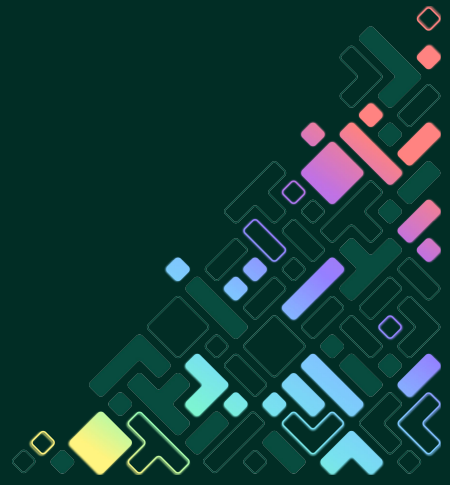
SAFETY-CRITICAL
SOFTWARE
SUMMIT

BASIL The FuSa Spice

Software Quality Management Tool



lpellecc@redhat.com
gpaoloni@redhat.com

THE LINUX FOUNDATION

#EmbeddedOSSummit

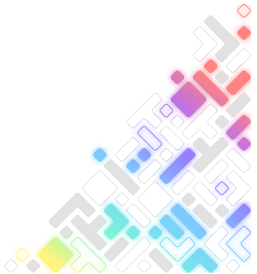
Who are we?

Luigi Pellecchia
Principal
Software Quality Engineer
Quality Engineering
In-vehicle OS

Gabriele Paoloni
Senior Principal
Software Engineer
Functional Safety
In-vehicle OS

Agenda

- Challenges in managing Quality in Linux
- Tool Capabilities
- Examples
- Code
- Roadmap
- How to Contribute

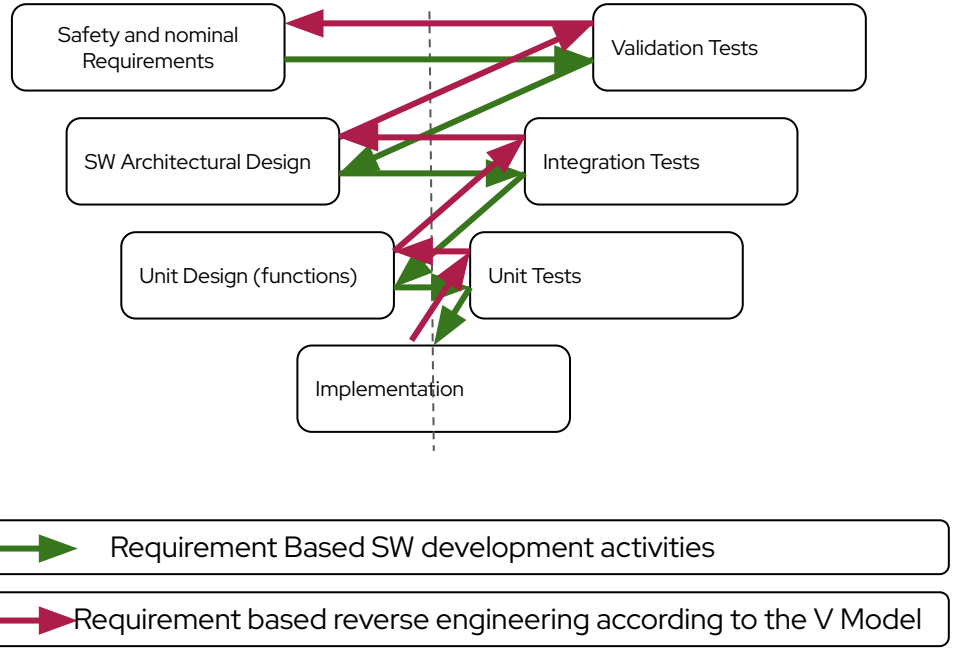


Challenges in managing quality in Linux

Feasibility in applying the V-Model to Linux

Most of QMS standard demands a SW development process that is compliant to the V-Model and is requirements centric, whereas the open source development model is code centric.

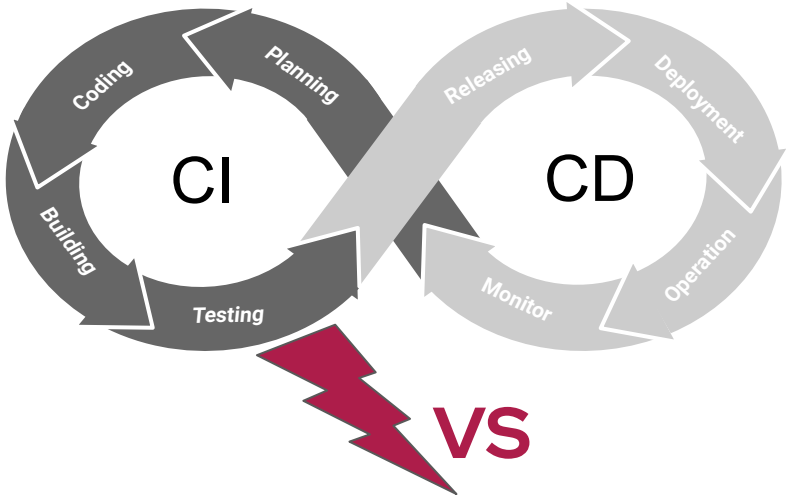
Reverse engineering of the code to accommodate the V-Model is very costly and maintaining the resulting set of evidences is challenging

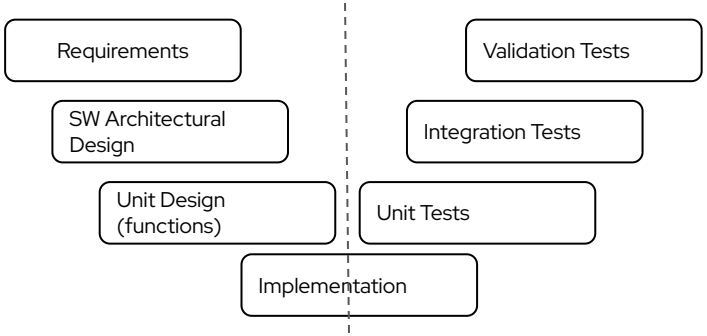


Recognized QMS not accommodating a CI/CD development process

- As of today there are no recognized nor standardized QMS that define quality management practices for SW CI/CD.

- Most of tools that are used to maintain a set of QM evidences are designed to fit a requirement based development process, not a CI/CD evolving code

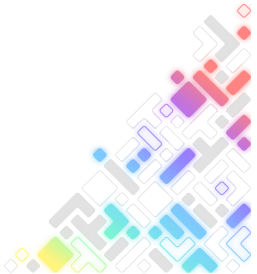




What is BASIL?

An Open Source Software Quality Management Tool.

It makes it easy to manage requirements, test specifications, test cases defining your traceability matrix against your software specifications.

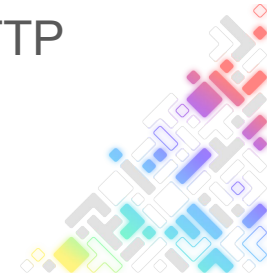


Why BASIL?

