Achieving end-to-end traceability in software projects using open source tools

Roadmap, challenges, and a reference implementation

Stanislav Pankevich, Reflex Aerospace GmbH
Software Product Assurance Conference 2025
ESA – ESTEC, Netherlands
22-25.09.2025

About the speaker

- Software Lead at Reflex Aerospace, a satellite manufacturing startup in Berlin.
- Several startups before, mostly small companies.

My worlds:

1) Space industry, spacecraft onboard software development.
2) Open source software development:
    a) **StrictDoc** documentation tool and **ReqIF** Python library
    b) **Mull** fault injection system for C/C++ and **FileCheck.py,** Python port for LLVM FileCheck.

This presentation is about a personal project. The views expressed are my own and do not represent those of my employer.

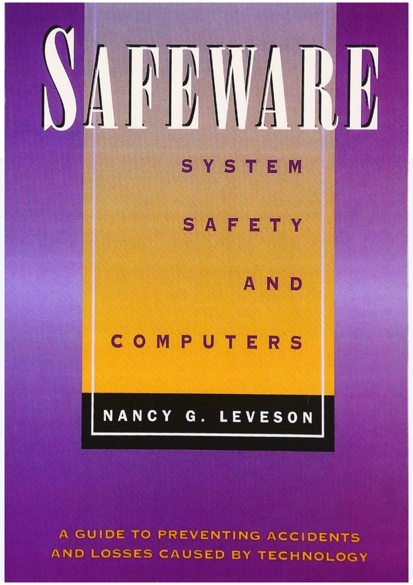
Agenda

- What is traceability, and why does it matter?
- The challenges of implementing traceability with open source software
- The StrictDoc project
- Further work and closing remarks

Motivation for this work: Bridging requirements and software

“The vast majority of accidents in which software was involved can be traced to requirements flaws and, more specifically, to incompleteness in the specified and implemented software behavior — that is, incomplete or wrong assumptions about the operation of the controlled system or required operation of the computer and unhandled controlled-system states and environmental conditions. Although coding errors often get the most attention, they have more of an effect on reliability and other qualities than on safety.”

— ***Nancy Leveson***, *Safeware: System Safety and Computers (1995)*


SAFEWARE
SYSTEM
SAFETY
AND
COMPUTERS
NANCY G. LEVESON
A GUIDE TO PREVENTING ACCIDENTS AND LOSSES CAUSED BY TECHNOLOGY

- How can developers work more effectively with requirements during software development?
- How to connect code to requirements?
- Which tools are needed to support this connection?

Requirement = (Statement + meta information)

Common fields:
`TITLE`, `STATEMENT`, `RATIONALE`, `COMMENT`

Unique identifiers:
`MID` / `UID` (machine- vs human-readable)

Additional meta information:
`STATUS`, `TAGS`, `LEVEL`, `PREFIX`

Industry-specific meta information:
e.g., `ASIL(A,B,C,D)`, `VERIFICATION (R,A,I,T)`

Relations/roles:
e.g., parent, child, parent/refines, etc.

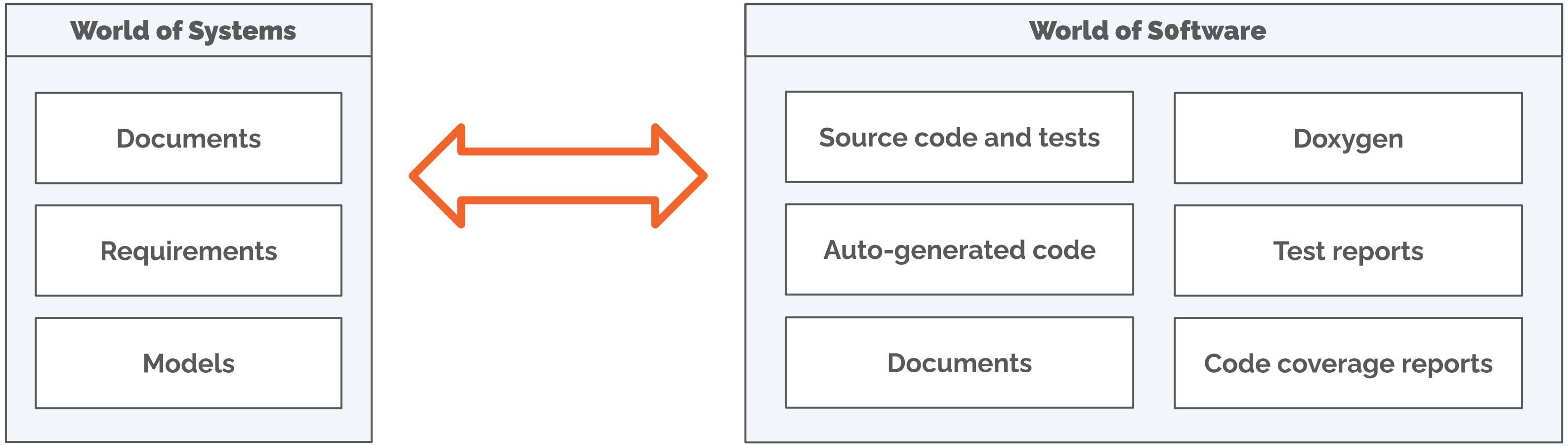
Example of a real-world requirement (adopted from ECSS-E-ST-40C):

