



ELISA
Enabling **Linux** in
Safety Applications

WORKSHOP

ELISA Workshop
Munich, Germany

November 18-20, 2025
Co-hosted with Red Hat





Who I am



Luigi Pellecchia



*Principal
Software Quality
Engineer*

In-vehicle OS
Red Hat

*Member of Technical
Steering Committee*

ELISA
(Linux Foundation)

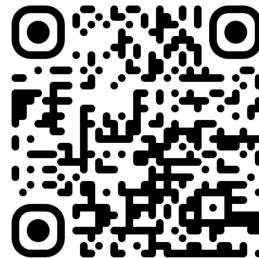
Agenda

- BASIL Overview
- What's new

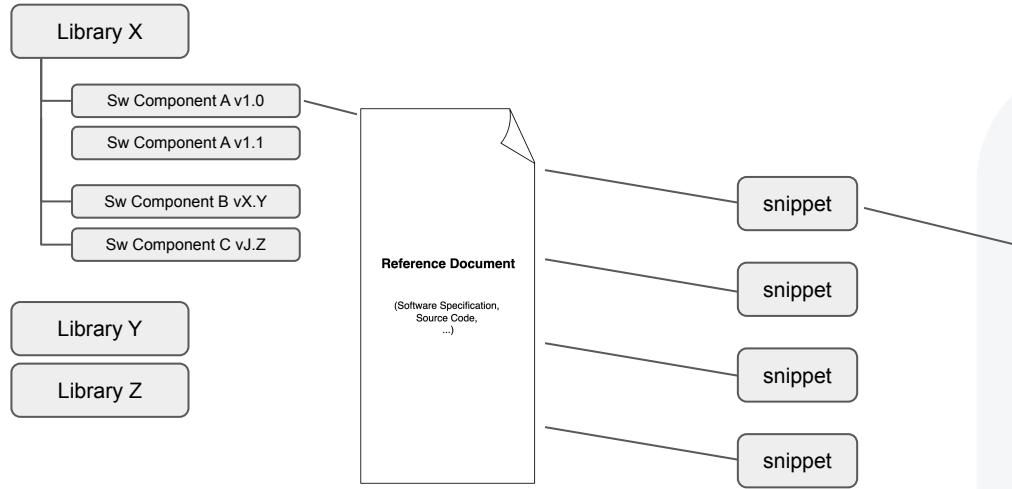
BASIL The FuSa Spice

Tool developed to manage software related work items, design their traceability towards specifications and ensure completeness of analysis

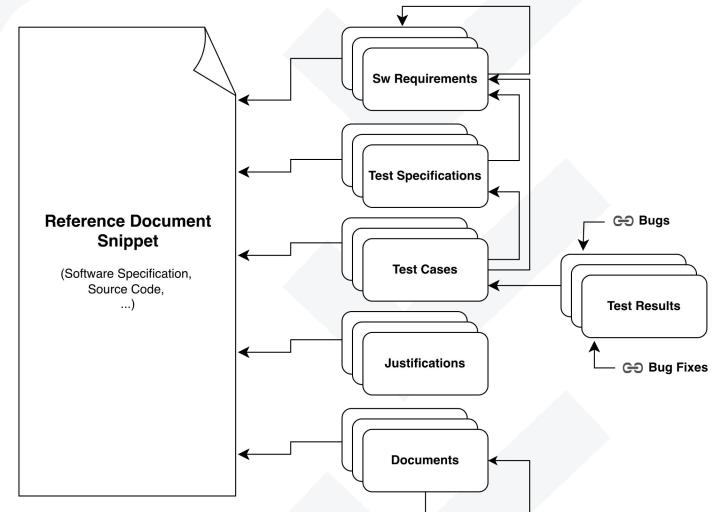
- Born at Red Hat to support RHIVOS Functional Safety ISO 26262 Compliance Certification
- BASIL name comes from ASIL B
- Presented to ELISA Project on June 2023 during the [Berlin Workshop](#)
- Open Sourced and hosted at [ELISA github](#)