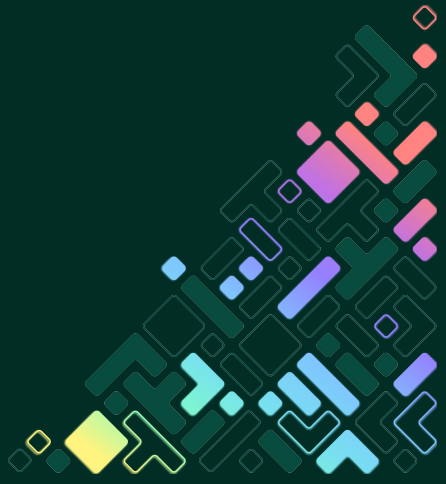
- BASIL fully integrates with the Linux configuration management tools (i.e. git and github). Therefore al the code changes BASIL can detect which QMS evidences are impacted

- BASIL is designed to fit a CI/CD development process by focusing on code and specification first.
  Requirements and test work items are linked to code and/or specification files; as the code and/or specifications evolve, BASIL can be used to evaluate the impact on requirements and testing.

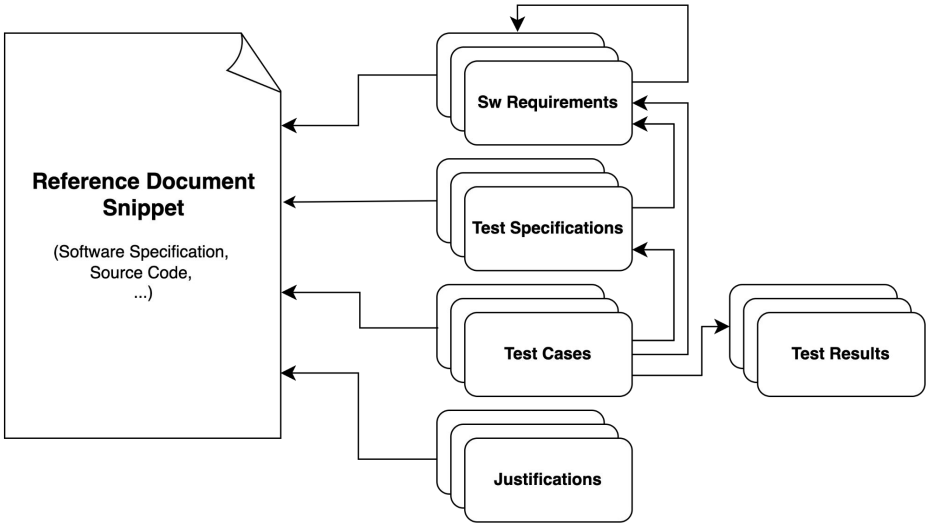
- BASIL facilitates open collaboration through SPDX support and HTTP API support



BASIL Tool Capabilities



Traceability first



Work items can be created only once a reference document has been specified using

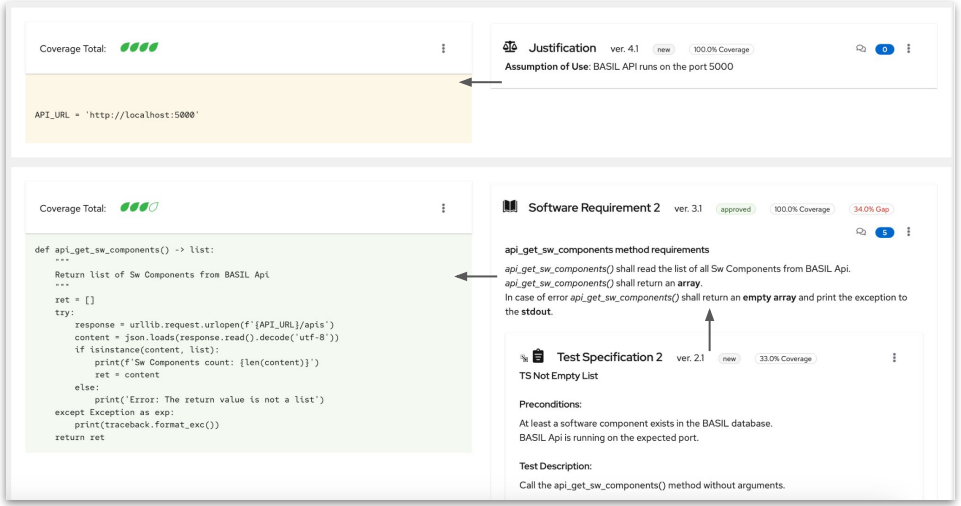
Direct Mapping

Work Items are created against a snippet of the Reference Document

Indirect Mapping

Work Items are created as child of another work item

Traceability first



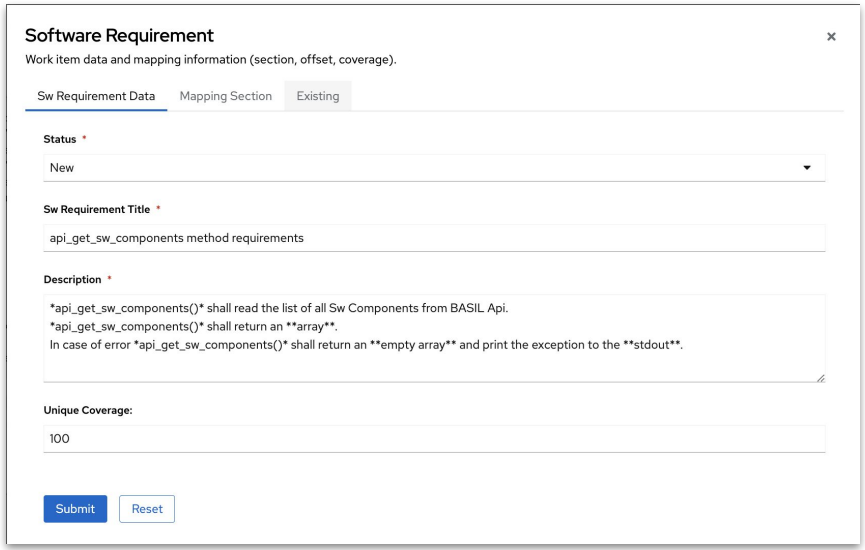
In BASIL work items are created defining a relationship and displayed following the defined relationships.

BASIL proposes different views focusing on Direct Mapping

- Sw Requirements
- Test Specifications
- Test Cases
- Justifications
- Raw Specification