UID: ECSS-E-ST-40_0860055

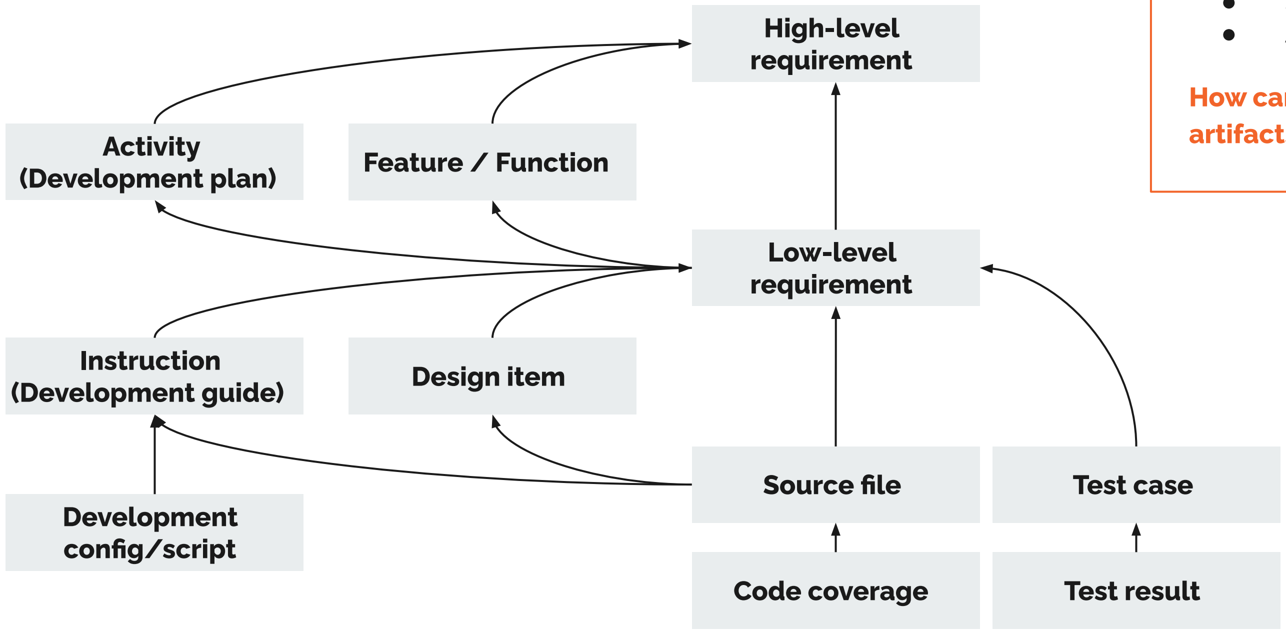
```
ECSS_REQ_ID: 5.4.2.3a

TITLE: Construction of a software logical model

STATEMENT: The supplier shall construct a logical
model of the functional requirements of the software
product.

EXPECTED_OUTPUT: Software logical model [TS, SRS; PDR].
```

What is Traceability?

In software development, traceability refers to the ability to track and link artifacts across the software development lifecycle. This typically means creating explicit connections between requirements, design, code, tests, and documentation.



Traceability graph and its nodes



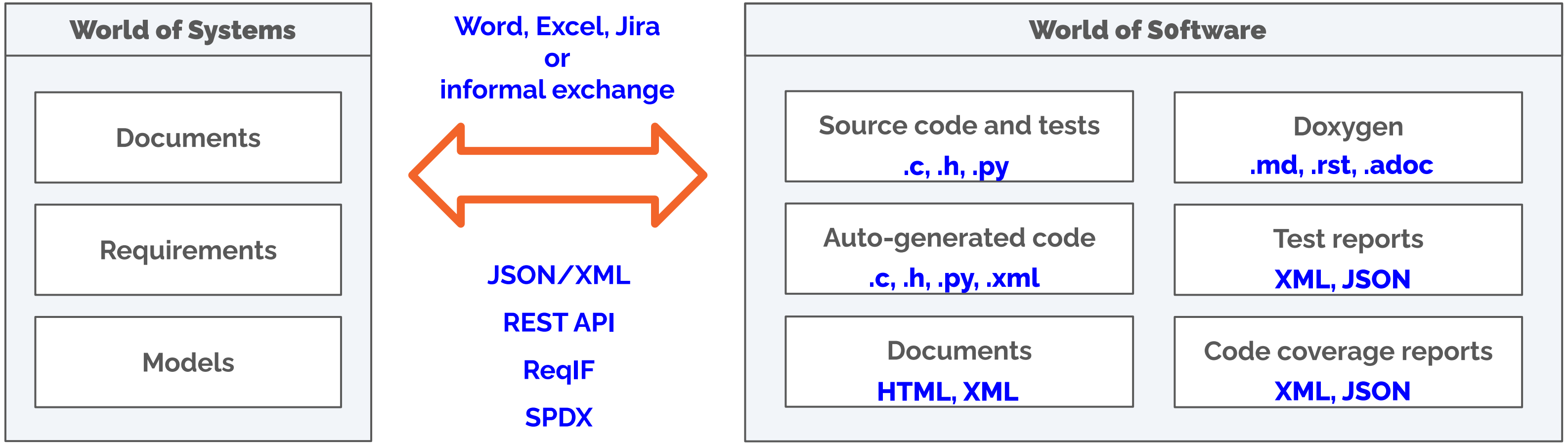
Mixed content from multiple tools:

