



SAFETY-CRITICAL
SOFTWARE
SUMMIT

BASIL The FuSa Spice



Software Quality Management Tool



#EmbeddedOSSummit

lpellecc@redhat.com
gpaoloni@redhat.com



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

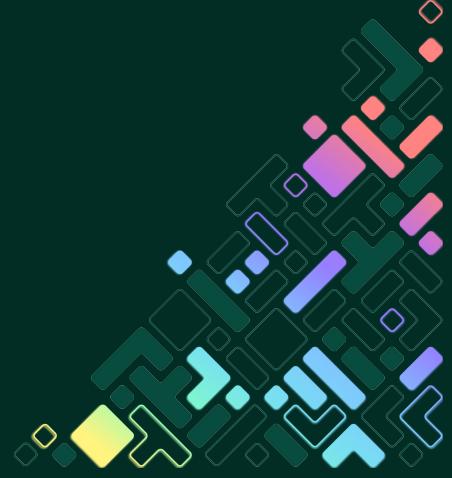


Agenda

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



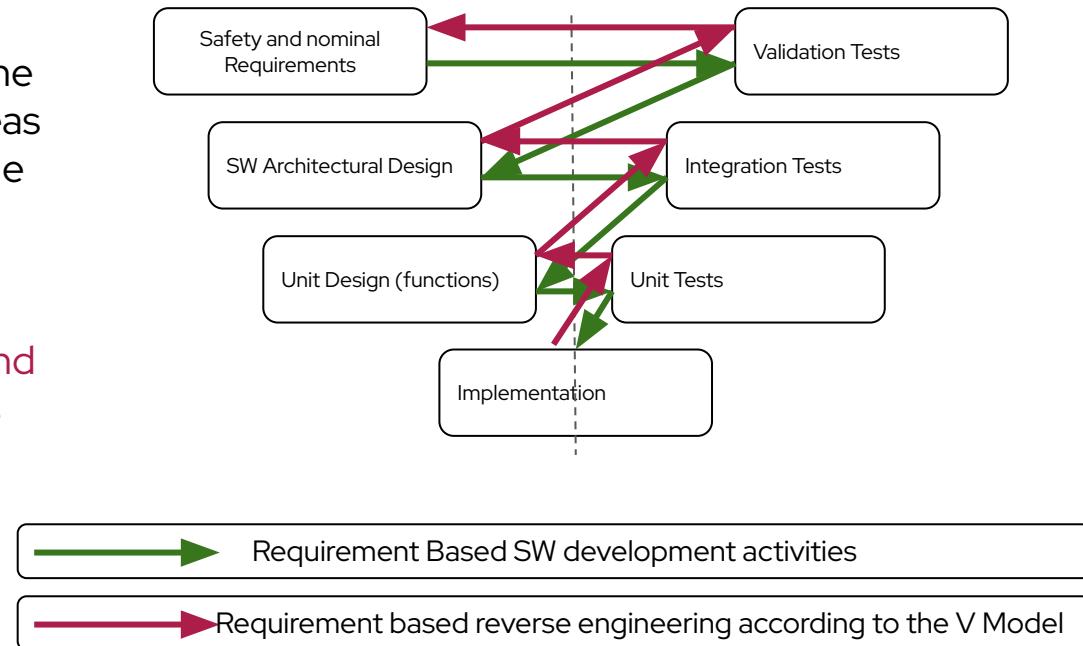
Challenges in managing quality in Linux



Feasibility in applying the V-Model to Linux

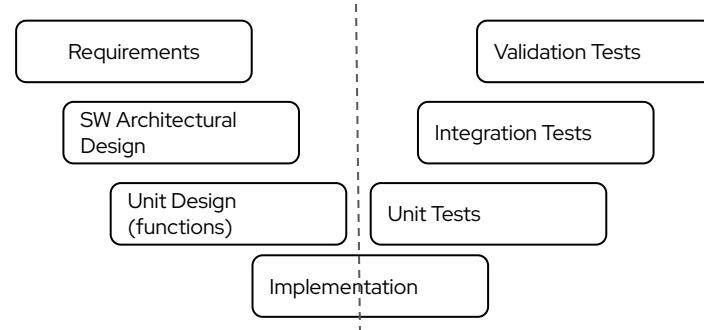
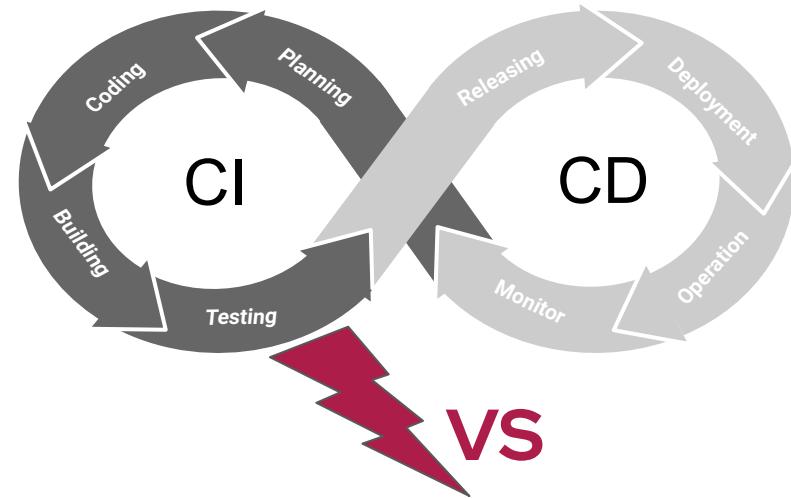
Most of QMS standard demands a SW development process that is compliant to the V-Model and is requirements centric, whereas the open source development model is code centric.