Default view can be selected for each sw component

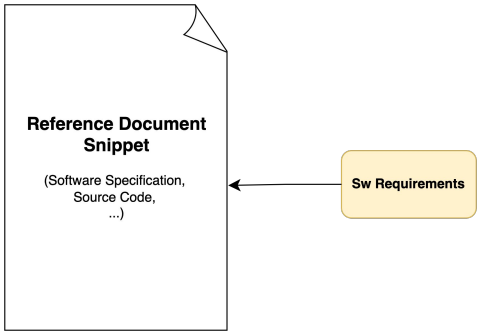
Work Items management



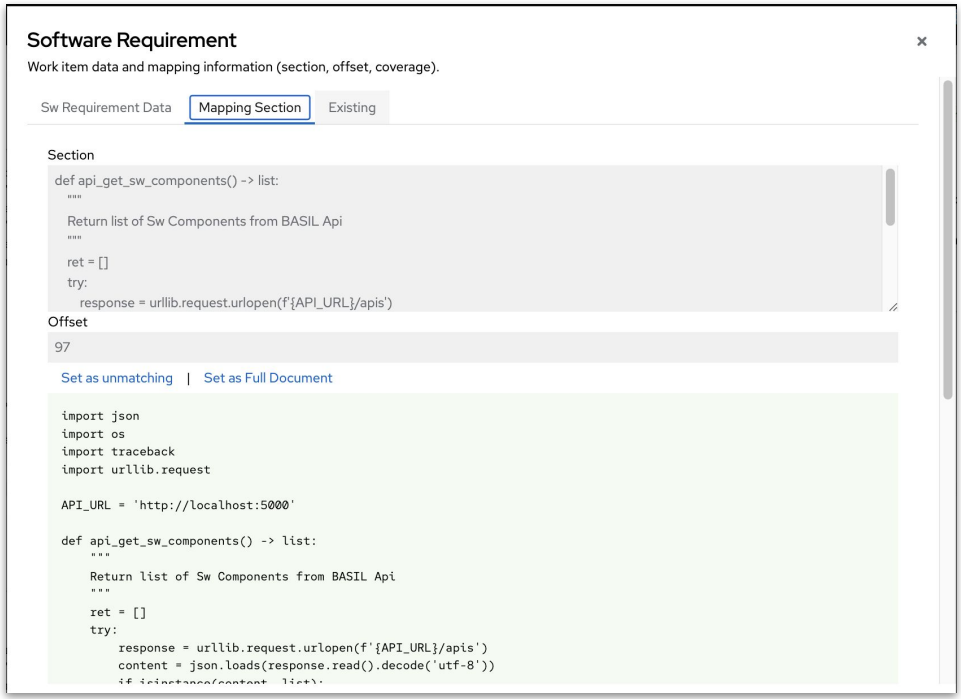
Example: Sw Requirement

Work Item data

- Status
- Title
- Description



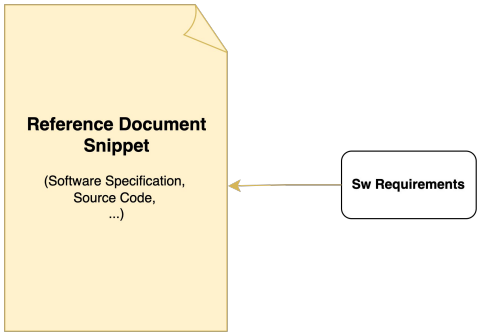
Work Items management



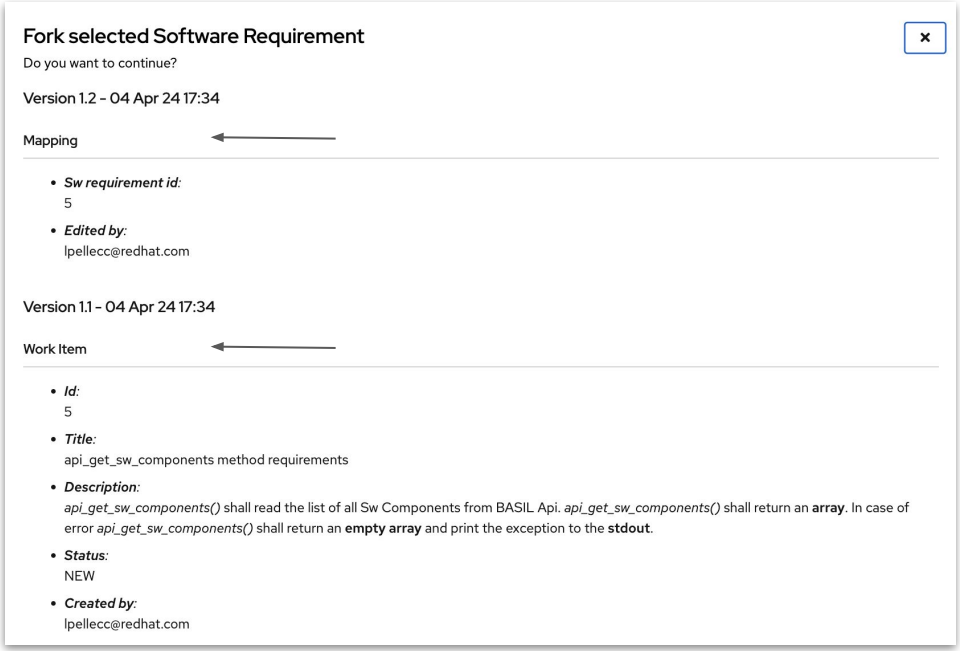
Example Sw Requirement

Relationship data

- Section
- Offset
- Coverage



History



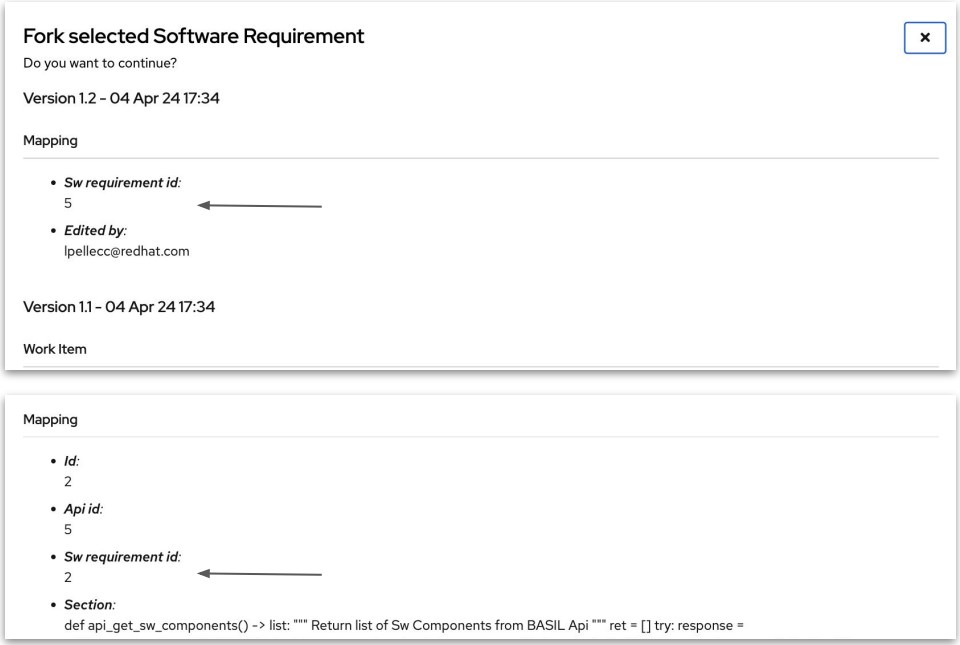
Version number (e.g. 4.2) is a combination of:

- Work Item version
- Mapping version

For each version BASIL will reports:

- User
- Date
- Fields affected

Fork



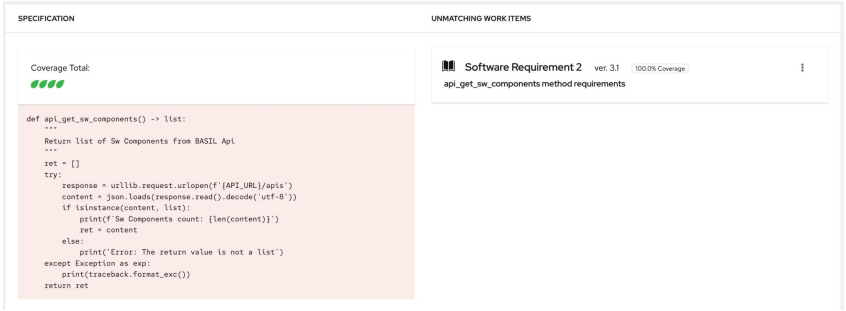
Why?