BASIL The FuSa Spice



Define the traceability matrix by creating the work items



BASIL - Strengths

- Open Source
- Web user interface and HTTP REST API
- Extended traceability and SPDX SBOM generation
- Keeps history of all changes
- Granular user permissions management
- Clarifies gaps and promotes collaboration
- In-app and email notifications
- Embedded test infrastructure and support for external test infrastructures
- Import Software Requirements (SPDX Model 3 json, yaml, json, csv, xlsx)
- Import Test Cases from remote test repositories (via tmt)

BASIL supported test Infrastructures

Test Infrastructure	Trigger and Trace	Trace pre existing runs	Type	Available starting from version
tmt	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Embedded	>= 1.4
Gitlab CI	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	External	>= 1.5
GitHub Actions	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	External	>= 1.5
KernelCI	<input type="checkbox"/>	<input checked="" type="checkbox"/>	External	>= 1.5
Testing Farm	<input checked="" type="checkbox"/>	<input type="checkbox"/>	External	>= 1.5
LAVA (Linaro)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	External	>= 1.7

A dark gray background featuring a complex network graph composed of numerous small, semi-transparent white dots connected by thin white lines, creating a sense of depth and connectivity.

What's new?



Apache deployment

BASIL / deploy / apache / 

Name
 ..
 .env
 README.md
 build_basil_api.sh
 build_basil_frontend.sh
 common.sh
 init_postgresql.sh
 run.sh

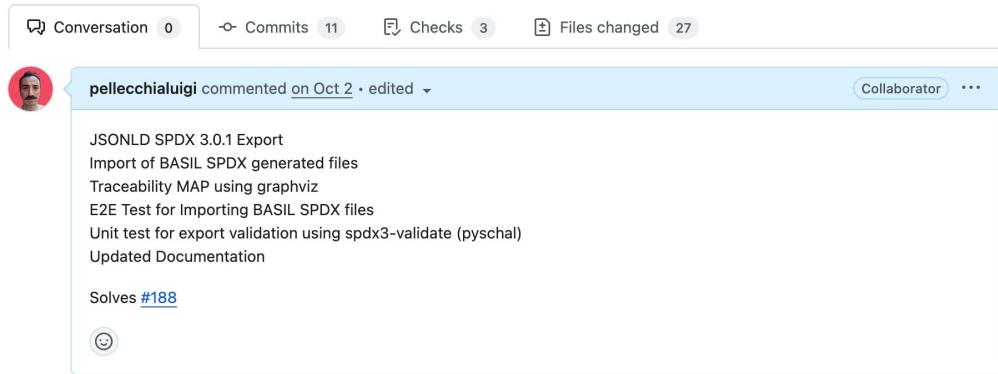
```
45 apache_deployment:
46   runs-on: ubuntu-latest
47   steps:
48     - name: Checkout
49       uses: actions/checkout@v4
50
51     - name: Define global variables
52       id: global_variables
53       run: |
54         echo "BRANCH_NAME=${{ github.head_ref || github.ref_name }}" >> "$GITHUB_OUTPUT"
55         sed -i 's/^BASIL_TESTING=.*$/BASIL_TESTING=1/' deploy/apache/.env
56         sed -i 's/^BASIL_API_PORT=.*$/BASIL_API_PORT=5005/' deploy/apache/.env
57         sed -i 's/^BASIL_APP_PORT=.*$/BASIL_APP_PORT=9056/' deploy/apache/.env
58         sed -i 's/^BASIL_ADMIN_PASSWORD=.*$/BASIL_ADMIN_PASSWORD=dummy_password/' deploy/apache/.env
59         sed -i 's/^BASIL_DB_PASSWORD=.*$/BASIL_DB_PASSWORD=dbSecret123/' deploy/apache/.env
60         TEMPDIR=$(mktemp -d)
61         sed -i "s|^BASIL_DB_PASSWORD=.*|BASIL_DB_PASSWORD=${TEMPDIR}|" deploy/apache/.env
62
63     - name: Make deploy/apache/run.sh executable
64       run: chmod +x deploy/apache/run.sh
65
66     - name: Run the deployment
67       run: |
68         cd deploy/apache && sudo ./run.sh
69
70
```

SPDX Export Refactoring

- SPDX export implemented at version 1.6.x was generating files that are not compliant with the SPDX 3.0.1 schema
- The first approach was based on SPDX3 python module
- A complete refactor of the feature was needed to comply with the SPDX 3.0.1 schema
- A CI test has been created to ensure BASIL exported files pass the schema validation

