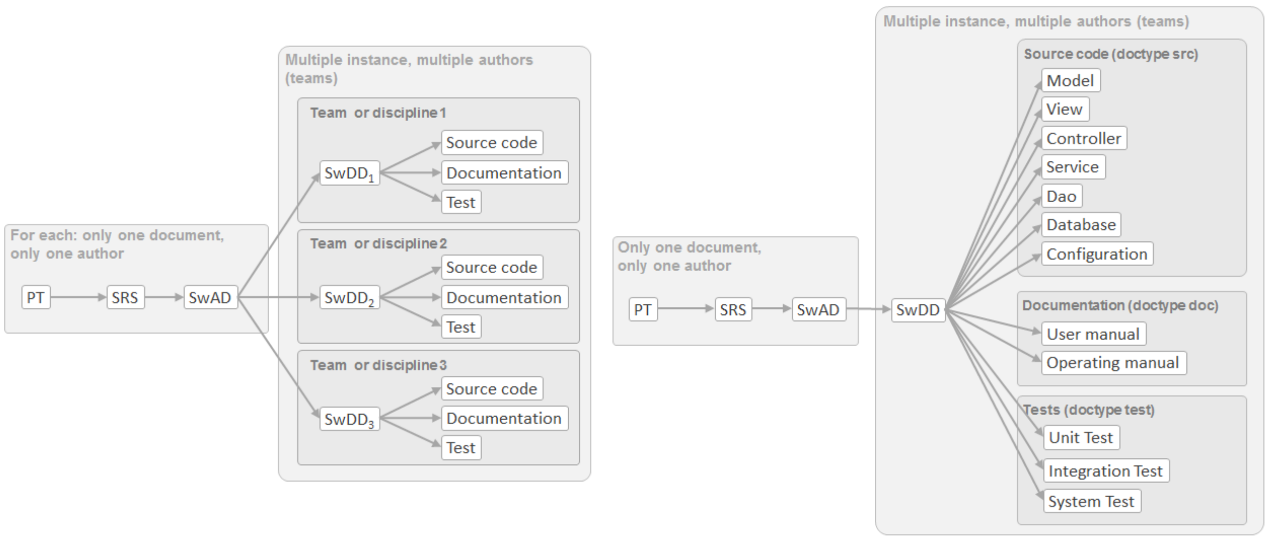
Open Fast Trace Ideas and Proposals

OFT



Document version: 0.3. Draft

Document history

Date	Version	Editor	Description
2017-07-11	0.1	C. Kuhnke	Created document
2017-07-22	0.2	C. Kuhnke	Added lessons learned from a project of the past added details and illustrations for requirements
2018-01-25	0.3	C. Kuhnke	Updated github URL for OFT, renamed open fast track to open fast trace, fixed some typos and grammer, updated Figure 6

Table of contents

References

[1] https://infohub.automotive.elektrobit.com/display/ITINFO/IT+Service+ReqM2
[2] https://ticket.asw.zone/browse/RVDDEV-261
[3] https://github.com/itsallcode/openfasttrace

1 Introduction

1.1 Objective and motivation

Project RVD has experienced some difficulties in requirements tracing. For a long period of time up to now it was not possible to provide sound numbers about the current quality of tracing. In general tracing was far from being complete or only acceptable.

Quality assurance department sticks to claim to have "100% tracing".

Efforts to improve or even fix tracing seem to be high but cannot even be estimated.

Some historic events and changes to processes and maybe mission might have damaged tracing in a way that seems hard to fix.

Additionally several experienced project members on different levels tend to agree that current available tooling is rather part of the problem than part of a solution.