If we have a work items used across multiple software components and we want to let it diverge just for one.

How?

Creating a new requirement copying work item and mapping information and keeping in the history the ID of the original one.

Reference Document changes

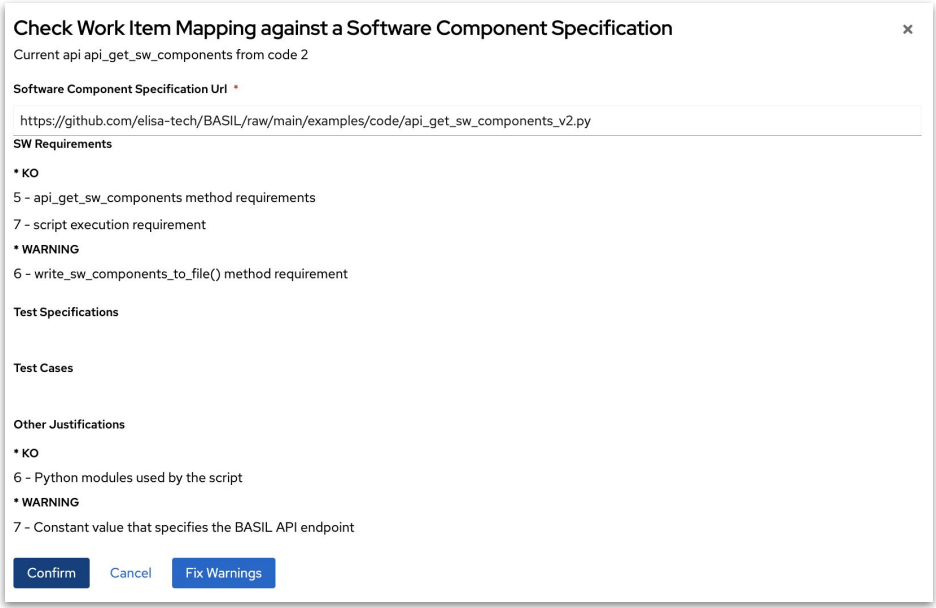


What happen?

There are 2 possibilities:

- the mapping no longer exists at all
- it exists but is shifted to a different offset

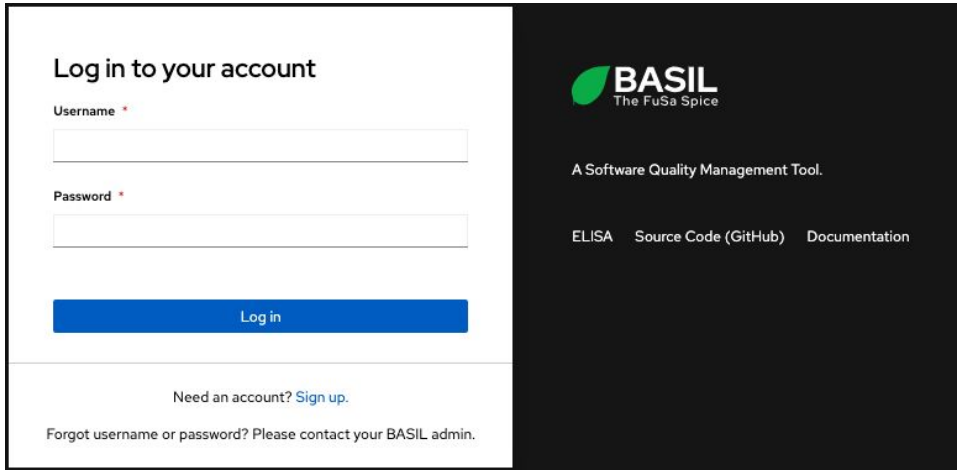
Reference Document changes



With BASIL we can check the effect of changes to the Reference Document before creating a new Sw Component version.

Once created the new version of the Sw Component we can automatically fix the warnings (mapping shifted in the document).

User Management

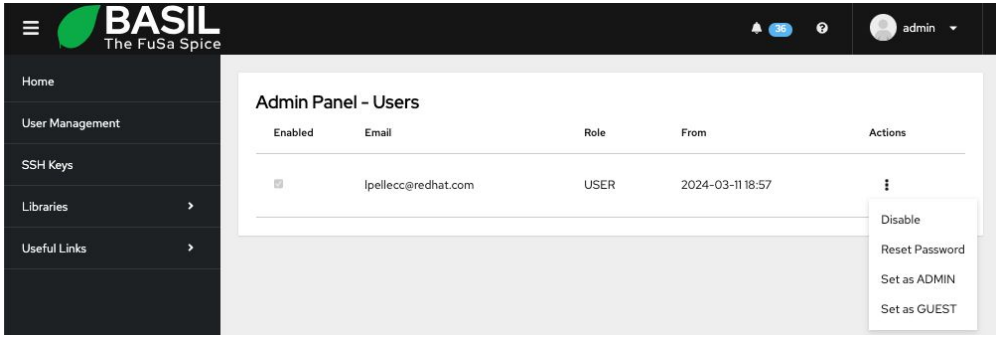


To have a public instance of BASIL available on the web, user management is a mandatory feature.

Goal:

- Define user roles
- Define permissions at Sw Component level

User Management - Roles



Roles:

- ADMIN
- USER
- GUEST (Default for new users)

Admin can access User Management to:

- Disable User account
- Reset User Password
- Change User Role

User Management - Admin

Build the Containers

You can build the API project using the Dockerfile-api.

Building this container you will also initialize the database.

If you need to host BASIL on a server you should specify following build argument:

- API_PORT (default is 5000)

BASIL will be configured with an admin user named **admin**. You can specify the **admin** user password with the following build argument:

- ADMIN_PASSWORD (default is **admin**)

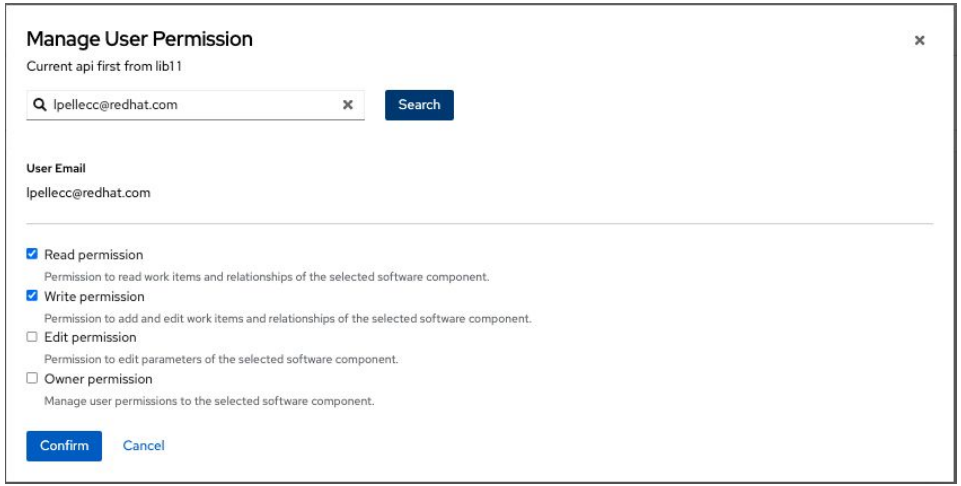
Here an example of docker build command:

```
docker build \
   --build-arg ADMIN_PASSWORD=your-desired-password \
   --build-arg API_PORT=1234 \
   -f Dockerfile-api \
   -t basil-api-image .
```

Creating a new instance of BASIL an admin user with ADMIN role will be created.

