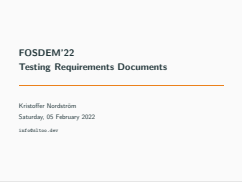
FOSDEM'22
Testing Requirements Documents

Kristoffer Nordström

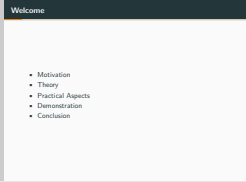
Saturday, 05 February 2022

info@sltoo.dev



Welcome

- Motivation
- Theory
- Practical Aspects
- Demonstration
- Conclusion



- Welcome everyone to FOSDEM22 from my side. Welcome to my talk about req. specification documents and how to test them. It's an honour to be closing this stream.
- Given the topic is requirements specifications, maybe we should specify the problems we're trying to solve.
- So, imagine ...

Motivation

- Automated traceability matrix
- *Consistent* and *up-to-date* documents
- Store *meta-information* with code

Motivation
- Automated traceability matrix
- *Consistent* and *up-to-date* documents
- Store *meta-information* with code

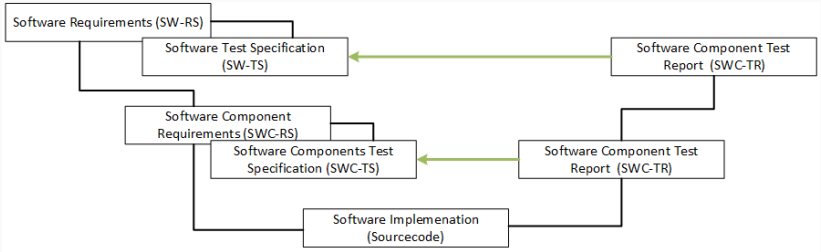
- being a reasonably junior developer and you're tasked with creating a traceability matrix in Word for given artefacts in a space application. The first time round you're doing work that needs to be done either way — fair game. By the time the fifth CR comes along, it gets tiresome.
- Alright, consultant live: new project, new customer, new set of documents to review. I can't recall any details but the fact that links and references didn't add up.
- To use Jon Holt's term in introduction to MBSE: Documents are a live *view* of the system (not pretty pictures). We have CI/CD for software, why not for documents?
- So far, challenges could be handled with any ALM tool on the market.
- But then, the project from the musty crypt, language features from an OS specific compiler. The OS itself isn't available for the target platform. At least there's a specification. Looking like it's been exported from some web based ALM tool that's long gone too.
- It came with one document that contained all requirements and use-cases. It appeared to have been exported from some web-based tool. All the information therein is lost time.
- myself: As you can guess, (FPGA) test-engineer for various space/railway applications 8 yrs
- 4'

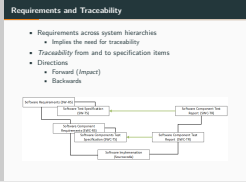
Theory

Theory

Requirements and Traceability

- Requirements across system hierarchies
  - Implies the need for traceability
- *Traceability* from and to specification items
- Directions
  - Forward (*Impact*)
  - Backwards





Let's start with a simple and theoretical example

Here Traceability only from items/issues

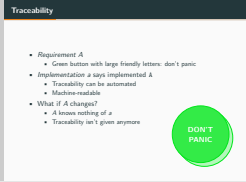
Traceability

- *Requirement A*
  - Red button to shut down system
- *Implementation a* says implemented A
  - Traceability can be automated
  - Machine-readable
- What if *A* changes?
  - *A* knows nothing of *a*

Traceability
PANIC

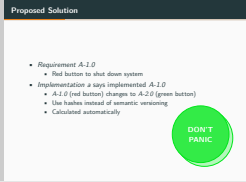
Traceability

- *Requirement A*
  - Green button with large friendly letters: don't panic
- *Implementation a* says implemented `A`
  - Traceability can be automated
  - Machine-readable
- What if *A* changes?
  - *A* knows nothing of *a*
  - Traceability isn't given anymore