- Documents
- Source code elements
- Auto-generated artifacts

How can we connect all these artifacts together?

Traceability — Three challenges

1) There is no tool that connects all artifacts together in a single representation.
2) There is limited open-source tooling for the World of Systems.
3) Exchanging data and metadata between tools and formats. Which human-readable format to use?



StrictDoc — Open source requirements tool

- Spare-time project for 2 core developers. 22 contributors so far.
- Created in 2019.
- Inspired by Doorstop's OSS approach to Git-based requirements management.
- Written in Python.
- Apache 2 license.
- 1.8K pull requests, 5K+ commits, 30K+ LOC.

Key highlights:

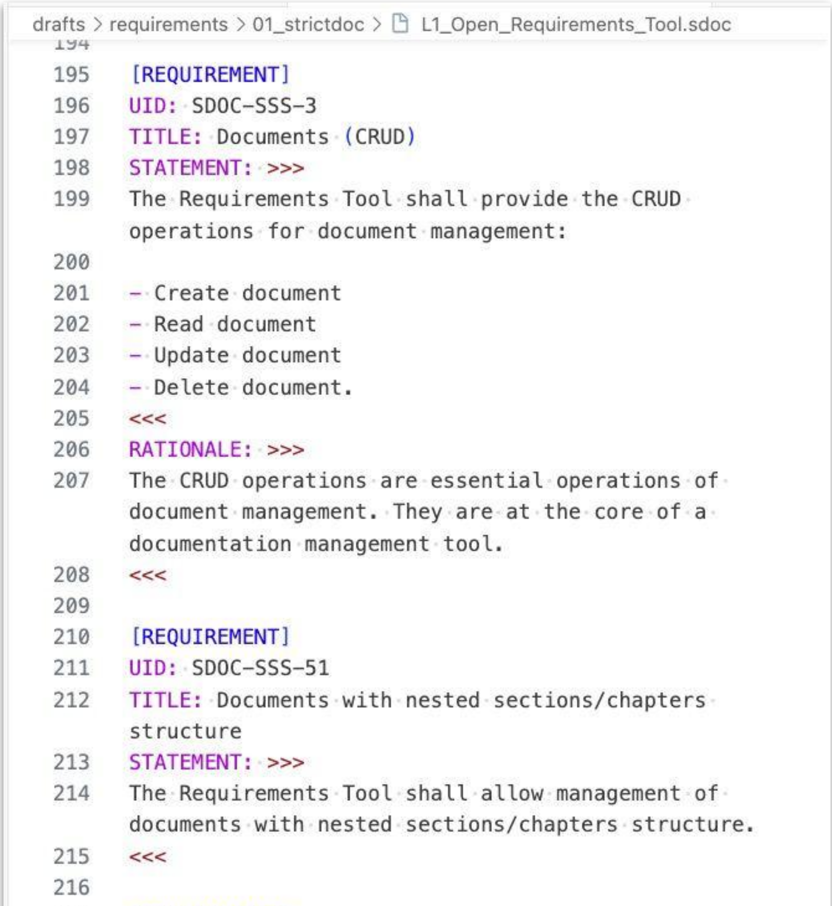
- **2020-2022**: Documentation generator, HTML export, ReqIF, tracing source files to requirements, document grammar and custom fields, traceability graph validations.
- **2023**: Web-based user interface. The HTML-to-PDF feature for publishing documents.
- **2024**: Language-aware traceability to C/C++ and Python.
- **2025**: Extending traceability to test and coverage reports, preparing for safety-related qualification.

.SDoc format

- Starting point: Format to support text and metadata.
- YAML frontmatter does not scale to large documents.
- RST directives do not support nested metadata.
- JSON is less human-readable, and so are HTML/XML.
- Nesting content in a document with 4+ chapter levels does not scale visually.