Its default password can be changed using a container build argument as described in the documentation.

User Management - Permissions

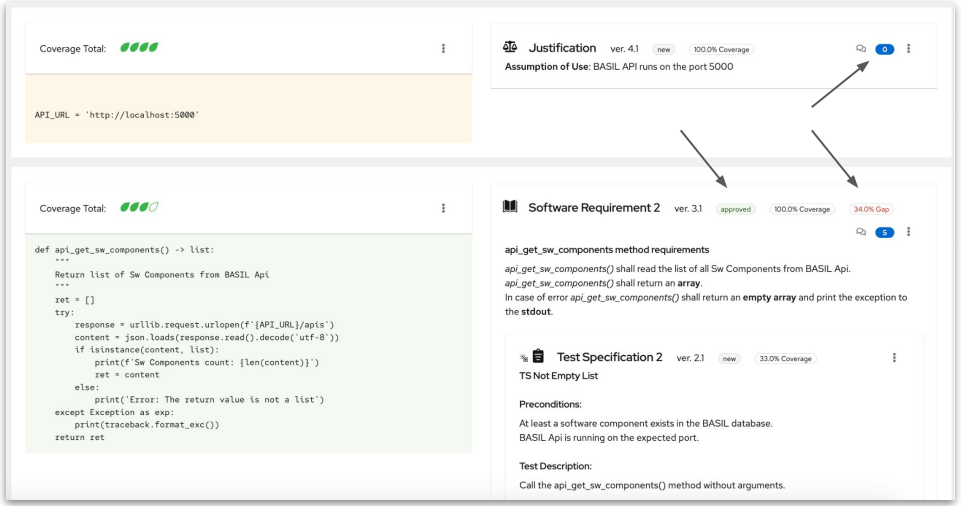


Sw Component owners can change user permissions.

The available permissions are:

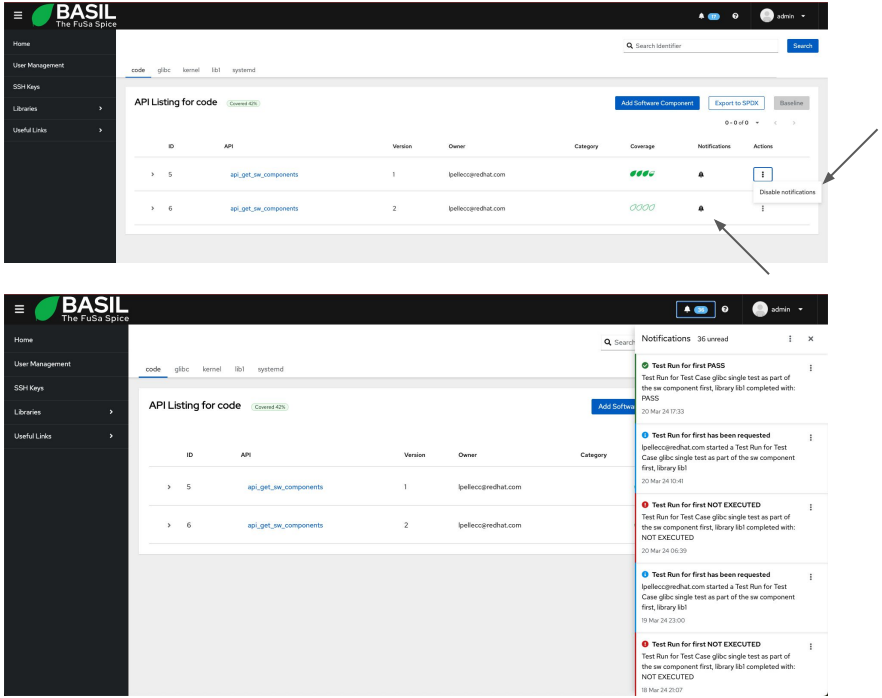
- Read
- Write
- Edit
- Owner

Collaboration



- With a web application changes are available to all the users at the same time
- Comments to direct mapped work items
- Approval / Rejection
- Highlight gap to simplify contributors onboarding
- Creating public instances we can let anyone joining and contributing

Notifications

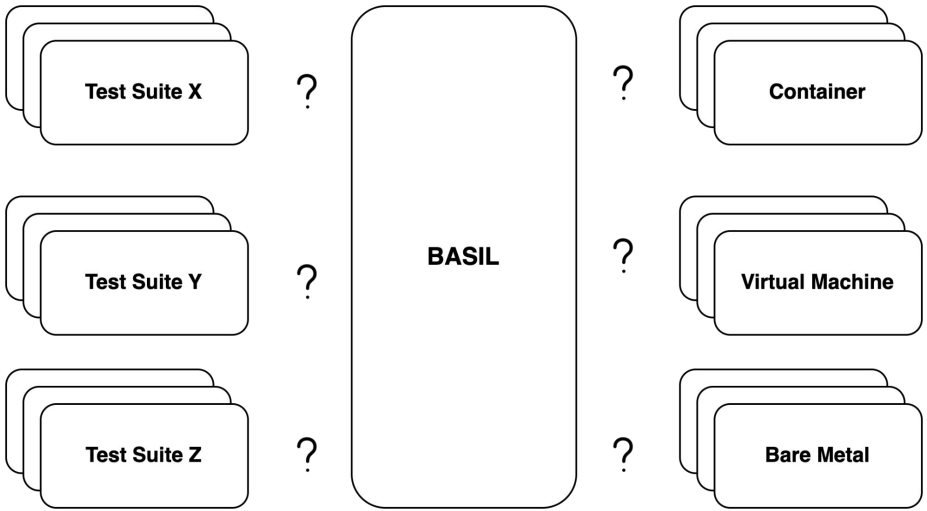


The user can select for which software components he wishes to receive notifications

Example of Notifications:

- New User
- Test Run requested/completed (result)
- Work Item changes
- Mapping changes

Test Case execution and results



How it is possible to let BASIL able to run any kind of Test Suite on a desired Test Environment?

We need an abstraction layer!

tmt a Test Management Tool

```
summary: Example fmf file

description: This is an example of an fmf metadata file

contact: Luigi Pellecchia <lpellecc@redhat.com>

component: basil

recommend: python3

test: python3 test_entrypoint.py

framework: shell

duration: 5m
```

tmt introduce an abstraction layer that allow BASIL to interact with any test suite.