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?

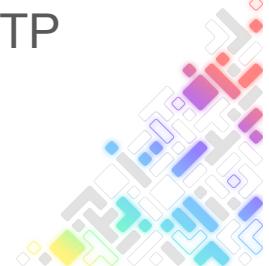
An Open Source Software Quality Management Tool.

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

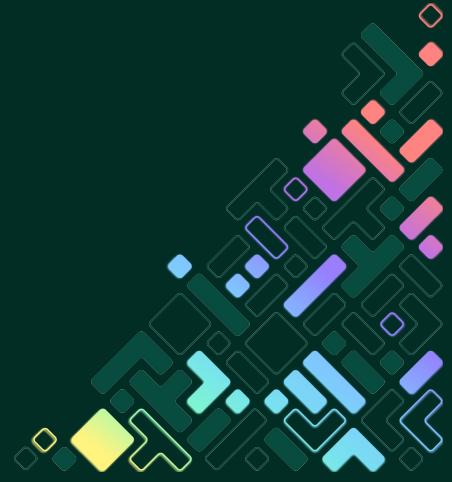


Why BASIL?

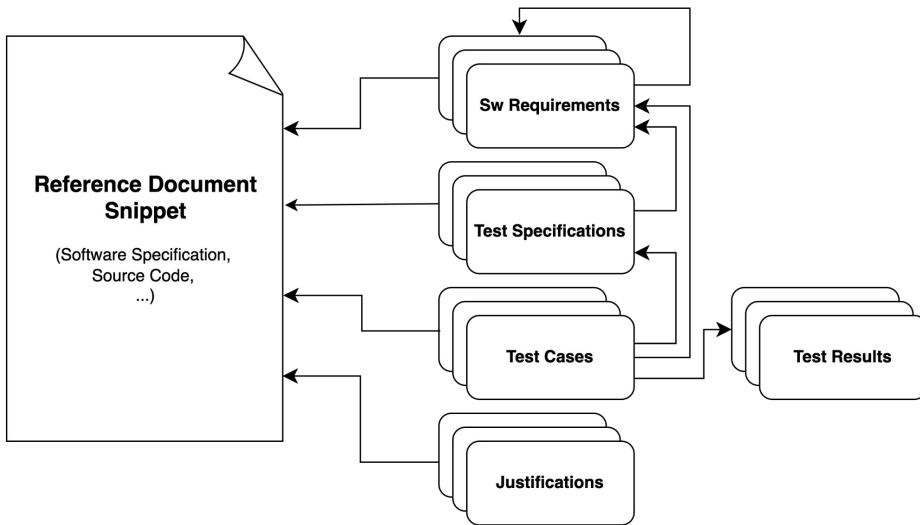
- BASIL fully integrates with the Linux configuration management tools (i.e. git and github). Therefore all the code changes BASIL can detect which QMS evidences are impacted
- BASIL is designed to fit a CI/CD development process by focusing on code and specification first.
Requirements and test work 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

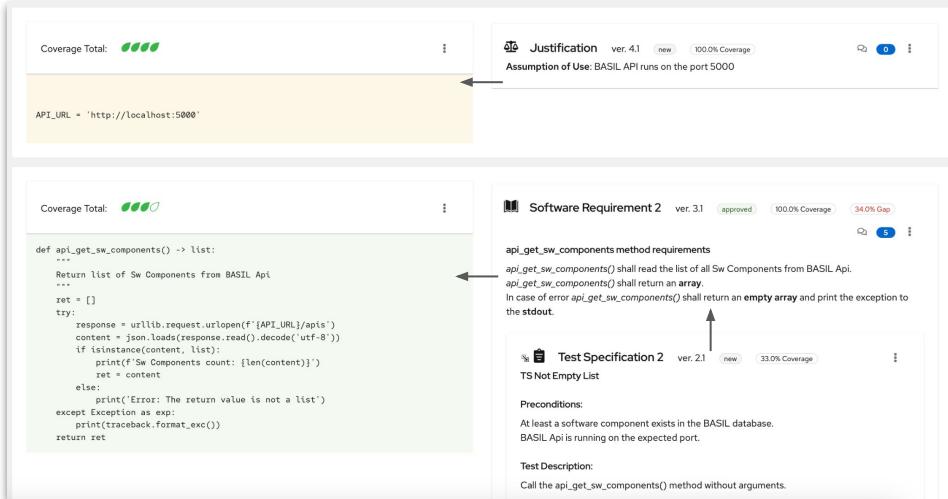
Work Items are created against a snippet of the Reference Document

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

Software Requirement
Work item data and mapping information (section, offset, coverage).