SPDX 3.0.1 Export #200

Merged pellecchialuigi merged 11 commits into main from issue-188 on Oct 2



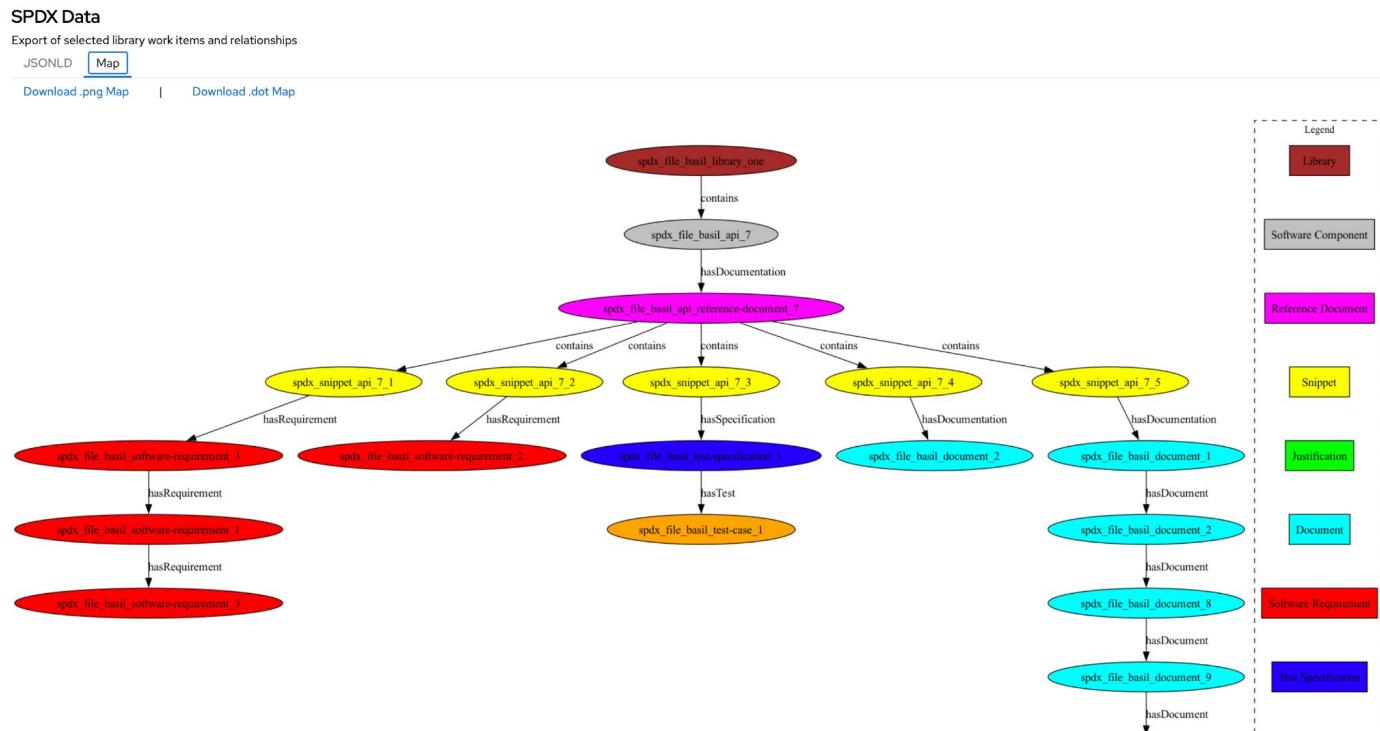
SPDX Export - CI Validation against spdx 3.0.1 schema