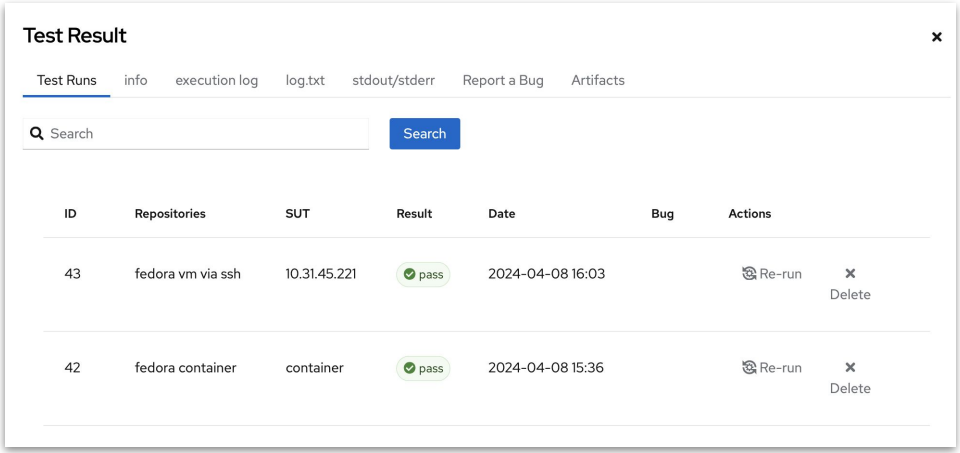
tmt uses the flexible metadata format (fmf) to describe test cases, test plans, and user stories.

github.com/teemtee

Some project numbers:

- 2000+ commits
- 100+ forks
- 52 releases
- 68 contributors

tmt a Test Management Tool - provision



tmt supports multiple provisioning mechanisms as described at https://tmt.readthedocs.io/en/stable/spec/plans.html#provision

BASIL is leveraging:

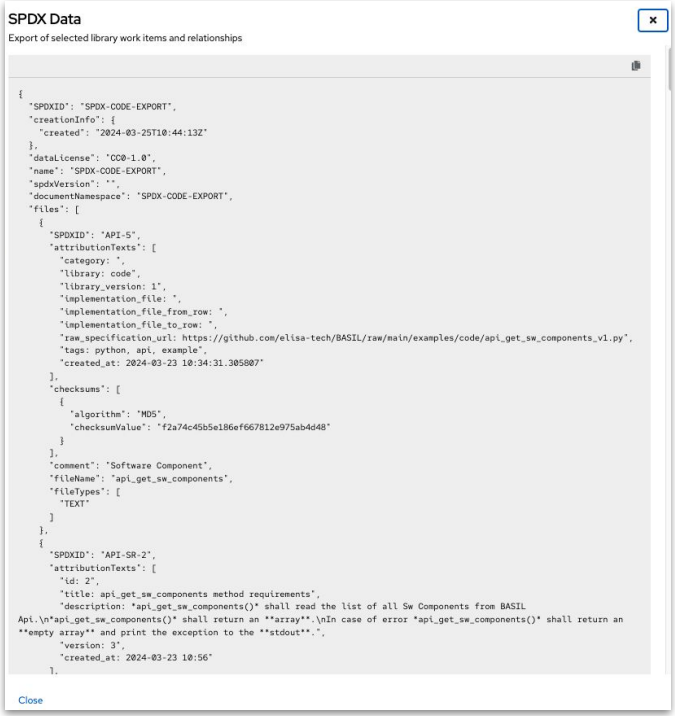
- **container** (default Fedora)
- **connect** (SSH)

tmt a Test Management Tool - artifacts

BASIL provides access to the artifacts available at the **$TMT_PLAN_DATA** folder.

```
1   summary: glibc single test
2   description: glibc single test
3   require:
4     - bison
5     - gcc
6     - git
7     - glibc
8     - make
9   test: |
10    CURRENT_TEST=${CURRENT_TEST:=libio/tst-fopenloc}
11    GLIBC_DOWNLOAD_PATH=$HOME/src
12    GLIBC_BUILD_PATH=$HOME/build/glibc
13    mkdir $GLIBC_DOWNLOAD_PATH || true
14    cd $GLIBC_DOWNLOAD_PATH
15    GIT_SSL_NO_VERIFY=true git clone --depth 1 https://sourceware.org/git/glibc.git
16    mkdir -p $GLIBC_BUILD_PATH
17    cd $GLIBC_BUILD_PATH
18    $GLIBC_DOWNLOAD_PATH/glibc/configure --prefix=/usr
19    make
20    make test t=$CURRENT_TEST
21    cp $GLIBC_BUILD_PATH/$CURRENT_TEST.out $TMT_PLAN_DATA/test.out
22    cp $GLIBC_BUILD_PATH/$CURRENT_TEST.test-result $TMT_PLAN_DATA/test.test-result
23  framework: shell
24  duration: 12h
```

SPDX



SPDX Export supported at Library level.

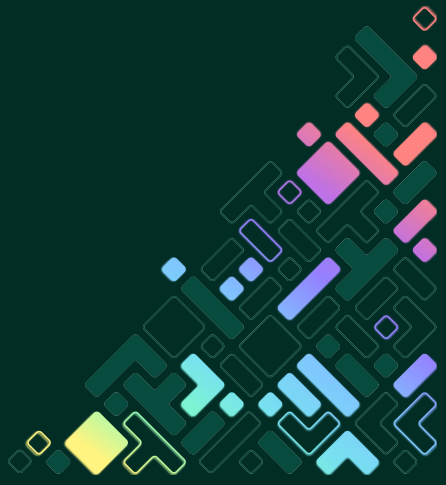
HTTP API


```
[
  {
    "id": 5,
    "api": "api_get_sw_components",
    "library": "code",
    "library_version": "1",
    "raw_specification_url": "https://github.com/elisa-tech/BASIL/raw/main/examples/code/api_get_sw_components_v1.py",
    "category": "",
    "checksum": "",
    "implementation_file": "",
    "implementation_file_from_row": "",
    "implementation_file_to_row": "",
    "created_by": "lpellecc@redhat.com",
    "edited_by": "lpellecc@redhat.com",
    "tags": "python, api, example",
    "version": "1",
    "srs_coverage": 100,
    "tss_coverage": 0,
    "tcs_coverage": 0,
    "covered": 100,
    "permissions": "r",
    "notifications": 0
  },
  {
    "id": 6,
    "api": "api_get_sw_components",
    "library": "code",
    "library_version": "2",
    "raw_specification_url": "https://github.com/elisa-tech/BASIL/raw/main/examples/code/api_get_sw_components_v2.py",
    "category": "",
    "checksum": "",
    "implementation_file": "",
    "implementation_file_from_row": "",
    "implementation_file_to_row": "",
    "created_by": "lpellecc@redhat.com",
    "edited_by": "lpellecc@redhat.com",
    "tags": "python, api, example",
    "version": "1",
    "srs_coverage": 100,
    "tss_coverage": 0,
    "tcs_coverage": 0,
    "covered": 0,
    "permissions": "r",
    "notifications": 0
  }
]
```