Sw Requirement Data Mapping Section Existing

Status *
New

Sw Requirement Title *
api_get_sw_components method requirements

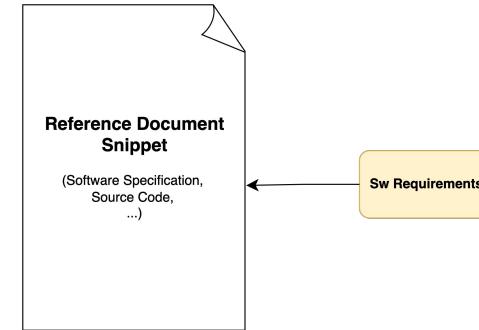
Description *
api_get_sw_components() shall read the list of all Sw Components from BASIL Api.
api_get_sw_components() shall return an **array**.
In case of error *api_get_sw_components()* shall return an **empty array** and print the exception to the **stdout**.

Unique Coverage:
100

Example: Sw Requirement

Work Item data

- Status
- Title
- Description



Work Items management

Software Requirement
Work item data and mapping information (section, offset, coverage).

Sw Requirement Data **Mapping Section** Existing

Section

```
def api_get_sw_components() -> list:  
    ...  
    Return list of Sw Components from BASIL Api  
    ...  
  
    ret = []  
    try:  
        response = urllib.request.urlopen(f'{API_URL}/apis')
```

Offset

97

[Set as unmatching](#) | [Set as Full Document](#)

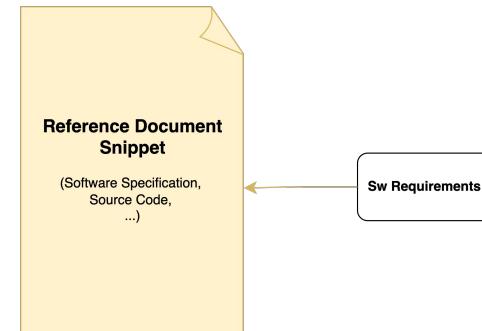
Code Snippet

```
import json  
import os  
import traceback  
import urllib.request  
  
API_URL = 'http://localhost:5000'  
  
def api_get_sw_components() -> list:  
    ...  
    Return list of Sw Components from BASIL Api  
    ...  
  
    ret = []  
    try:  
        response = urllib.request.urlopen(f'{API_URL}/apis')  
        content = json.loads(response.read().decode('utf-8'))  
        if isinstance(content, list):
```

Example Sw Requirement

Relationship data

- Section
- Offset
- Coverage



History

Fork selected Software Requirement

Do you want to continue?

Version 1.2 - 04 Apr 24 17:34

Mapping

- *Sw requirement id:*
5
- *Edited by:*
lpellecc@redhat.com

Version 1.1 - 04 Apr 24 17:34

Work Item

- *Id:*
5
- *Title:*
api_get_sw_components method requirements
- *Description:*
api_get_sw_components() shall read the list of all Sw Components from BASIL Api. *api_get_sw_components()* shall return an array. In case of error *api_get_sw_components()* shall return an empty array and print the exception to the stdout.
- *Status:*
NEW
- *Created by:*
lpellecc@redhat.com

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

Fork selected Software Requirement

Do you want to continue?

Version 1.2 - 04 Apr 24 17:34

Mapping

- *Sw requirement id:*
5 ←
- *Edited by:*
lpellecc@redhat.com

Version 1.1 - 04 Apr 24 17:34

Work Item

Mapping

- *Id:*
2
- *Api id:*
5
- *Sw requirement id:*
2 ←
- *Section:*
def api_get_sw_components() -> list: """ Return list of Sw Components from BASIL Api """ ret = [] try: response =

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.