BASIL / api / test / test_spdx_api_validation.py

Code Blame 589 lines (497 loc) · 20.8 KB

```
427     def test_spdx_api_export_and_validation(client, user_authentication, comprehensive_spdx_test_data):
428         try:
429             # Run spdx3-validate on the generated file
430             print(f"Running spdx3-validate on {temp_file_path}")
431             result = subprocess.run(
432                 ["spdx3-validate", "--json", temp_file_path, "--spdx-version", "auto"],
433                 capture_output=True,
434                 text=True,
435                 timeout=60, # 60 second timeout
436             )
437
438             print(f"spdx3-validate exit code: {result.returncode}")
439             if result.stdout:
440                 print(f"STDOUT: {result.stdout}")
441             if result.stderr:
442                 print(f"STDERR: {result.stderr}")
443
444             # Check validation results
445             if result.returncode == 0:
446                 print("✓ SPDX validation passed successfully")
447                 assert True, "SPDX validation successful"
448             else:
449                 print("✗ SPDX validation failed")
450                 # Don't fail the test immediately - let's analyze what went wrong
451                 validation_errors = result.stderr or result.stdout
452                 print(f"Validation errors: {validation_errors}")
453
454             # You can add specific assertions here based on expected validation issues
455             # For now, we'll record the failure but continue
456             pytest.fail(f"SPDX validation failed with errors: {validation_errors}")
457
```

SPDX Export - Graph generation

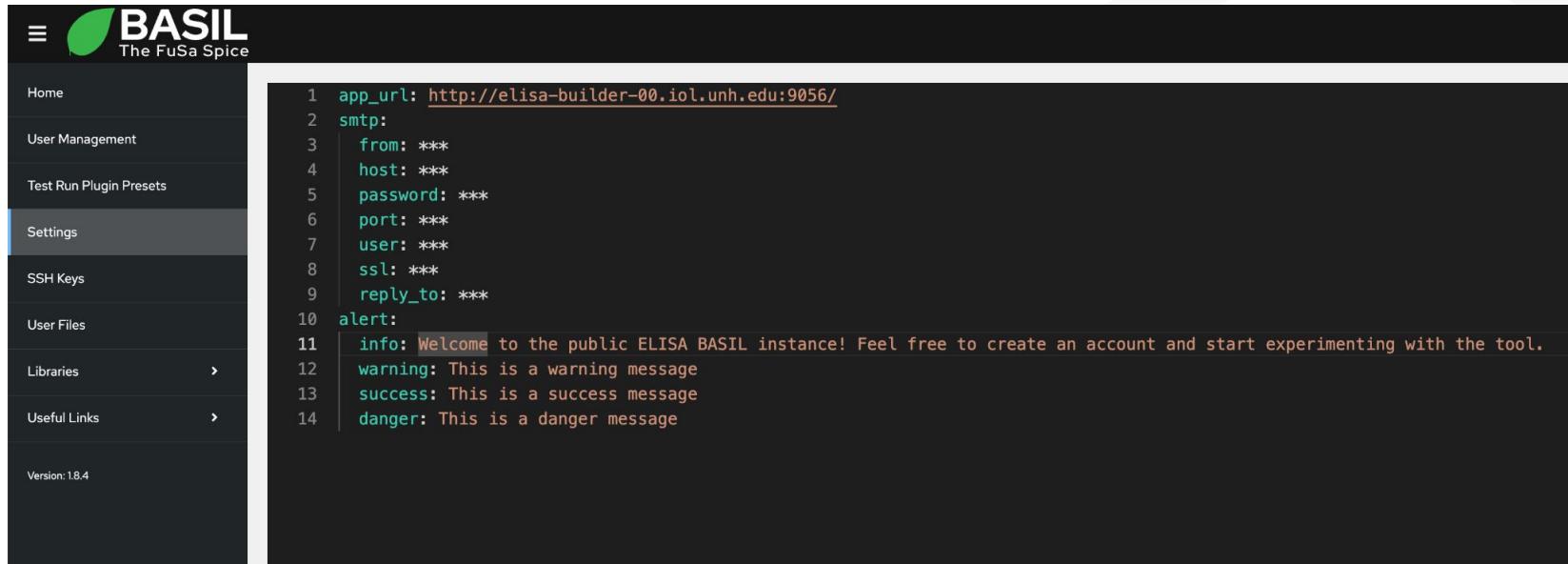


Configurable Alert Messages

The screenshot shows the ELISA BASIL web application interface. The left sidebar contains navigation links: Home, User Management, Test Run Plugin Presets, Settings, SSH Keys, User Files, Libraries (with a dropdown arrow), and Useful Links (with a dropdown arrow). The version information Version: 1.8.4 is also listed. The main content area has a header with a welcome message: "Welcome to the public ELISA BASIL instance! Feel free to create an account and start experimenting with the tool." Below this is a navigation bar with tabs: experimentation (which is active), syscalls, and test. The main section is titled "API Listing for experimentation" and shows a table with one row. The table has columns: ID, API, Version, and Owner. The single entry is: ID 1, API nanosleep, Version 1, and Owner admin. A small badge indicates "Covered 10.4%".