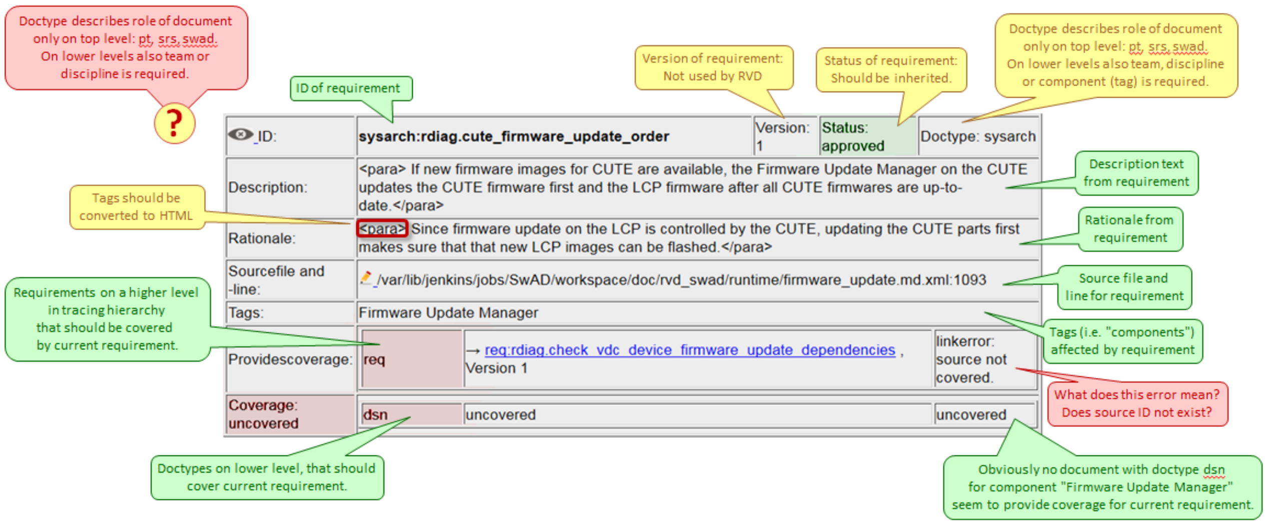
This document focusses on the tooling and provides requirements and proposals to improve tooling for requirements tracing.

ReqM2 has some deficits: Usability, wording, slow, hard to use, hard to understand, customise to processes, feature filtering does not work as expected.

1.2 Scope and audience

- Developers
- Architects
- Project management
- Product owners
- Quality assurance

1.3 Overview ReqM2


Doctype describes role of document only on top level: pt, srs, swad. On lower levels also team or discipline is required.
?
ID of requirement
Version of requirement: Not used by RVD
Status of requirement: Should be inherited.
Doctype describes role of document only on top level: pt, srs, swad. On lower levels also team, discipline or component (tag) is required.
Description text from requirement
Tags should be converted to HTML
Rationale from requirement
Requirements on a higher level in tracing hierarchy that should be covered by current requirement.
Source file and line for requirement
Tags (i.e. "components") affected by requirement
What does this error mean? Does source ID not exist?
Doctypes on lower level, that should cover current requirement.
Obviously no document with doctype dsn for component "Firmware Update Manager" seem to provide coverage for current requirement.

ID:	sysarch:rdiag.cute_firmware_update_order	Version: 1	Status: approved	Doctype: sysarch
Description:	<para> If new firmware images for CUTE are available, the Firmware Update Manager on the CUTE updates the CUTE firmware first and the LCP firmware after all CUTE firmwares are up-to-date.</para>			
Rationale:	<para>Since firmware update on the LCP is controlled by the CUTE, updating the CUTE parts first makes sure that that new LCP images can be flashed.</para>			
Sourcefile and -line:	/var/lib/jenkins/jobs/SwAD/workspace/doc/rvd_swad/runtime/firmware_update.md.xml:1093			
Tags:	Firmware Update Manager			
Providescoverage:	req	→ req:rdiag.check_vdc_device_firmware_update_dependencies , Version 1	linkerror: source not covered.	
Coverage: uncovered	dsn	uncovered	uncovered	

Figure 1 Sample requirement in current output of ReqM2

2 Requirements

- XML input (*.oreqm ?)
- light-weight script language
  - python
  - groovy
- runnable on as many platforms as possible, i.e.
  - easy to port to different environments
  - have minimal dependencies
  - comment: this would also point to java for example
- output:
  - HTML + JavaScript?
  - containing text and as much of original input as possible
- Features
  - Tags (Components) from architecture
  - groups of components assigned to a team / discipline
  - milestones from SRS
  - status of requirements: approved / not approved
  - different levels: SRS, SwAD, SwDD, impl, test
  - support arbitrary doctypes in source files (e.g. interface, impl, database, service, ...) but cluster to common abstract groups, e.g. "src"
- arbitrary tags in input artefacts, convert (by XSLT?) to unified target DTD
- filtering: inspect by
  - by level
  - component(s)
  - covered / uncovered / partially

2.1 Aspects of features

For particular features, especially filters some aspects need to be considered. Some general aspects apply to several features.