Proposed Solution

- *Requirement A-1.0*
  - Red button to shut down system
- *Implementation a* says implemented *A-1.0*
  - *A-1.0* (red button) changes to *A-2.0* (green button)
  - Use hashes instead of semantic versioning
  - Calculated automatically



Why hashes: no tool or manual changes required, it's all derived

Let's see how it looks on an example

Example Requirement

```
Name: VCD Writer Inputs
Topic: ReqsDocument
Description: The output from ...
Rationale: Make the process as ...
Status: external
Owner: development
Effort estimation: 1
Invented on: 2020-05-30
Invented by: default
Type: requirement
```

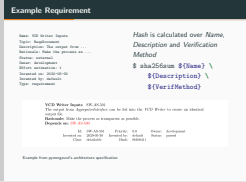
Hash is calculated over *Name*, *Description* and *Verification Method*

```
$ sha256sum ${Name} \
    ${Description} \
    ${VerifMethod}
```

VCD Writer Inputs SW-AS-501
The output from *AggregatorInterface* can be fed into the *VCD Writer* to create an identical output file.
Rationale: Make the process as transparent as possible.
Depends on: SW-AS-500

Id:	SW-AS-501	Priority:	0.0	Owner:	development
Invented on:	2020-05-30	Invented by:	default	Status:	passed
Class:	detailable	Hash:	f8d68d11		

Example from pymergevcd's architecture specification



- Every requirement is in its own file
- A from previous slides is now SW-AS-501
- Version n.0 is now F8D68D11

Testing the Example Requirement

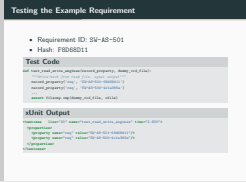
- Requirement ID: SW-AS-501
- Hash: F8D68D11

Test Code

```
def test_read_write_engines(record_property, dummy_vcd_file):
    """Write-back from read file, equal output"""
    record_property('req', 'SW-AS-501-f8d68d11')
    record_property('req', 'SW-AS-500-4c1a395a')
    ...
    assert filecmp.cmp(dummy_vcd_file, ofile)
```

xUnit Output

```
<testcase   line="20" name="test_read_write_engines" time="2.830">
  <properties>
    <property name="req" value="SW-AS-501-f8d68d11"/>
    <property name="req" value="SW-AS-500-4c1a395a"/>
  </properties>
</testcase>
```

Traceability Matrix

Table A.1: Traceability Matrix Table

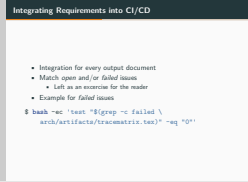
Req. ID	Status	UT
SW-AS-1	passed	passed
SW-AS-100	open	open
SW-AS-101	open	open
SW-AS-102	open	open
SW-AS-200	failed	failed
SW-AS-201	passed	passed
SW-AS-202	passed	passed
SW-AS-300	passed	passed
SW-AS-301	open	open
SW-AS-302	open	open
SW-AS-500	passed	passed
SW-AS-501	passed	passed

Now it should be straightforward to integrate it into any CI pipeline

Integrating Requirements into CI/CD

- Integration for every output document
- Match *open* and/or *failed* issues
  - Left as an excercise for the reader
- Example for *failed* issues

```
$ bash -ec 'test "$(grep -c failed \
    arch/artifacts/tracematrix.tex)" -eq "0"'
```

9'

sltoo in Practice

sl. . . what?