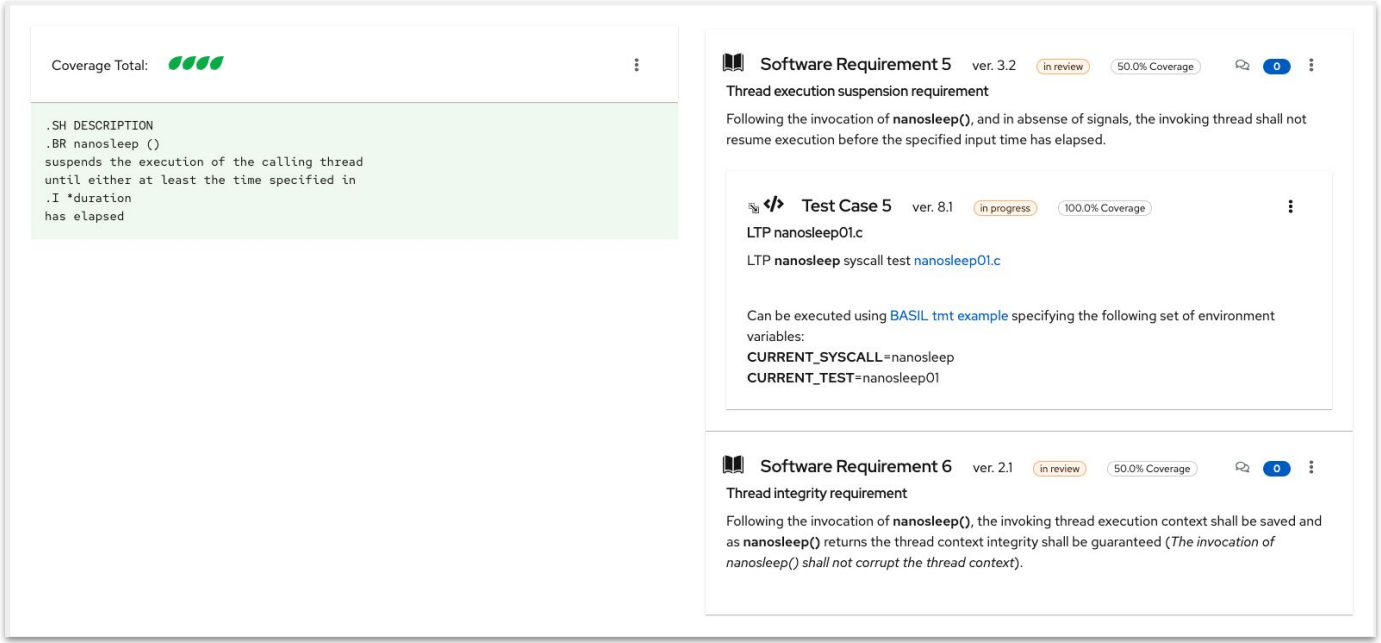

Any action in BASIL is performed via HTTP Api

It can be used to simplify the integration in CI/CD and other toolchain or to retrieve information to fill reports

Examples



Linux kernel syscalls analysis

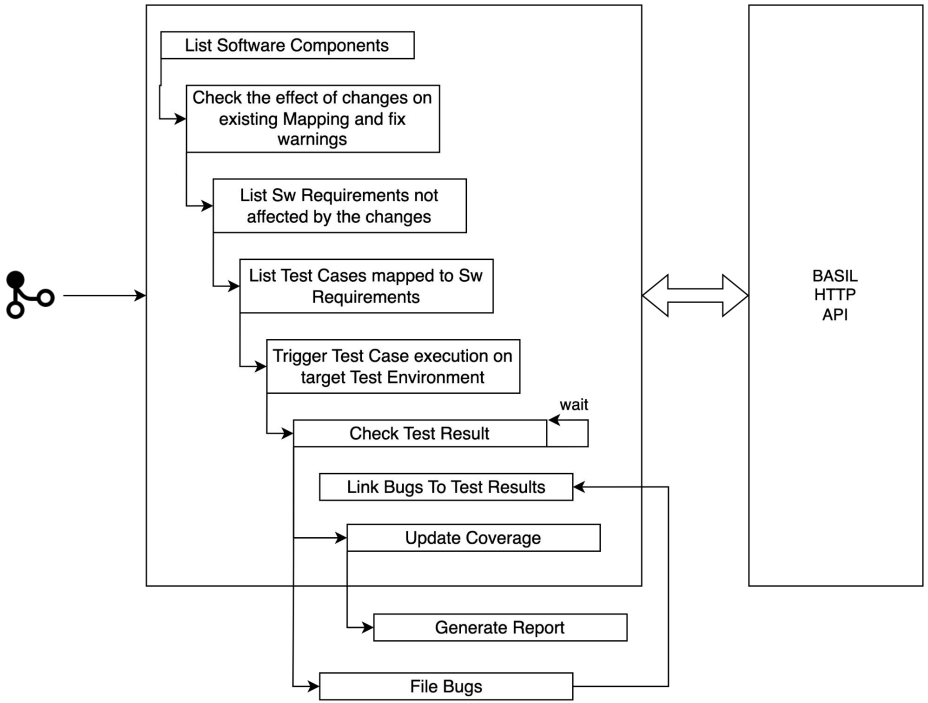


Man Pages are used as a software specification by developers everyday.

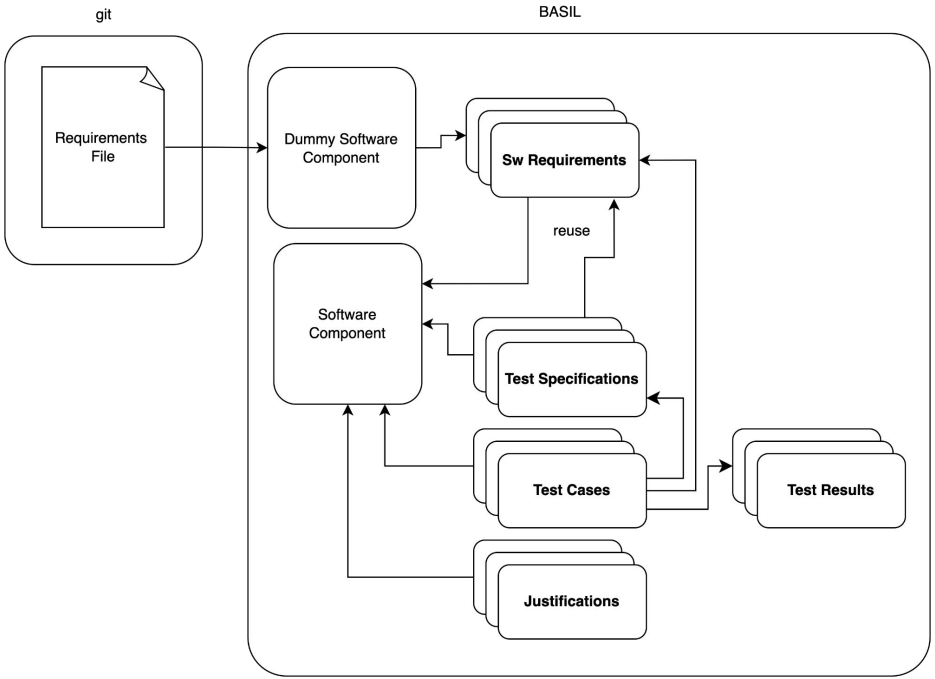
BASIL can be used to:

- evaluate current Man Pages test coverage
- to keep LTP project aligned to Man Pages and Source Code

