FOSDEM’22
Testing Requirements Documents

Kristoffer Nordström
Saturday, 05 February 2022
info@sltoo.dev
Welcome

- Motivation
- Theory
- Practical Aspects
- Demonstration
- Conclusion
Motivation

- Automated traceability matrix
- Consistent and up-to-date documents
- Store meta-information with code
Theory
Requirements and Traceability

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

- **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

**DON'T PANIC**
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

```bash
$ 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

<table>
<thead>
<tr>
<th>Id</th>
<th>SW-AS-501</th>
<th>Priority: 0.0</th>
<th>Owner: development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invented on</td>
<td>2020-05-30</td>
<td>Invented by: default</td>
<td>Status: passed</td>
</tr>
<tr>
<td>Class:</td>
<td>detailable</td>
<td>Hash: f8d68d11</td>
<td></td>
</tr>
</tbody>
</table>
Testing the Example Requirement

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

Test Code

```python
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

```xml
<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>
```
Table A.1: Traceability Matrix Table

<table>
<thead>
<tr>
<th>Req. ID</th>
<th>Status</th>
<th>UT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW-AS-1</td>
<td>passed</td>
<td>passed</td>
</tr>
<tr>
<td>SW-AS-100</td>
<td>open</td>
<td>open</td>
</tr>
<tr>
<td>SW-AS-101</td>
<td>open</td>
<td>open</td>
</tr>
<tr>
<td>SW-AS-102</td>
<td>open</td>
<td>open</td>
</tr>
<tr>
<td>SW-AS-200</td>
<td>failed</td>
<td>failed</td>
</tr>
<tr>
<td>SW-AS-201</td>
<td>passed</td>
<td>passed</td>
</tr>
<tr>
<td>SW-AS-202</td>
<td>passed</td>
<td>passed</td>
</tr>
<tr>
<td>SW-AS-300</td>
<td>passed</td>
<td>passed</td>
</tr>
<tr>
<td>SW-AS-301</td>
<td>open</td>
<td>open</td>
</tr>
<tr>
<td>SW-AS-302</td>
<td>open</td>
<td>open</td>
</tr>
<tr>
<td>SW-AS-500</td>
<td>passed</td>
<td>passed</td>
</tr>
<tr>
<td>SW-AS-501</td>
<td>passed</td>
<td>passed</td>
</tr>
</tbody>
</table>
Integrating Requirements into CI/CD

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

```bash
$ bash -ec 'test "$(grep -c failed \n    arch/artifacts/tracematrix.tex)" -eq "0"'
```
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
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 (II) – Distribution
Excel Workflow (III) – Merging
Every document has 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

<table>
<thead>
<tr>
<th>Version Tag</th>
<th>Hash</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS/1A - 0aec3ad0</td>
<td># good</td>
<td></td>
</tr>
<tr>
<td>RS/1A-8-g76b3ffe</td>
<td># tainted</td>
<td></td>
</tr>
</tbody>
</table>

Example excerpt from page 7:

---

Version SWC-RS/1A-8-g76b3ffe — 76b3ffe4
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?
Demonstration
• 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
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
Questions
Appendix
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
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
Traceability features are in the beta releases.

Document to document traceability requires beta-4 at least

```bash
$ pip3 install sltoo>=25.1.0b4
$ wget https://kown7.github.io/pymergevcd/assets/template_project.zip
```
Alternatives

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

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