ID	API	Version	Owner
1	nanosleep	1	admin

Configurable Alert Messages



The screenshot shows the ELISA BASIL web application interface. The left sidebar contains navigation links: Home, User Management, Test Run Plugin Presets, Settings (which is selected), SSH Keys, User Files, Libraries, and Useful Links. Below the sidebar, it says "Version: 1.8.4". The main content area displays a configuration snippet for alert messages:

```
1 app_url: http://elisa-builder-00.iol.unh.edu:9056/
2 smtp:
3   from: ***
4   host: ***
5   password: ***
6   port: ***
7   user: ***
8   ssl: ***
9   reply_to: ***
10 alert:
11   info: Welcome to the public ELISA BASIL instance! Feel free to create an account and start experimenting with the tool.
12   warning: This is a warning message
13   success: This is a success message
14   danger: This is a danger message
```

New Release available check

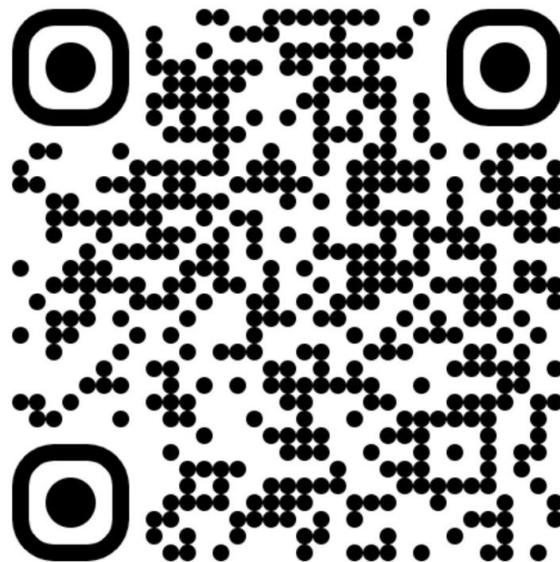
The screenshot shows the public ELISA BASIL instance homepage. The header features the BASIL logo with the tagline "The FuSa Spice". A navigation bar on the left includes links for Home, Login, Sign In, Libraries, and Useful Links. A message at the top right says, "Welcome to the public ELISA BASIL instance! Feel free to create an account and start experimentation." Below this, there are tabs for experimentation, syscalls, and test, with "experimentation" selected. The main content area displays an "API Listing for experimentation" with a status badge indicating "Covered 10.4%". A table lists one API entry:

ID	API	Version	Owner
1	nanosleep	1	admin

A message at the bottom left of the sidebar indicates "Version: 1.8.4" and "New version available".