EMBEDDED
OPEN SOURCE
SUMMIT



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

SPECIFICATION	UNMATCHING WORK ITEMS
Coverage Total: 	
<pre>def api_get_sw_components() -> list: """ Return list of Sw Components from BASIL API ... ret = [] try: response = urllib.request.urlopen(f'{API_URL}/apis') content = json.loads(response.read()).decode('utf-8') if isinstance(content, list): print(f' Sw Components count: {len(content)}') ret += content else: print('Error: The return value is not a list') except Exception as exp: print(traceback.format_exc()) return ret</pre>	<p>Software Requirement 2 ver. 3.1 100.0% Coverage api_get_sw_components method requirements</p>



Reference Document changes

Check Work Item Mapping against a Software Component Specification

Current api api_get_sw_components from code 2

Software Component Specification Url •

https://github.com/elisa-tech/BASIL/raw/main/examples/code/api_get_sw_components_v2.py

SW Requirements

* KO

5 - api_get_sw_components method requirements

7 - script execution requirement

* WARNING

6 - write_sw_components_to_file() method requirement

Test Specifications

Test Cases

Other Justifications

* KO

6 - Python modules used by the script

* WARNING

7 - Constant value that specifies the BASIL API endpoint

Confirm

Cancel

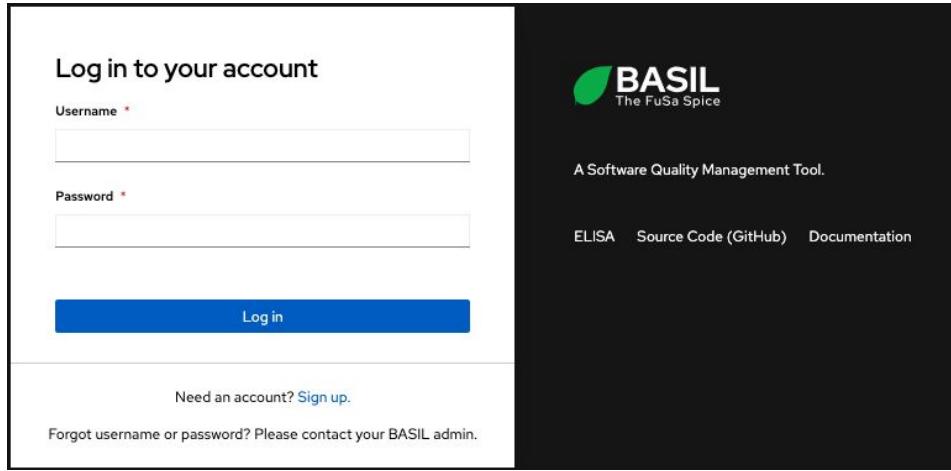
Fix Warnings

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



The image shows two side-by-side screenshots. On the left is a screenshot of a login form titled "Log in to your account". It has fields for "Username" and "Password", both marked with a red asterisk indicating they are required. Below the password field is a blue "Log in" button. At the bottom of the form are links for "Need an account? [Sign up.](#)" and "Forgot username or password? Please contact your BASIL admin.". On the right is a screenshot of the BASIL logo, which consists of a green stylized leaf icon followed by the word "BASIL" in white capital letters, with "The FuSa Spice" in smaller white text below it. Below the logo, the text "A Software Quality Management Tool." is displayed. At the bottom, there are links for "ELISA", "Source Code (GitHub)", and "Documentation".

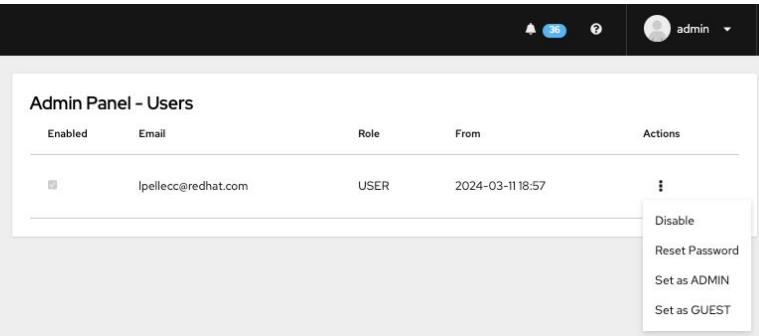
To have a public instance of BASIL available on the web, user management is a mandatory feature.

Goal:

- Define user roles
- Define permissions at Sw Component level



User Management - Roles



The screenshot shows the BASIL Admin Panel - Users interface. On the left is a sidebar with links: Home, User Management (selected), SSH Keys, Libraries, and Useful Links. The main area has a table titled "Admin Panel - Users" with columns: Enabled, Email, Role, From, and Actions. One row is visible for the email lpellecc@redhat.com, which is Enabled, has the Role USER, and was created on 2024-03-11 18:57. The Actions column for this row shows a dropdown menu with options: Disable, Reset Password, Set as ADMIN, and Set as GUEST.

Enabled	Email	Role	From	Actions
■	lpellecc@redhat.com	USER	2024-03-11 18:57	<ul style="list-style-type: none">⋮DisableReset PasswordSet as ADMINSet as GUEST

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

Manage User Permission

Current api first from lib11

User Email: lpellet@redhat.com

Search

Read permission
Permission to read work items and relationships of the selected software component.