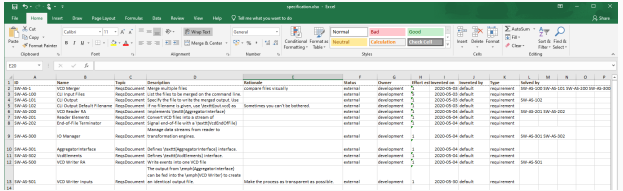
- *sltoo* (currently) a fork of rmtoo
  - text-file based req. tracking tool
- Tracking requirements in text files with git
  - might be ideal for engineering department
- Defining system behaviour is a team effort
- Solution not for everyone / command-line unfamiliar UI

sl. . . what?
- sltoo (currently) a fork of rmtoo
  - text-file based req. tracking tool
- Tracking requirements in text files with git
  - might be ideal for engineering department
- Defining system behaviour is a team effort
- Solution not for everyone / command-line unfamiliar UI

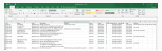
- Fork of the venerable rmtoo (requirements management tool)
- Guess: most people aren't familiar with the tooling/console –> Which tool is "everyone" familiar with?

Excel Workflow (I)

- Defining system collaborative effort
- Familiarity / Ease-of-use
- Consistency of Documents
  - The *Truth* is always in your repository
  - Templating for branding
- Works if all you've got is Office and E-Mail
- Getting Started: Edit example Excel-Sheet



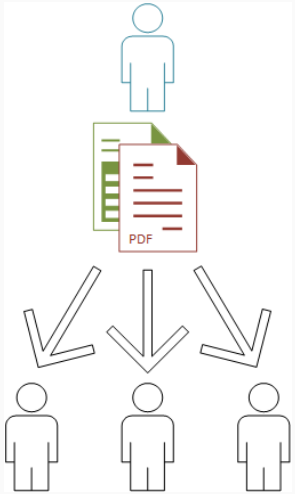
Excel Workflow (I)
- Defining system collaborative effort
- Familiarity / Ease-of-use
- Consistency of Documents
  - The Truth is always in your repository
  - Templating for branding
- Works if all you've got is Office and E-Mail
- Getting Started: Edit example Excel-Sheet



Simply use the excel sheet as long as no documents or verification tasks are necessary

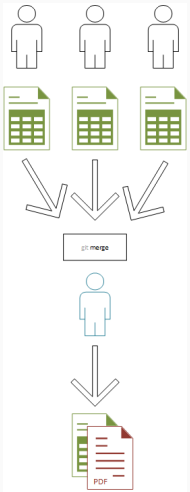
Stepping stone for requirements tracking

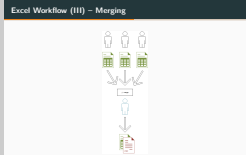
Excel Workflow (II) – Distribution



Excel Workflow (II) – Distribution

Excel Workflow (III) – Merging





- Commit for every author individually -> also put name in commit for easier lookup
- Truth stays in repository
- `git tag` makes new document releases as easy as possible

Document Baseline

Every document has a its own version tag

```
$ git tag -a RS/1A
$ git describe $(git log -n 1 --format=%H -- docs/reqs)
```

The output from `git describe` will be used as document baseline

```
RS/1A - 0aec3ad0              # good
RS/1A-8-g76b3ffe - 76b3ffe4   # tainted
```

Example excerpt from page 7:

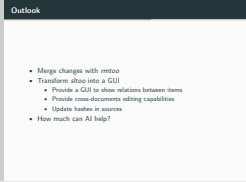
Version SWC-RS/1A-8-g76b3ffe — 76b3ffe4 7

Document Baseline
Every document has a its own version tag
$ git tag -a RS/1A
$ git describe $(git log -n 1 --format=%H -- docs/reqs)
The output from git describe will be used as document baseline
RS/1A - 0aec3ad0 # good
RS/1A-8-g76b3ffe - 76b3ffe4 # tainted
Example excerpt from page 7:
Version SWC-RS/1A-8-g76b3ffe — 76b3ffe4 7

Exercise to reject tainted documents in referenced documents

Outlook

- Merge changes with *rmtoo*
- Transform *sltoo* into a GUI
  - Provide a GUI to show relations between items
  - Provide cross-documents editing capabilities
  - Update hashes in sources
- How much can AI help?