Demo on ELISA instance

Create an account and start experimenting with it



<http://elisa-builder-00.iol.unh.edu:9056>

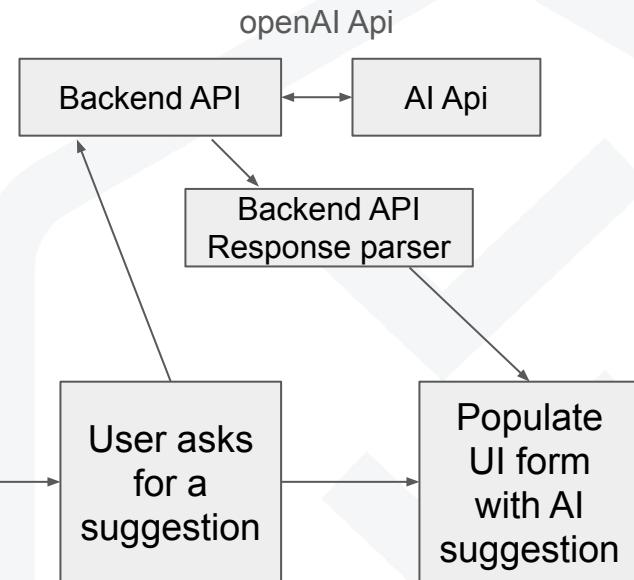
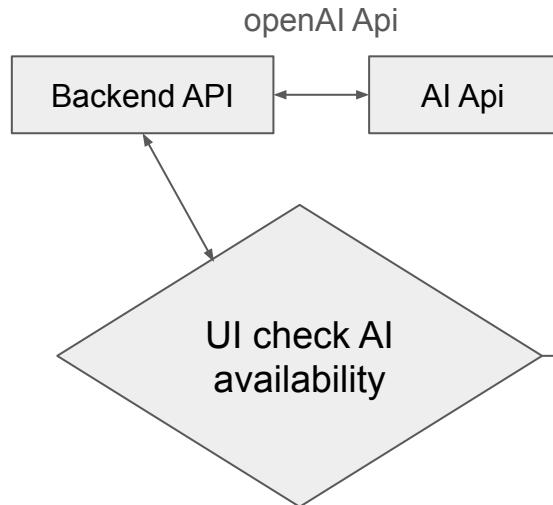
AI Support

- Admins can configure an external AI tool and communicate with it via openAI
- UI will enable components in case of successful connection with the AI endpoint

Can be used to:

- Design work item from BASIL tool
 - Default prompts are asking to generate YAML like format that can be parsed and used to populate the UI in case of success
- Implement a Test Case based on Test Specification
 - Default prompts are asking to generate a file that will be stored in the user files section

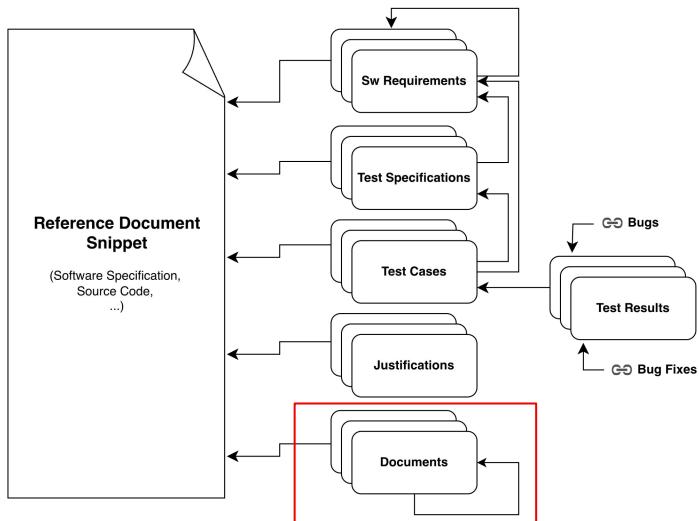
AI Support



Link to presentation at Open Source Summit Europe 2025

<https://www.youtube.com/watch?v=17ftlTXjGI&t=2s&pp=ygUkYmFzaWwgd2hhCdzlG9wZW4gc291cmNIIHN1bW1pdCAyMDI1>

Hierarchical Documents Mapping



Document 1 ::

ver. 2.1 new 3.1% Completion

Top Level Document

This is a document mapped to the API reference document

Type: file

Url: /Users/lpelucc/Documents/tmp/github/BASIL/api/user-files/1/BASIL.git_20250618233433.json

SPDX Relation: AFFECTS

Document 2 ::

ver. 2.1 new 6.3% Completion

This a Document mapped to a document

Description of a document mapped to another document

Type: file

Url: <http://this-is-an-url>

SPDX Relation: PREREQUISITE

Document 8 ::

ver. 3.1 in progress 12.5% Completion

This is another document

Description of another document used as an example of multiple documents mapping

Type: file

Url: <http://this-is-another-ur>

Traceability as Code

- Work items handled in git repositories
- Scan multiple external git repositories to extract data
- Shareable configuration file generates same results
- Automatically takes into account new work items
- Recreate the traceability of a target git commit
- Not tied to a single work item format
- Can be used in CI as the feature is provided by a command line tool
- Easy to extend with custom rules
- Can be used in CI automating the generation of an SBOM

