part1-IEC-62443-Certification-with



# Traceability model

- Structure based on arc42, extended with requirements, compliance matrix, threat model, and user guide
- Requirements trace to Robot/pytest tests and GitLab reviews
- Audit focused on StrictDoc "Compliance Matrix" document including conformity statements
- Interface to external standards by converting their outline to requirement stubs in *.sdoc



# Linux kernel proposal

- Starting point is ELISA's Linux Kernel Requirements Template
- Based on SPDX-* tags: Looks compatible with StrictDoc's capabilities
- **Challenge: Can the proposal be implemented?**

# Proof of concept

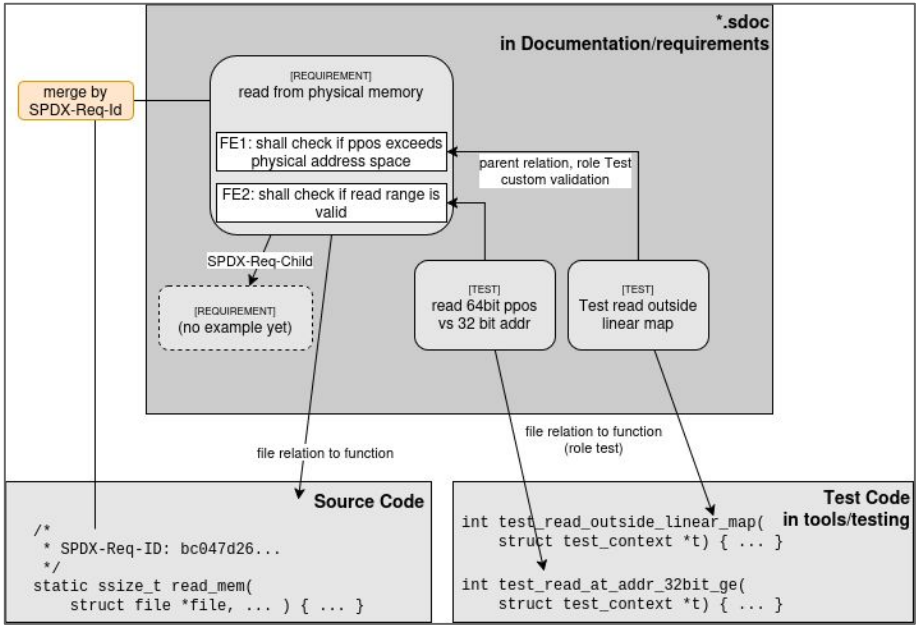
- Most of the requested features were already provided by StrictDoc
- Identified deltas, implemented them, many are merged
- The source code is a fragment of Linux source tree with the patches by ELISA applied
- Using .sdoc for sidecar requirements files
- POC available on GitHub, rendered to GitHub Pages

# Hands-on demo

https://github.com/strictdoc-project/linux-strictdoc

# Traceability model: Linux

- Simpler: ELISA proposed only Requirements and Source Code functions as traceable items
- Our proposal has tracing with tests
- Things like POSIX compliance matrix and linking test reports to be discussed



```
[REQUIREMENT]
MID: d945eb59099b0359858b0b3f3aeb15fd9c188ffabc06b5d9d3fb4e72e645c71e
SPDX-Req-Sys: Character Drivers and Misc
TITLE: write_mem

[REQUIREMENT]
MID: 97c02e40efb914aa7368b8fa6025f467d7dc51049b3df9223ef45571e8eebaf2
SPDX-Req-Sys: Character Drivers and Misc
TITLE: mmap_mem

[REQUIREMENT]
MID: ed61250a1e26264928301bb1f6c04f313dab998ca99f53fe2450fcb11d995073
SPDX-Req-Sys: Character Drivers and Misc
TITLE: memory_lseek
COMMENT: >>>
The memory devices use the full 32/64 bits of the offset, and so we cannot
check against negative addresses: they are ok. The return value is weird,
though, in that case (0).
<<<
COMMENT: >>>
Also note that seeking relative to the "end of file" isn't supported:
it has no meaning, so passing orig equal to SEEK_END returns -EINVAL.
<<<

[REQUIREMENT]
MID: 53a11329f9090a6c2be01569f59e52f2d4c0c84d0763ae3c0fc73a0c8ce5e705
SPDX-Req-Sys: Character Drivers and Misc
TITLE: open_port

[REQUIREMENT]
MID: ebb05eb27aa17e09a44d76f73e383705fe1781a36ef4cede5859bfdc210a7221
SPDX-Req-Sys: Character Drivers and Misc
TITLE: memory_open

[[/SECTION]]

[[SECTION]]
Documentation/requirements/charmisc.sdoc                    73,11         13%

[TEST]
MID: selftests/devmem:open_devnum
TITLE: memory_open FE_3
DESCRIPTION: Test open /dev/mem provides the correct min, maj
RELATIONS:
- TYPE: Parent
  VALUE: ebb05eb27aa17e09a44d76f73e383705fe1781a36ef4cede5859bfdc210a7221
  ROLE: Test
- TYPE: File
  ROLE: Test
  VALUE: tools/testing/selftests/devmem/tests.c
  FUNCTION: test_open_devnum
- TYPE: File
  ROLE: Test
  VALUE: tools/testing/selftests/devmem/devmem.c
  LINE_RANGE: 33, 36
Documentation/requirements/charmisc.sdoc                    108,1         32%
```

```
/**
 * SPDX-Req-ID: ebb05eb27aa17e09a44d76f73e383705fe1781a36ef4cede5859bfdc210a7221
 * SPDX-Req-Text:
 * memory_open - set the filp f_op to the memory device fops and invoke open().
 * @inode: inode of the device file.
 * @filp: file structure for the device.
 *
 * Function's expectations:
 * 1. This function shall retrieve the minor number associated with the input
 *   inode and the memory device corresponding to such minor number;
 *
 * 2. The file operations pointer shall be set to the memory device file operations;
 *
 * 3. The file mode member of the input filp shall be OR'd with the device mode;
 *
 * 4. The memory device open() file operation shall be invoked.
 *
 * Assumptions of Use:
 * 1. The input inode and filp are expected to be non-NULL.
 *
 * Context: process context.
 *
 * Return:
 * * 0 on success
 * * %-ENXIO - the minor number corresponding to the input inode cannot be
 *   associated with any device or the corresponding device has a NULL fops
 *   pointer
 * * any error returned by the device specific open function pointer
 *
 * SPDX-Req-End
 */
static int memory_open(struct inode *inode, struct file *filp)
{
        int minor;
        const struct memdev *dev;

        minor = iminor(inode);
        if (minor >= ARRAY_SIZE(devlist))
                return -ENXIO;

        dev = &devlist[minor];
        if (!dev->fops)
                return -ENXIO;

        filp->f_op = dev->fops;
        filp->f_mode |= dev->fmode;

        if (dev->fops->open)
                return dev->fops->open(inode, filp);

        return 0;
}

static const struct file_operations memory_fops = {
drivers/char/mem.c                                          972,1         95%
```