SDoc ('strict-doc') is a practical compromise inspired by:

- YAML – nested meta information fields
- TOML – keys in square brackets
- XML/HTML – opening/closing tags for nested content
- ASN.1 – Capital letters.


drafts > requirements > 01_strictdoc > L1_Open_Requirements_Tool.sdoc

```
[REQUIREMENT]
UID: SDOC-SSS-3
TITLE: Documents (CRUD)
STATEMENT: >>>
The Requirements Tool shall provide the CRUD
operations for document management:

- Create document
- Read document
- Update document
- Delete document.
<<<
RATIONALE: >>>
The CRUD operations are essential operations of
document management. They are at the core of a
documentation management tool.
<<<

[REQUIREMENT]
UID: SDOC-SSS-51
TITLE: Documents with nested sections/chapters
structure
STATEMENT: >>>
The Requirements Tool shall allow management of
documents with nested sections/chapters structure.
<<<

```

Two ways to link code and requirements

1. **Relation markers: Link code to requirements.**
   Use when can control **source code** 🛠️ directly.

2. **Forward relations: Link requirements to source code.**
   Use when **source code** 🔒 cannot be modified.

```python
class ProjectStatisticsGenerator:
    def export(
        self,
        project_config: ProjectConfig,
        traceability_index: TraceabilityIndex,
    ) -> Markup:
    """
    Export project statistics to an HTML page.

    @relation(SDOC-SRS-97, scope=function)
    """
```

```
[REQUIREMENT]
UID: SDOC-SRS-97
TITLE: Project statistics generator
STATEMENT: StrictDoc shall generate project statistics.
RELATIONS:
- TYPE: File
  VALUE: strictdoc/generators/project_statistics.py
  FUNCTION: ProjectStatisticsGenerator.export
```

Attach metadata to source code

- The comments are parsed as SDoc nodes.
- An auto-generated document is created from a provided template.

Use case example:

- Generate a test specification from source code.
- The test spec items are linked to requirements.

```
/**
 * \brief Test example
 *
 * @relation(REQ-1, scope=function)
 *
 * INTENTION: ...
 *
 * INPUT: ...
 *
 * EXPECTED_RESULTS: ...
 */
TEST_CASE("Test example", testExample)
{
    ...
}
```

StrictDoc — Two workflows

1

Command-line / IDE workflow

- Export documents.
- The default export is static HTML.

```
strictdoc export <input_dir>
```

2

Web interface

- Work with editable documentation.
- Starts a web server. Static HTML that is editable.

```
strictdoc server <input_dir>
```

SDoc and other formats

- StrictDoc supports exporting to **PDF**, **Excel**, **ReqIF**, **SPDX**, **RST**, **JSON**, and importing from **Excel** and **ReqIF**.
- The most developed format for bi-directional exchange from/to SDoc is ReqIF.

ReqIF (mostly implemented)

- Python ReqIF parser/unparser is part of StrictDoc.
- SDoc and ReqIF are mostly compatible.
- SDoc export is ReqIF schema-compliant.
- ReqIF uses XHTML for "rich" markup → StrictDoc supports the RST, HTML, pure Text markup modes.
- Future work: Handling images and tables.

SPDX (ongoing)

- SPDX is parsed using the official SPDX libraries.
- The SPDX Functional Safety Working Group (FuSa) is working on standardizing the requirements model.
- Tech. exchange between StrictDoc and SPDX FuSa:
- Can SPDX be used for creating/editing requirements, not just auditing?
- SDoc as a valid SPDX representation: Establishing bi-directional equivalence.

When a general algorithm is hard to achieve, a custom Python converter always works.

Use case: The Zephyr project

- The Zephyr Safety Working Group is working to make the Zephyr RTOS certifiable for use in safety-related projects:
  - Zephyr Safety Overview and Safety FAQ
- StrictDoc is used in three ways:
  - **Engineering:** Technical requirements are captured in SDoc files in a dedicated Git repository: https://github.com/zephyrproject-rtos/reqmgmt.
  - **Safety and Quality Management:** Capturing plans, verification checklists and guidelines. Safety-related IEC 61508 compliance documentation.
  - **Safety Assessors:** The assessment checklist captures the assessor's expectations and serves as the entry point for finding compliance information in the Zephyr documentation.

See also our FOSDEM 2024 talk: Application of the SPDX Safety Profile in the Safety Scope of the Zephyr Project.

Use case:
Operating system development at Linutronix

- Technical documentation for IGLOS, a secure Linux-based OS for industrial use, started in 2024 from scratch using StrictDoc (no commercial RE tools).
- Structure based on arc42, extended with requirements, compliance matrix, threat model, and user guide.
- 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.
- Requirements trace to Robot/pytest tests and GitLab reviews.
  Code accessed via Git submodules. Minor scripting for custom import/export.
  External PDFs interfaced by converting ToC to *.sdoc.
- Certification according to IEC 62443-4-2 accomplished, EU CRA upcoming.
  Gap analysis by sending HTML/PDF exports to certification body.
  Audit focused on StrictDoc "Compliance Matrix" document including conformity statements.