- Replace your current ALM tool with a diverse set of existing/open tools

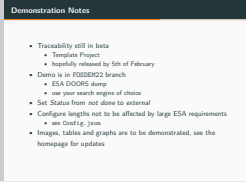
Demonstration

Demonstration

Demonstration Notes

- Traceability still in beta
  - Template Project
  - hopefully released by 5th of February
- Demo is in `FOSDEM22` branch
  - ESA DOORS dump
  - use your search engine of choice
- Set *Status* from *not done* to *external*
- Configure lengths not to be affected by large ESA requirements
  - see `Config.json`
- Images, tables and graphs are to be demonstrated, see the homepage for updates

- Note to self: write a better script



Conclusion

Storytime Revisited

- Requirements shipped with code ✓
  - Including relational meta-information
  - Use commit message to document the *why*
- Traceability matrix automated ✓
- Continuously updated documentation ✓
  - Document Versioning (baselining) ✓

A familiar UI for all stakeholders included

Storytime Revisited
- Requirements shipped with code ✓
  - Including relational meta-information
  - Use commit message to document the why
- Traceability matrix automated ✓
- Continuously updated documentation ✓
  - Document Versioning (baselining) ✓

A familiar UI for all stakeholders included

- Where did we leave off? Have we solved all the problems we set out in the beginning
- Your Jira workspace will be gone
- This concludes this talk and this stream. If everything went according to schedule I'm around for questions

Questions

Questions

Appendix

Questions
Appendix

rmtoo – Introductions

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

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

Traceability Rationale

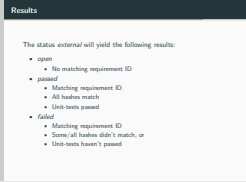
- Traceability for the given requirements
- Bring code and documentation into same repository
- Integrate into build-system
  - Detect upstream changes to requirements
  - Quickly identify affected code-regions
- No silver bullet for verification

Traceability Rationale
- Traceability for the given requirements
- Bring code and documentation into same repository
- Integrate into build-system
  - Detect upstream changes to requirements
  - Quickly identify affected code-regions
- No silver bullet for verification

Results

The status *external* will yield the following results:

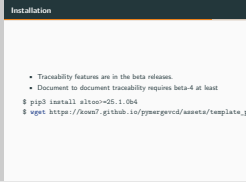
- *open*
  - No matching requirement ID
- *passed*
  - Matching requirement ID
  - All hashes match
  - Unit-tests passed
- *failed*
  - Matching requirement ID
  - Some/all hashes didn't match, or
  - Unit-tests haven't passed



Installation

- Traceability features are in the beta releases.
- Document to document traceability requires beta-4 at least

```
$ pip3 install sltoo>=25.1.0b4
$ wget https://kown7.github.io/pymergevcd/assets/template_p
```

Alternatives

- Sphinx-Needs
- Octane ALM
- Codebeamer
- Aligned elements
- See Wikipedia
- ...

Alternatives
- Sphinx-Needs
- Octane ALM
- Codebeamer
- Aligned elements
- See Wikipedia
- ...

Future Developements

- Write Parser for *Test Reports* ✓
- Documents with the correct identifier automatically solve the specification
  - Document Formats:
    - docx (maybe with pandoc)
    - LaTeX✓
    - Text
    - CAD Files from HW/Mechanical
- GUI with multi-documents support (RS/TS/..)
  - Simplify design process

Future Developements
- Write Parser for *Test Reports* ✓
- Documents with the correct identifier automatically solve the specification
  - Document Formats:
    - docx (maybe with pandoc)
    - LaTeX✓
    - Text
    - CAD Files from HW/Mechanical
- GUI with multi-documents support (RS/TS/..)
  - Simplify design process

Final Thoughts

- Never test against your requirements
- Always write some form of test specification

Final Thoughts
- Never test against your requirements
- Always write some form of test specification

Licensing

This work is licensed under a Creative Commons “Attribution-NonCommercial-ShareAlike 3.0 Unported” license.

Licensing
This work is licensed under a Creative Commons “Attribution-NonCommercial-ShareAlike 3.0 Unported” license.