Traceability as Code - Workflow

- Software components definition
- Repository configuration and candidate files extraction
- Reference document sections extraction
- Candidate work items data extraction per field
- Candidate work items data filtering
- Candidate work items data refactoring
- Traceability generation

Traceability as Code - Repository files filtering

Define rules to identify the files containing the work items data

- Is targeting a git commit/branch
- Reusable via YAML anchor
- Filter over
 - Files
 - Folders
 - File content

```
repository: &repository_config
url: "https://github.com/elisa-tech/BASIL.git"
branch: "main"
filename_pattern: "*.*"
folder_pattern: "*examples*"
hidden: False
file_contains: ["read_mem"]
file_not_contains: []
```

Traceability as Code - Work item field data extraction

Define how each field should be populated identifying **start** and **end** rules that will be applied over the candidate files.

The match condition can generate list of sections.

Supports relational search with **closest** match in a target **direction**

Allows **split** of a section based on a **delimiter** to generate multiple matches

```
description:  
  start:  
    line_contains: " read_mem()  
  closest:  
    line_contains: "Function's expectations:"  
      direction: "up"  
  end:  
    line_contains: "* Context"  
  split:  
    by: "\n*\n"
```

Traceability as Code - Work items candidates extraction

Define rules to extract work items data from identified candidate files:

Rules for each work item fields

- Magic variables
- Relational search
- Support constant values
- List generation from each result

```
software_requirements:  
  rules:  
    - name: "sr1"  
      repository: *repository_config  
      skip_top_items: 1  
      title:  
        value: |  
          "Function expectation __software_requirement_index__"  
      description:  
        start:  
          line_contains: " read_mem("
          closest:  
            line_contains: "Function's expectations:"  
            direction: "up"  
        end:  
          line_contains: "* Context"  
        split:  
          by: "\n*\n"
```

Traceability as Code - Work items candidates filtering

Define rules to filter over the identified candidate work items:

- contains
- not contains
- regex

```
software_requirements:  
  rules:  
    - name: "sr1"  
      repository: *repository_config  
      skip_top_items: 1  
      title:  
        value: |  
          "Function expectation __software_requirement_index__"  
      filter:  
        contains: ["keep", "1"]  
        Case_sensitive: false
```

Traceability as Code - Work items candidates refactoring

Define rules to refactor and format work items data:

- replace
- regex substitution
- uppercase
- lowercase
- left trim characters
- right trim characters
- global trim
- prefix
- suffix

```
software_requirements:  
  rules:  
    - name: "sr1"  
      repository: *repository_config  
      skip_top_items: 1  
      title:  
        value: |  
          "Function expectation __software_requirement_index__"  
          rstrip: "!?)]},;:"  
      transform:  
        - how: replace  
          what: what_to_find  
          with: replace_with  
        - how: prefix  
          value: "Requirement: "
```