IGLOS
SECURE BEACON
by LINUTRONIX

Industrial IT Security
TÜV SÜD
IACS Component

Secure Product Development Lifecycle assessed & monitored according to IEC 62443-4-1
Technical Security Requirements assessed according to IEC 62443-4-2

The potential use of StrictDoc in the space industry

- ECSS standards form the foundation of traceability in any European space project.
- ECSS standards are available as Doors and Excel imports.
- ReqIF export from Doors could be an option if the full ECSS content was needed, e.g., images and tables.

The following interfaces are readily available:

Interface 1:
Convert ECSS Excel to SDoc with Python

Convert ECSS Excel to SDoc with a Python script. Each ECSS document's requirements are stored in a dedicated SDoc file. The requirement metadata is preserved as-is.

Interface 2:
Work with ECSS DRDs encoded in SDoc format

The strictdoc-templates repository contains the examples of the ECSS-E-ST-40C DRD templates for:

- Software System Specification (SSS)
- Software Requirements Specification (SRS).

Further work

- Multi-user Git workflow
  - Concurrent use
  - Committing to Git from the web UI
  - User accounts
- Integration with other tools
- Analyzing and acting on requirements with AI
- Testing large multi-repo projects
- Traceability mechanics is a research topic in its own right:
  - The tooling is in place but how to connect requirements to software in the best way?
  - The easiest approach: connect requirements to whole source files.
  - How to join a parent project traceability graph with third-party OSS/OTS project graphs?

Closing remarks

1) The words of Systems and Software can be connected with open source software.
2) Translation between tools and formats is possible and is an ongoing topic.
3) Individuals and companies are invited to collaborate, with a special welcome to the space industry. 🚀

Can we make requirements an effective, systematic part of software development?

Contact information and relevant links

StrictDoc core team:

- Stanislav Pankevich, s.pankevich@gmail.com
- Maryna Balioura, mettta@gmail.com

Documentation:

- StrictDoc project
- StrictDoc roadmap
- StrictDoc large slide deck

Get in touch with the community:

- StrictDoc GitHub issues
- StrictDoc's mailing list and Discord channel
- StrictDoc office hours — Every Tuesday, 17:00–18:00 CET

StrictDoc's large slide deck:

Backup slides

Requirements as a tool

- Requirements capture project decisions.
- Requirements are the foundation of traceability.
  - Each requirement is uniquely identified.
  - Requirements can be linked to each other for formal tracking.
- Requirements can be used to manage work.
  - Requirements drive project organization and work breakdown.
  - Requirements define contracts and acceptance criteria.
- Requirements can be used for measurements.
- Requirements are powerful abstractions.

SYS-222 The spacecraft shall be designed and verified to achieve a minimum reliability of 99.7% over the mission duration, as defined from launch through end-of-life, including all nominal and contingency operations.

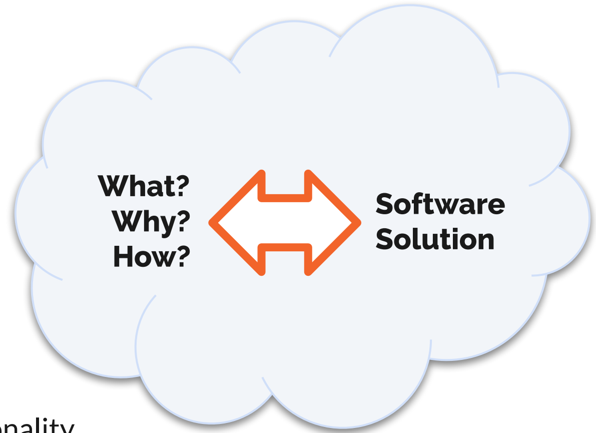
MIS-111 ← SYS-222

60% of requirements are already implemented by a contractor.