Write permission
Permission to add and edit work items and relationships of the selected software component.

Edit permission
Permission to edit parameters of the selected software component.

Owner permission
Manage user permissions to the selected software component.

Confirm Cancel



Collaboration

The screenshot displays a web application interface for managing software development tasks. On the left, there are two code editor panes showing Python code. The top pane contains a configuration variable `API_URL = 'http://localhost:5000'`. The bottom pane shows a function `api_get_sw_components()` which sends a GET request to the API URL and handles the response. In the center, there are three cards:

- Justification**: ver. 4.1, 100.0% Coverage. Assumption of Use: BASIL API runs on the port 5000.
- Software Requirement 2**: ver. 3.1, approved, 100.0% Coverage, 34.0% Gap. Description: `api_get_sw_components` method requirements. It states that the method shall read the list of all Sw Components from BASIL Api and return an array. It also specifies that if an error occurs, it should return an empty array and print the exception to the stdout.
- Test Specification 2**: ver. 2.1, new, 33.0% Coverage. TS Not Empty List. Preconditions: At least a software component exists in the BASIL database. BASIL Api is running on the expected port. Test Description: Call the `api_get_sw_components()` method without arguments.

- 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 screenshot shows the BASIL interface for managing software components. On the left, there's a sidebar with links like Home, User Management, SSH Keys, Libraries, and Useful Links. The main area is titled 'API Listing for code' (Covered 42%). It shows two entries:

ID	API	Version	Owner	Category	Coverage	Notifications	Actions
5	api_get_sw_components	1	lpellicc@redhat.com		0000		
6	api_get_sw_components	2	lpellicc@redhat.com		0000		

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

This screenshot shows the same BASIL interface as above, but it displays a list of notifications on the right side of the screen. There are four notifications listed:

- Test Run for FIRST PASS (green circle)
- Test Run for FIRST has been requested (blue circle)
- Test Run for FIRST NOT EXECUTED (red circle, highlighted with a red box)
- Test Run for FIRST NOT EXECUTED (red circle)

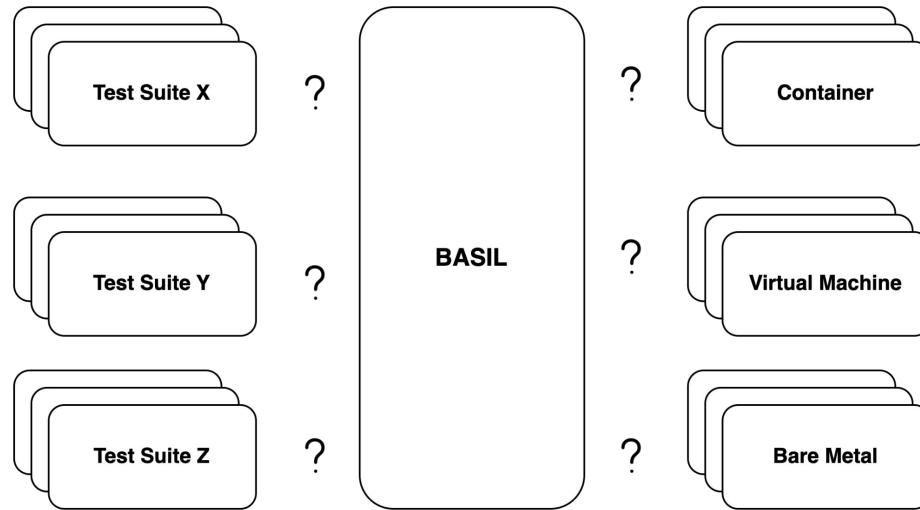
The notifications are timestamped and show details about the test runs.

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

Test Result

Test Runs info execution log log.txt stdout/stderr Report a Bug Artifacts

Search

ID	Repositories	SUT	Result	Date	Bug	Actions
43	fedora vm via ssh	10.31.45.221	pass	2024-04-08 16:03	Re-run	Delete
42	fedora container	container	pass	2024-04-08 15:36	Re-run	Delete

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.

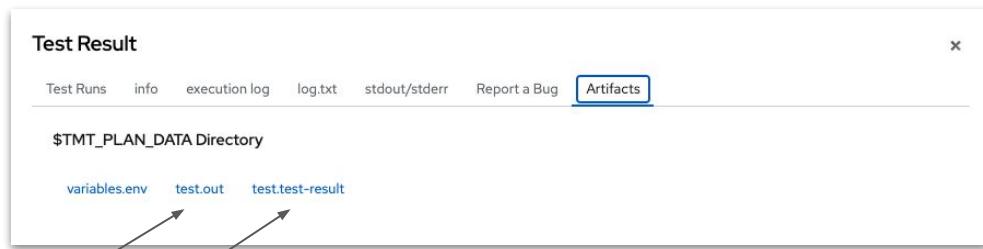
```
1 summary: glibc single test
2 description: glibc single test
3 require:
4   - bison
5   - gcc
6   - git
7   - glibc
8   - make
9 test: |
10 CURRENT_TEST=${CURRENT_TEST:=libio/tst-fopenloc}
11 GLIBC_DOWNLOAD_PATH=$HOME/src
12 GLIBC_BUILD_PATH=$HOME/build/glibc
13 mkdir $GLIBC_DOWNLOAD_PATH || true
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
```