# Further work

- Connect requirements, tests and test reports
  - Test report format for Linux tests has to be evaluated and integrated.
  - Make function expectations individually traceable with composite requirement
- Use SPDX-* fields for tests as well
- Usage of SPDX-REQ-CHILD and SPDX-REQ-REF to be clarified
- Integration with Sphinx and kernel-doc
- Performance tuning for very large projects
- Can you think of useful traceability metrics and validations?

# Contact information

- Tobias Deiminger: tobias.deiminger@linutronix.de
- Stanislav Pankevich: s.pankevich@gmail.com
- StrictDoc project: https://github.com/strictdoc-project/strictdoc

Join StrictDoc Office Hours: **Every Tuesday**, 🕔 17:00–18:00 CET

# Licensing of Workshop Results

All work created during the workshop is licensed under Creative Commons Attribution 4.0 International (CC-BY-4.0) [https://creativecommons.org/licenses/by/4.0/] by default, or under another suitable open-source license, e.g., GPL-2.0 for kernel code contributions.

You are free to:

- Share — copy and redistribute the material in any medium or format
- Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

/

/Documentation/

/Documentation/requirements/

**Character Drivers and Misc**
charmisc.sdoc

**Tracing**
tracing.sdoc

SECTION

# 2. Low-Level Requirements

SECTION

## 2.1. devmem

SECTION

### 2.1.1. Requirements

REQUIREMENT

#### 2.1.1.1. read_mem

SPDX-Req-ID: bc047d26a37376a1adb7f95b6baeebde9865042d0ff9e6fdc448d4db3eb627ef

SPDX-Req-Sys: Character Drivers and Misc

RELATIONS (Child): → selftests/devmem:read_at_addr_32bit_ge read_mem FE_1 (Test)
→ selftests/devmem:read_outside_linear_map read_mem FE_2 (Test)
→ selftests/devmem:read_allowed_area read_mem FE_3.2 (Test)
→ selftests/devmem:read_allowed_area_ppos_advance read_mem FE_4 (Test)
→ selftests/devmem:read_restricted_area read_mem FE_3.3, FE_3.3.1, FE_3.2.2 (Test)
→ selftests/devmem:read_secret_area read_mem FE_???

RELATIONS (File): </> drivers/char/mem.c, *lines: 78-216*, function read_mem()

SPDX-Req-Text:

read_mem - read from physical memory (/dev/mem).
@file: struct file associated with /dev/mem.
@buf: user-space buffer to copy data to.
@count: number of bytes to read.
@ppos: pointer to the current file position, representing the physical address to read from.

This function checks if the requested physical memory range is valid and accessible by the user, then it copies data to the input user-space buffer up to the requested number of bytes.

# Setup and usage

```
pipx install strictdoc
git clone https://github.com/strictdoc-project/linux-strictdoc
cd linux-strictdoc
strictdoc export .            # validate and render to HTML
strictdoc manage auto-uid .   # hash generation, drift detection
```

# File structure

```
.
├── Documentation
│   └── requirements
│       ├── charmisc.sdoc          # side-car
│       └── tracing.sdoc           # side-car
├── drivers
│   └── char
│       └── mem.c                  # Linux code with inlined LLRs
├── kernel
│   └── trace
│       └── trace_events.c         # Linux code with inlined LLRs
├── strictdoc_config.py            # StrictDoc project
configuration
└── tools
    ├── requirements
    │   └── validation_plugin.py   # custom requirement validations
    └── testing
        └── selftests
            └── devmem
                └── tests.c        # tests for /dev/mem LLRs
```