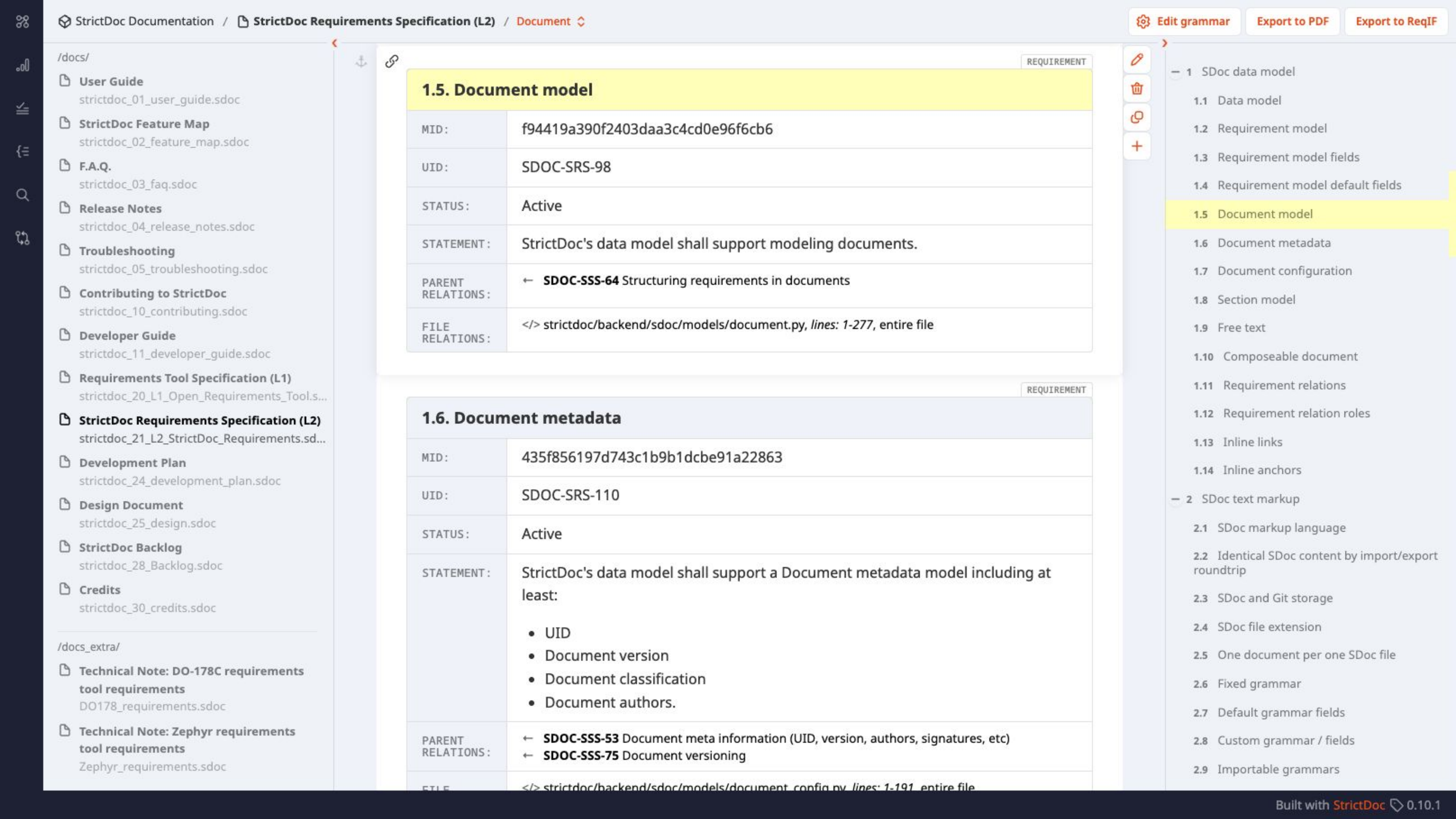
87% of requirements are verified by test.

Requirements define the 'public interface' of the work performed; the source code is the implementation.

Why does Traceability matter?



- **Connect intent with implementation**
  - What was requested is what was implemented.
- **Coverage information**
  - Reduce the chance of missing requirements or building incorrect functionality.
  - Ensure every requirement is tested and verified (forward traceability).
  - Ensure every feature or test is tied to an actual need (backward traceability).
- **Traceable development process and project memory**
  - Prove that contractual or legal obligations are fulfilled.
  - Onboard new engineers and preserve rationale over time.
  - Prevent duplication, scope creep, and orphaned work.
- **Impact analysis**
  - If a requirement changes, what else has to be changed?


StrictDoc Documentation / StrictDoc Requirements Specification (L2) / Document
Edit grammar
Export to PDF
Export to ReqIF

/docs/

- **User Guide** strictdoc_01_user_guide.sdoc
- **StrictDoc Feature Map** strictdoc_02_feature_map.sdoc
- **F.A.Q.** strictdoc_03_faq.sdoc
- **Release Notes** strictdoc_04_release_notes.sdoc
- **Troubleshooting** strictdoc_05_troubleshooting.sdoc
- **Contributing to StrictDoc** strictdoc_10_contributing.sdoc
- **Developer Guide** strictdoc_11_developer_guide.sdoc
- **Requirements Tool Specification (L1)** strictdoc_20_L1_Open_Requirements_Tool.s...
- **StrictDoc Requirements Specification (L2)** strictdoc_21_L2_StrictDoc_Requirements.sd...
- **Development Plan** strictdoc_24_development_plan.sdoc
- **Design Document** strictdoc_25_design.sdoc
- **StrictDoc Backlog** strictdoc_28_Backlog.sdoc
- **Credits** strictdoc_30_credits.sdoc