Test Result

Test Runs info execution log log.txt stdout/stderr Report a Bug Artifacts

\$TMT_PLAN_DATA Directory

variables.env test.out test.test-result



SPDX

SPDX Data

```
{ "SPDXID": "SPDX-CODE-EXPORT", "creationInfo": { "created": "2024-03-25T10:44:13Z" }, "dataLicense": "CC0-1.0", "name": "SPDX-CODE-EXPORT", "spdxVersion": "", "documentNamespace": "SPDX-CODE-EXPORT", "files": [ { "SPDXID": "API-5", "attributionTexts": [ { "category": "", "library": "code", "library_version": "1", "implementation_file": "", "implementation_file_from_zow": "", "implementation_file_to_row": "", "rwx_specification_url": "https://github.com/elisa-tech/BASIL/raw/main/examples/code/api_get_sw_components_v1.py", "tags": "python, api, example", "created_at": "2024-03-23 10:34:31.305807" } ], "checksums": [ { "algorithm": "MD5", "checksumValue": "f2a74c45b5e186eff667812e975abdd48" } ], "comment": "Software Component", "filename": "api_get_sw_components", "filetypes": [ "TEXT" ] }, { "SPDXID": "API-SR-2", "attributionTexts": [ { "id": "2", "title": "api_get_sw_components method requirements", "description": "api_get_sw_components() shall read the list of all Sw Components from BASIL API.\napi_get_sw_components() shall return an **array**.\nIn case of error apt_get_sw_components() shall return an empty array** and print the exception to the \"stdout\".", "version": "3", "created_at": "2024-03-23 10:56" } ] }
```

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

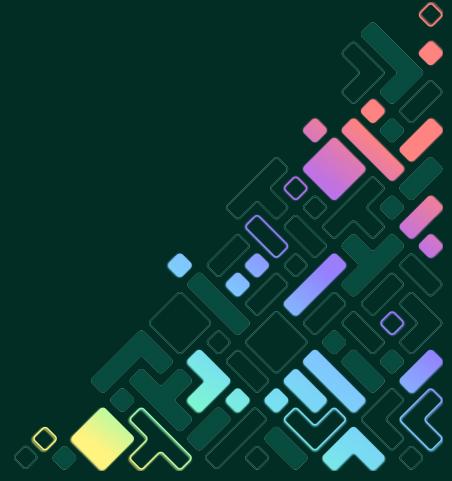
It can be used to simplify the integration in CI/CD and other toolchain or to retrieve information to fill reports



EMBEDDED
OPEN SOURCE
SUMMIT



Examples



Linux kernel syscalls analysis

Coverage Total: 

.SH DESCRIPTION
.BR nanosleep ()
susends the execution of the calling thread
until either at least the time specified in
.I *duration
has elapsed

 Software Requirement 5 ver. 3.2 in review 50.0% Coverage  0 

Thread execution suspension requirement

Following the invocation of `nanosleep()`, and in absense of signals, the invoking thread shall not resume execution before the specified input time has elapsed.

 Test Case 5 ver. 8.1 in progress 100.0% Coverage  0 

LTP nanosleep01c

LTP nanosleep syscall test `nanosleep01.c`

Can be executed using [BASIL tmt example](#) specifying the following set of environment variables:

`CURRENT_SYSCALL=nanosleep`
`CURRENT_TEST=nanosleep01`

 Software Requirement 6 ver. 2.1 in review 50.0% Coverage  0 

Thread integrity requirement

Following the invocation of `nanosleep()`, the invoking thread execution context shall be saved and as `nanosleep()` returns the thread context integrity shall be guaranteed (*The invocation of nanosleep() shall not corrupt the thread context*).

