rmtoo – Traceability

Kristoffer Nordström

April 11, 2021

Introduction

rmtoo

- File-based requirements tracking tool
    - one file for each requirement
    - group requirements by topics
    - document and requirements LaTeXbased
- Convert requirements into PDF document

An introduction presentation into *rmtoo* and with more details.

Outline

This slideshow provides an overview over the new features not mentioned in the presentation below.

- Introduction into the traceability features
- Export and import of *xlsx* files
- Show missing features and how to solve them

Traceability

Scenario

The idea for this solution emerged when a requirements document with its traceability matrix was finalised. Then the inevitable happened: a change request.

- Requirements have changed, and
- some other code has to be changed as well.

Requirements (I)

The following problems have been solved using *sltoo*

- Keep requirements and code synced
- Update test specification and keep tests synced
- Detect upstream changes to requirements
- Quickly identify affected code-regions
- Traceability matrix must be correct at any given time

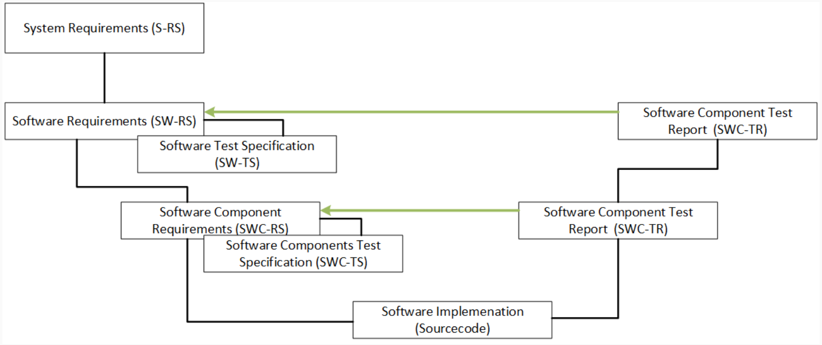
Requirements (II)

- Code and requirements belong together
  - Same repository
  - CI runs unit-tests
  - CI creates traceability matrix automatically
- Unit-tests point to test specification
  - Changes to tests must be reflected in test specification
  - Manual work
  - No silver bullet
  - *Backwards* arrow in V-model
- Changes to test specification must fail CI toolchain
  - For unchanged unit-tests
  - Link not obvious from specification
  - CI tool must validate
  - *Forwards* arrow in V-model

sltoo Traceability

Introduction

- V-Model as reference
- Directions for Traceability
  - Forwards (black)
  - Backward (green)



System Requirements (S-RS)
Software Requirements (SW-RS)
Software Test Specification (SW-TS)
Software Component Test Report (SWC-TR)
Software Component Requirements (SWC-RS)
Software Components Test Specification (SWC-TS)
Software Component Test Report (SWC-TR)
Software Implemenation (Sourcecode)

Figure 1: Traceability Overview

Backwards — Unit-tests

Every specification-item has a name, e.g., `SWC-TS-102`, and a unique hash (more later). Every unit-test lists the specifications it solves.

The following unit-test will test the aforementioned *software component test specification* item *102*.

```
def test_adding_req(self, record_property):
    record_property('req', 'SWC-TS-102-96ac8522')
    assert True
```

Backwards — Traceability Matrix Input

- Running `pytest` will yield a *xunit* file `result.xml`
- This file is used to generate the traceability matrix
- Any unit testing framework can be used if the XML is equivalent

```
<testcase time="0.048" name="rmttest_adding_req" line="89"
 file="rmtoo/tests/RMTTest-Output/RMTTest-Xls.py"
 classname="rmtoo.tests.RMTTest-Output.RMTTest-Xls.RMTTest(
  <properties>
    <property name="req" value="SWC-TS-102-96ac8522"/>
  </properties>
</testcase>
```

Forwards

The previously test requirement `SWC-TS-102` will change and with it it's hash-value.

Hence the test on the previous page will fail the traceability matrix because the hash *96ac8522* has changed.

Time for your engineer to ensure if/what needs to change.

```python
def test_adding_req(self, record_property):
    record_property('req', 'SWC-TS-102-96ac8522')
    assert True
```

Specification Hash

- SHA256 hash calculated over sum of
  - Description,
  - Titel, and
  - Verification Method (if available)
- Rationale is only informative

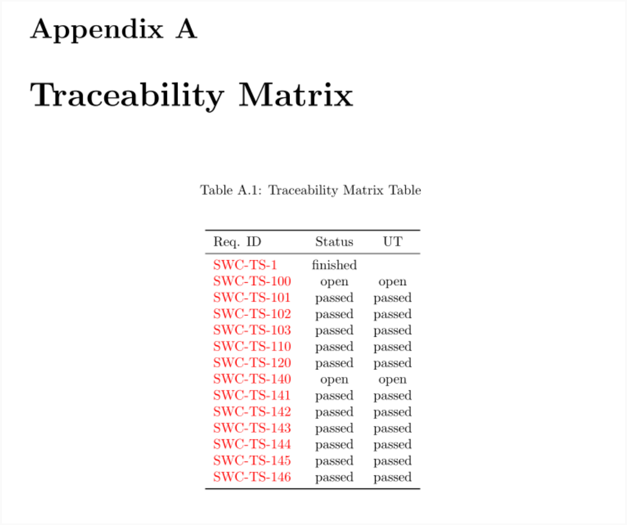
Results

The status *external* will yield the following results:

- *open*
  - No matching requirement ID
- *passed*
  - Matching requirement ID
  - All hashes match
  - Unit-tests passed
- *failed*
  - Matching requirement ID
  - Some/all hashes didn't match, or
  - Unit-tests haven't passed

Example

From the test specification in *rmtoo's* `testspe` folder.



Appendix A

Traceability Matrix

Table A.1: Traceability Matrix Table

Req. ID	Status	UT
SWC-TS-1	finished	
SWC-TS-100	open	open
SWC-TS-101	passed	passed
SWC-TS-102	passed	passed
SWC-TS-103	passed	passed
SWC-TS-110	passed	passed
SWC-TS-120	passed	passed
SWC-TS-140	open	open
SWC-TS-141	passed	passed
SWC-TS-142	passed	passed
SWC-TS-143	passed	passed
SWC-TS-144	passed	passed
SWC-TS-145	passed	passed
SWC-TS-146	passed	passed

Example (II)

Example project with versioned documents: pymergevcd

Future Developements

- Write Parser for *Test Reports*
  - Documents with the correct identifier automatically solve the specification
  - Document Formats:
    - docx (maybe with pandoc)
    - $\LaTeX$
- Cross-Document References
  - *Solved by external*
    - *Solved by* is used in *downwards* direction in the V.
    - Only within document at the moment. Makes merging documents easier.
  - *Depends on external*
    - References requirements in other documents, can be with or without hashes.
    - Think about extending the current *Depends on* handler (deprecated) for use as external (leftwards, upwards) reference.

Excel Support

Rationale

- Good-enough GUI
- Suits love it
- The *Truth* is still in your repository
- Import from Excel to repository
- Export every build to a new Excel file
- Templating for branding

Final Thoughts

Installation

```
pip3 install sltoo==24.5b5
```

Traceability features are in the beta releases.

Public Service Announcement

- Never test against your requirements
    - Always write some form of test specification
    - Consider cucumber for acceptance testing