/docs_extra/

- **Technical Note: DO-178C requirements tool requirements** DO178_requirements.sdoc
- **Technical Note: Zephyr requirements tool requirements** Zephyr_requirements.sdoc

REQUIREMENT

1.5. Document model

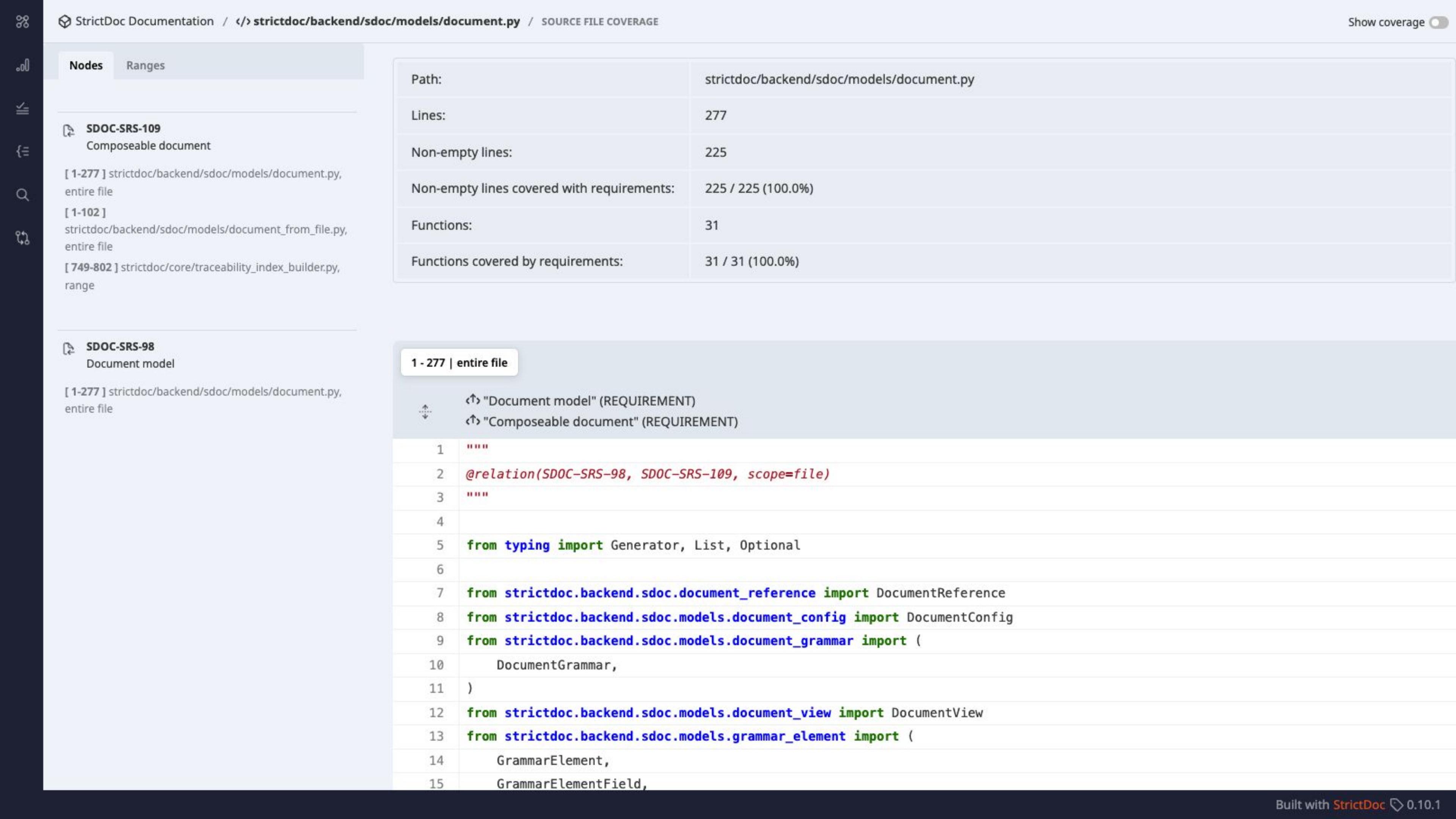
MID:	f94419a390f2403daa3c4cd0e96f6cb6
UID:	SDOC-SRS-98
STATUS:	Active
STATEMENT:	StrictDoc's data model shall support modeling documents.
PARENT RELATIONS:	← **SDOC-SSS-64** Structuring requirements in documents
FILE RELATIONS:	</> strictdoc/backend/sdoc/models/document.py, *lines: 1-277*, entire file