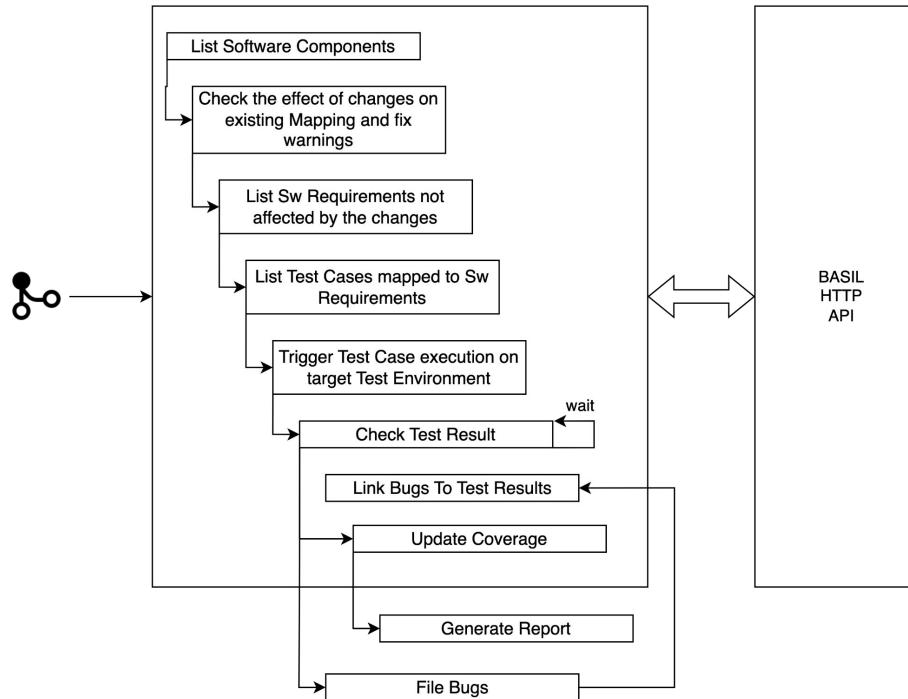
Man Pages are used as a software specification by developers everyday.

BASIL can be used to:

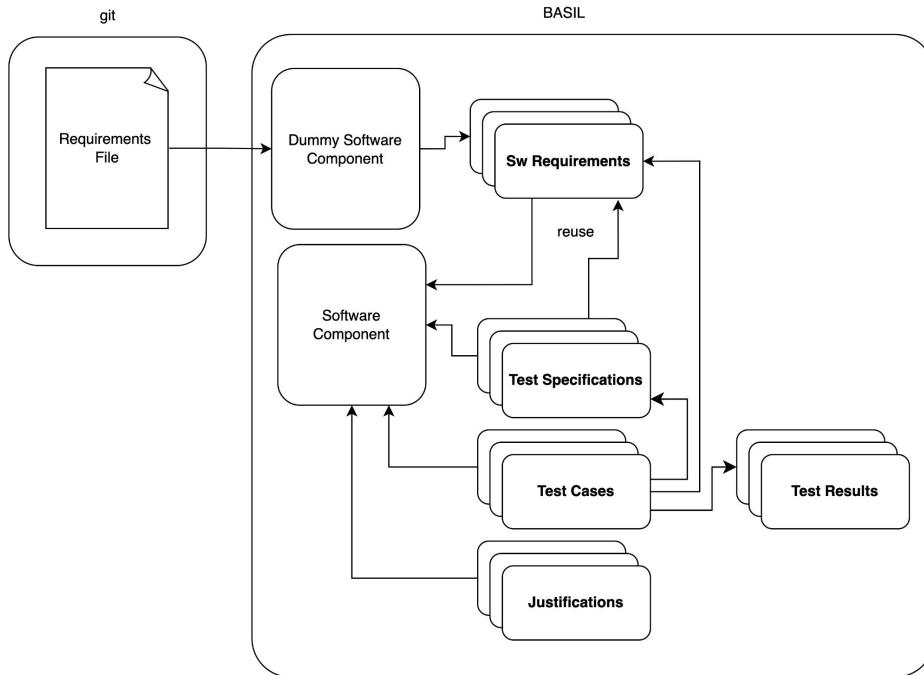
- evaluate current Man Pages test coverage
- to keep LTP project aligned to Man Pages and Source Code