Change request impact analysis example



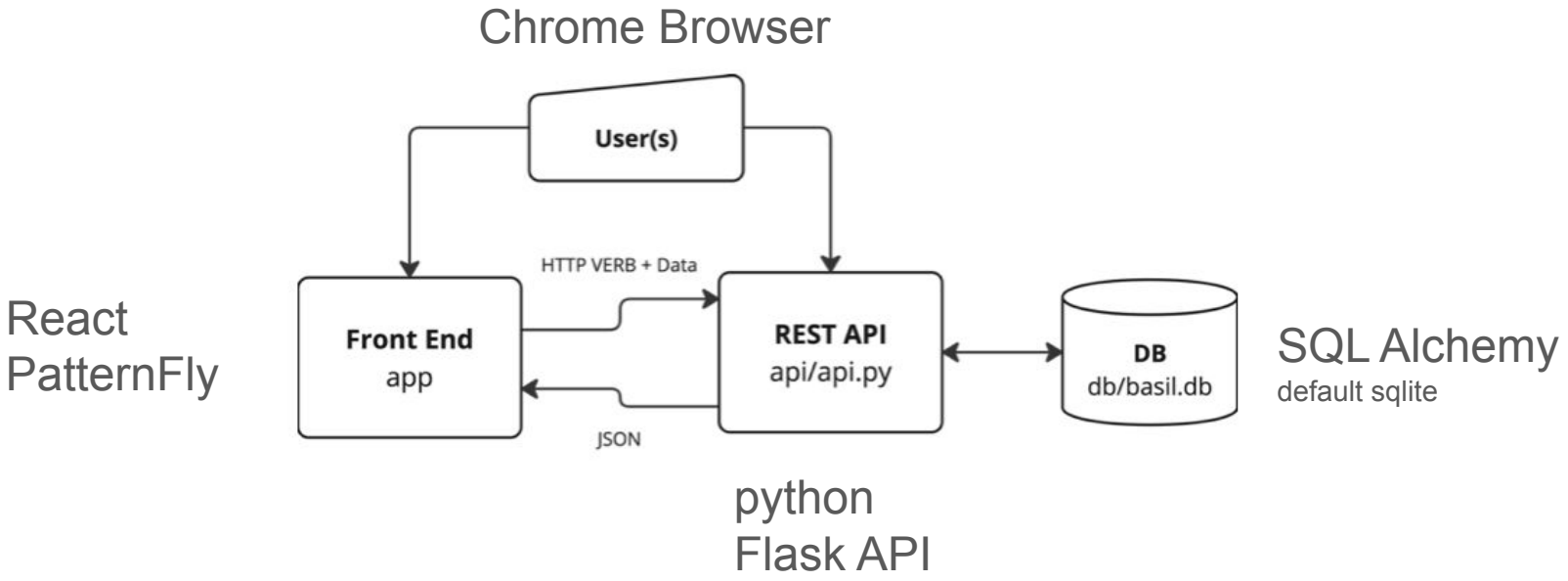
Sw Requirements from files - Zephyr



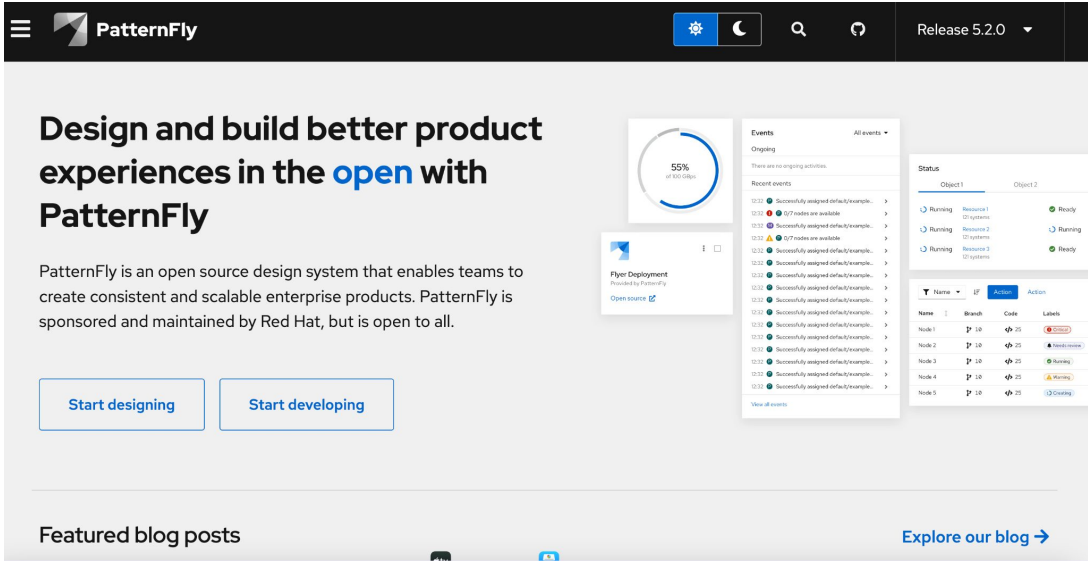
- A Dummy Sw Component can be used to generate BASIL Sw Requirements
- A changes to the Requirements File in git is reflected in BASIL affecting the mapping of Software Requirements in the Sw Components
- Sw Requirements derived from the Requirements File can be reused in BASIL in a real Software Component and leverage BASIL capabilities to create additional traceability

Code

github.com/elisa-tech/BASIL



PatternFly



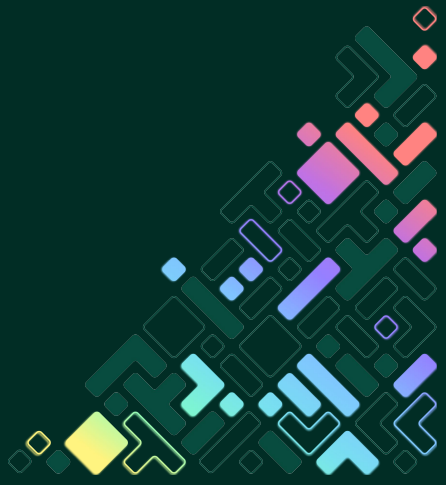
Designed with great attention to Accessibility it provides:

- **Patterns**
- **Layouts**
- **Components**
- **Set of design kits and libraries in Figma**

Code available at
https://github.com/patternfly

PatternFly-react-seed app
can get you started quickly with new projects

Roadmap

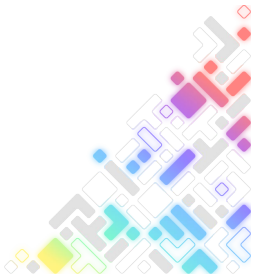


Tool Development

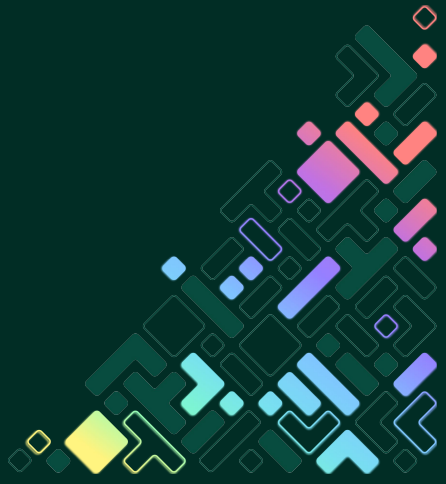
- Sw Components as work item
- Support multiple reference documents for each Sw Component
- Baselines
- Pagination
- Extend API and APP Test Suites

Communities

- ELISA BASIL Instance



How to Contribute



Documentation

Documentation is available at
https://basil-the-fusa-spice.readthedocs.io

github issues
https://github.com/elisa-tech/BASIL/issues

SAFETY-CRITICAL SOFTWARE SUMMIT