REQUIREMENT

1.6. Document metadata

MID:	435f856197d743c1b9b1dcbe91a22863
UID:	SDOC-SRS-110
STATUS:	Active
STATEMENT:	StrictDoc's data model shall support a Document metadata model including at least: • UID • Document version • Document classification • Document authors.
PARENT RELATIONS:	← **SDOC-SSS-53** Document meta information (UID, version, authors, signatures, etc) ← **SDOC-SSS-75** Document versioning
FILE	</> strictdoc/backend/sdoc/models/document_config.py, *lines: 1-191*, entire file

Built with StrictDoc 0.10.1


StrictDoc Documentation / </> strictdoc/backend/sdoc/models/document.py / SOURCE FILE COVERAGE

SDOC-SRS-109
Composeable document

[1-277] strictdoc/backend/sdoc/models/document.py, entire file

[1-102] strictdoc/backend/sdoc/models/document_from_file.py, entire file

[749-802] strictdoc/core/traceability_index_builder.py, range

SDOC-SRS-98
Document model

[1-277] strictdoc/backend/sdoc/models/document.py, entire file

Path:	strictdoc/backend/sdoc/models/document.py
Lines:	277
Non-empty lines:	225
Non-empty lines covered with requirements:	225 / 225 (100.0%)
Functions:	31
Functions covered by requirements:	31 / 31 (100.0%)

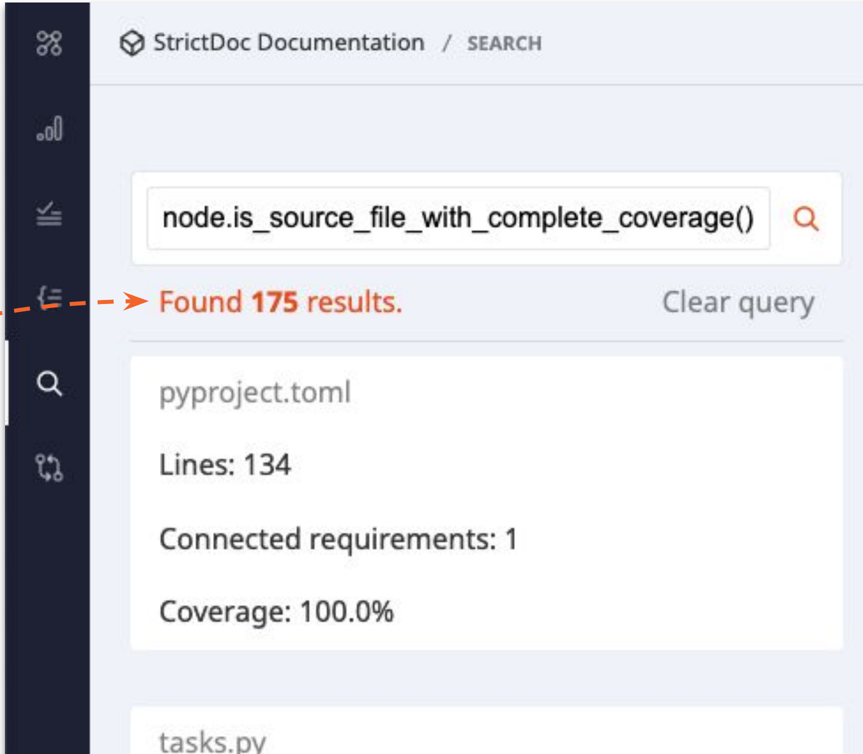
1 - 277 | entire file

"Document model" (REQUIREMENT)
"Composeable document" (REQUIREMENT)

```python
"""
@relation(SDOC-SRS-98, SDOC-SRS-109, scope=file)
"""

from typing import Generator, List, Optional

from strictdoc.backend.sdoc.document_reference import DocumentReference
from strictdoc.backend.sdoc.models.document_config import DocumentConfig
from strictdoc.backend.sdoc.models.document_grammar import (
    DocumentGrammar,
)
from strictdoc.backend.sdoc.models.document_view import DocumentView
from strictdoc.backend.sdoc.models.grammar_element import (
    GrammarElement,
    GrammarElementField,
```

Traceability metrics

- Two interconnected features: Statistics and Search.
- Writing project-specific statistics generators in Python.
- Calculate total numbers of entities.
- Calculate the number of present and missing relations.
- Each statistic can be queried individually.
- Inspired by the ECSS SW metrication HB:
  - Requirements coverage
  - Calculate a number of TBD/TBC.



node.is_source_file_with_complete_coverage()

Found 175 results.

pyproject.toml

Lines: 134

Connected requirements: 1

Coverage: 100.0%

tasks.py

Feedback from the user community

- Many users want the same feature set, many users want something else.
- As of Q3 2025, there are 65 open feature requests of various types.
- Features that are in high demand are implemented with higher priority.
- Tech keywords from the requested features and interfaces to other tools, new and recently implemented:

Excel

JavaScript LaTeX Doors

ReqIF

AsciiDoc Bash SPDX

Polarion

Ada

Markdown Rust JUnit TestIF

OpenAPI

Robot framework

OSLC

Capella MBSE Enterprise Architect