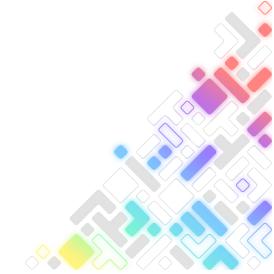
Change request impact analysis example



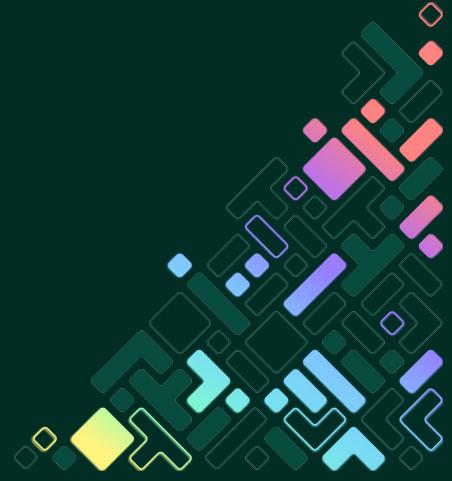
Sw Requirements from files - Zephyr



- 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

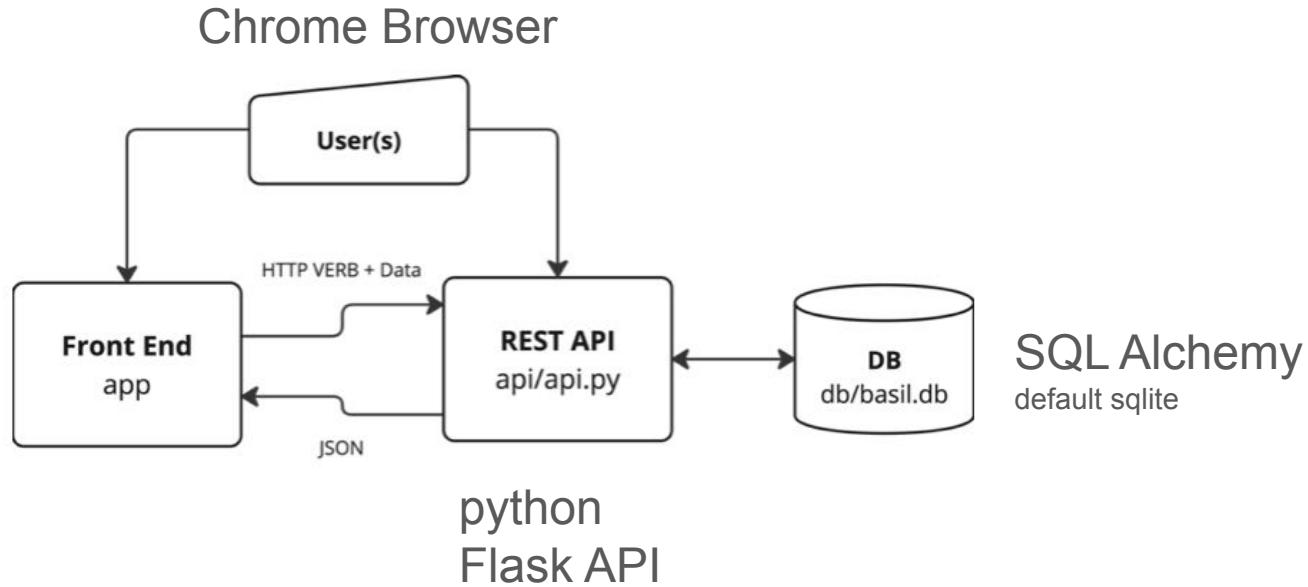


Code



github.com/elisa-tech/BASIL

React
PatternFly



PatternFly

The screenshot shows the PatternFly website. On the left, there's a section titled "Design and build better product experiences in the open with PatternFly". Below it, a paragraph explains that PatternFly is an open source design system. At the bottom of this section are two buttons: "Start designing" and "Start developing". To the right, there's a large dashboard area. The top part of the dashboard shows a circular progress bar at 55% completion. Below it is a "Flyer Deployment" card with the text "Provided by PatternFly" and "Open source". The main dashboard area has sections for "Events" (with a "Recent events" log) and "Status" (a table showing the status of various objects and resources). At the bottom of the dashboard, there's a "Featured blog posts" section and a "Explore our blog" button.

Designed with great attention to Accessibility it provides:

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

Code available at

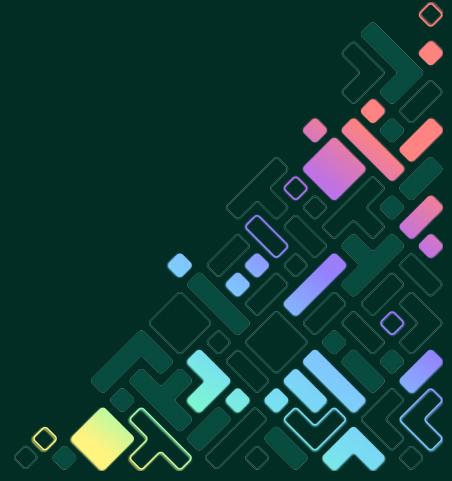
<https://github.com/patternfly>

[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



EMBEDDED
OPEN SOURCE
SUMMIT



Documentation

BASIL

Navigation

Contents:

[How does it works?](#)

[Key Concepts](#)

[How to run it](#)

[Functional Safety](#)

[E2E Testing with cypress](#)

[Notifications](#)

[Test Run and Test Results](#)

[User Management](#)

[Work Items](#)

Quick search

Go



Documentation

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.

Contents:

- [How does it works?](#)
 - [Software Components](#)
 - [Work Item Mapping](#)
- [Key Concepts](#)
 - [Reference Document](#)
 - [Re-Use Work Items](#)

Documentation is available at

<https://basil-the-fusa-spice.readthedocs.io>

github issues

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





SAFETY-CRITICAL SOFTWARE SUMMIT