Feature	Aspects
Tags (Components) from architecture	(1) (2)
groups of components assigned to a team / discipline	(1) (2) (3)
milestones from SRS	(2)
status of requirements: approved / not approved	(2)
support arbitrary doctypes in source files (e.g. interface, impl, database, service, ...) but cluster to common abstract groups, e.g. "src"	(4)
Key	
(1) see section 2.6 Architectural components vs. project teams or disciplines *(2) see section 2.5 Multiple inheritance* *(3) see section 2.7.1* Different levels and cardinalities *(4) see section 2.7.3* Abstract high-level overview	

Table 1 Aspects of features.

2.2 General Alternatives and ideas

- XSLT or data structure inside script?
- Use separate Browser window for filtering controls?
- Keep objects and relations as tree (graph) in memory, maybe without texts
    - id
    - doctype
    - provides coverage
    - needs coverage
    - status (only for top-level or defined doctypes)

2.3 How, where, when to filter?

Current Javascript of ReqM2 is too weak.

- Correct filtering needs to consider tree structures.
- E.g. Milestone and Status are defined only in top-level and should be inherited.
- Coverage also needs to be inherited.
- Also partial inheritance needs to be considered

On the other hand filter could be implemented heterogeneously:

- filter for Milestone, Status, coverage in pre-processing (XSLT or script)
- other filters can be implemented in JavaScript

2.4 Consequences

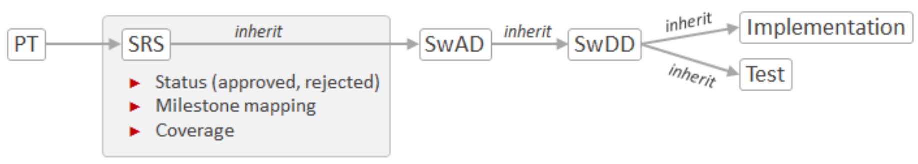


Figure 2 RVD artefacts relevant for tracing and inheritance of attributes

If some attributes are defined only in artefacts of particular doctypes or on top-level, then "tracing" can be considered only with respect to those artefacts.

For example tracing of MasterData cannot be determined by looking only at SwAD and SwDD (for MasterData) but also SRS needs to be considered, since only this document defines milestones and status (like approved, rejected …). These attributes need to be inherited.

Another option would be to create a precompiled version of SwAD including the current definitions from upstream artefacts.

2.5 Multiple inheritance

A single requirement in SwAD can (partially) cover several requirements from SRS. This introduces partial inheritance.

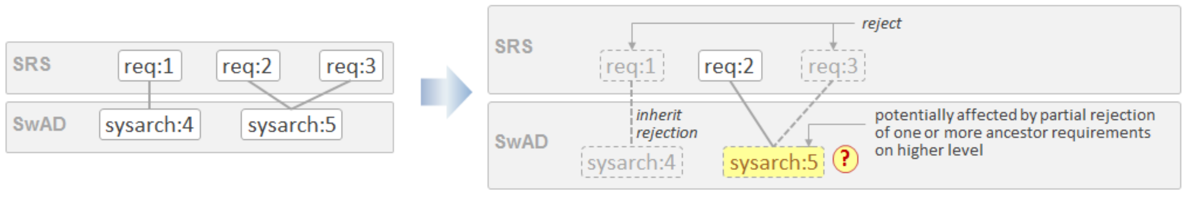


Figure 3 Example for multiple inheritance in tracing hierarchy

For example, if a requirement req:1 in SRS is rejected, then derived requirement sysarch:4 should also be rejected, since it only covers req:1.

If rejecting req:3 then the consequences for sysarch:5 are not clear, since sysarch:5 not only covers req:3 but also req:2 and req:2 is not rejected. So in general sysarch:5 cannot be assumed to be rejected also. The status of sysarch:5 is unclear, potential consequences for sysarch:5 are

- reject, too
- keep unchanged
- update content to respect reject of req:3

So the formal effect of rejecting req:3 on sysarch:5 can only be to change the status of sysarch:5 to "potentially affected by partial rejection of one or more ancestor requirements on higher level". Final resolution needs to be manually by a human editor. He can then decide to change sysarch:5 in one or more of the following ways

- remove coverage for the ancestor requirement(s)
- mark sysarch:5 as "rejected"
- update content of sysarch:5

2.5.1 Generalisation

The same holds true for milestones.