Traceability as Code - Work items generation

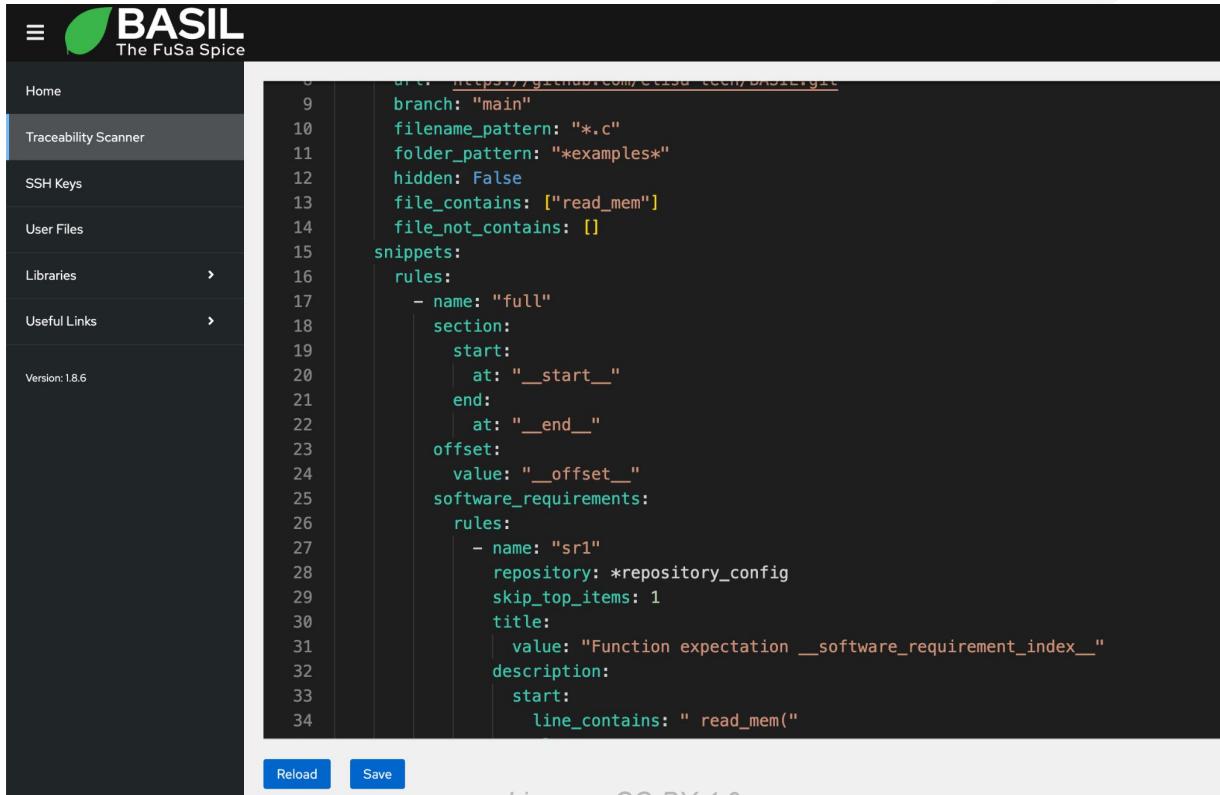
Check if the work items are already available in the BASIL database based on their content and create new ones if not.

Traceability as Code - Example

```
/**  
 * 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.  
 *  
 * Function's expectations:  
 *  
 * 1. This function shall check if the value pointed by ppos exceeds the  
 *    maximum addressable physical address;  
 *  
 * 2. This function shall check if the physical address range to be read  
 *    is valid (i.e. it falls within a memory block and if it can be mapped  
 *    to the kernel address space);  
 *  
.....  
*/  
  
static ssize_t read_mem(struct file *file, char __user *buf,  
                      size_t count, loff_t *ppos)
```

```
software_requirements:  
  
rules:  
- name: "linux kernel requirements"  
  repository: *linux_repo_config  
  skip_top_items: 1  
  title:  
    value: "Function expectation  
__software_requirement_index__"  
  description:  
    start:  
      line_contains: "__api__()  
    closest:  
      line_contains: "Function's expectations:"  
      direction: "up"  
    end:  
      line_contains: "* Context"  
  split:  
    by: "\n*\n"  
transform:  
- how: "regex_sub"  
  what: " +"  
  with: "  
- how: "suffix"  
  value: ".."  
  rstrip: ",.;!:?"
```

Traceability as Code - UI



The screenshot shows the BASIL Traceability Scanner interface. The left sidebar contains navigation links: Home, Traceability Scanner (which is selected), SSH Keys, User Files, Libraries, and Useful Links. Below these is the version information: Version: 1.8.6. The main area is a code editor displaying a YAML configuration file. The file includes a URL, a branch, filename and folder patterns, and various rules and snippets. At the bottom of the code editor are two buttons: Reload and Save.

```
git: "https://github.com/CERN-CCS/BASIL.git"
branch: "main"
filename_pattern: "*.c"
folder_pattern: "*examples*"
hidden: False
file_contains: ["read_mem"]
file_not_contains: []
snippets:
  rules:
    - name: "full"
      section:
        start:
          at: "__start__"
        end:
          at: "__end__"
        offset:
          value: "__offset__"
      software_requirements:
        rules:
          - name: "sr1"
            repository: *repository_config
            skip_top_items: 1
            title:
              value: "Function expectation __software_requirement_index__"
            description:
              start:
                line_contains: " read_mem("
```

Reload Save

Demo



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.



ELISA
Enabling Linux in
Safety Applications

WORKSHOP