The described effects can happen on any of the levels of tracing hierarchy.

2.6 Architectural components vs. project teams or disciplines

This section describes a lesson learned from a previous project that is relevant for requirements tracing.

2.6.1 Situation

Architectural document in the project used to divide the system into components and subcomponents, which is generally a good idea.

Also the project developing the system was divided into several teams. Each team worked on one or more disciplines. Teams were distributed over several locations, some locations in different countries and time zones.

Unfortunately the components named in the architectural document were on a much more detailed level than the teams. So every time someone was reading the architectural document he was stumbling across names of components he never had heard of.

Everyone was wondering about

- which team a particular component might be assigned to,
- which team might be supposed to create or work on that component and
- which team he could turn to if he had questions on that component.

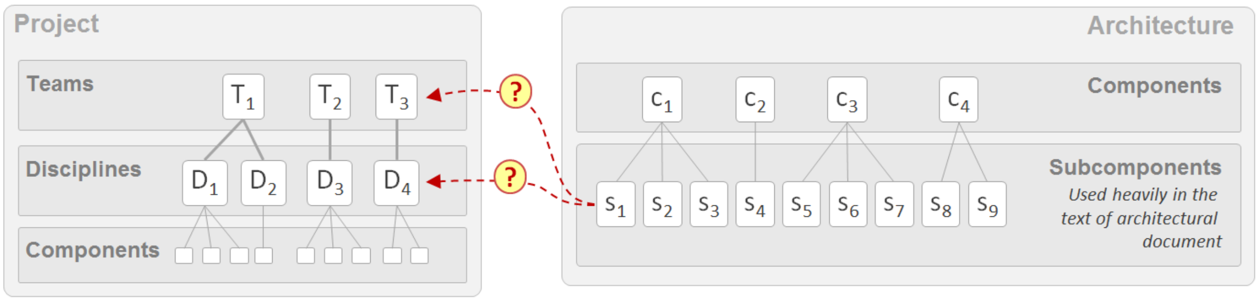


Figure 4 Project teams and disciplines vs. architectural components and subcomponents

These kinds of problems affected project management, requirements engineer, integrator, testers, product owners and many developers. So in general it turned out that this design decision upon the architecture of the architectural document was a bad idea.

2.6.2 Conclusions

However if such might happen to another project and if we assume that this situation cannot be changed (and it cannot be changed easily) then requirements tracing should at least provide support to ease this kind of problems.

First of all the relation R between architectural components and project teams or disciplines needs to be made explicit and described in a formal way. Requirements tracing needs to maintain a table T that tells for each architectural component which project team or discipline it belongs to.

architectural component	project team or discipline
s_1	T_3
s_2	T_3
s_3	T_2
s_4	T_2
s_5	T_1
s_6	T_1

Table 2 Example for relation of architectural components and teams or disciplines from project

2.6.3 Consequences

Who should maintain the table T describing architectural components and assigned project teams or disciplines?

As architectural components are defined by the architectural document, the architectural document might also introduce new components from time to time. How could other people in the project notice?

For that the authors of the architectural document also need to maintain the relation between architectural components and project teams or disciplines. Each version of architectural document needs to come with an updated table T. Table T should be part of each delivery of the architectural document.

Even better, though, would be to just avoid having different granularities in architectural document and in project setup.

2.7 Doctypes

2.7.1 Different levels and cardinalities

On higher levels of requirements tracing hierarchy a doctype identifies uniquely a specific document and its author:

doctype	Document	Author
pt	Product target	Customer
req	SRS, System requirements specification	Requirements manager
sysarch	SwAD, Software architectural design	System architect

Table 3 doctypes for requirements on higher levels of requirements tracing hierarchy

On lower levels you could find the following doctypes

doctype	Document	Author
dsn	SwDD, Software detailed design	Developers
src	Source code	Developers
test	Tests	Developers
doc	Documentation	Developers

Table 4 doctypes for requirements on lower levels of requirements tracing hierarchy

In opposite to higher levels

- artefacts on lower levels have multiple instances
- each instance lives in the scope of a different discipline and may be maintained by a different team

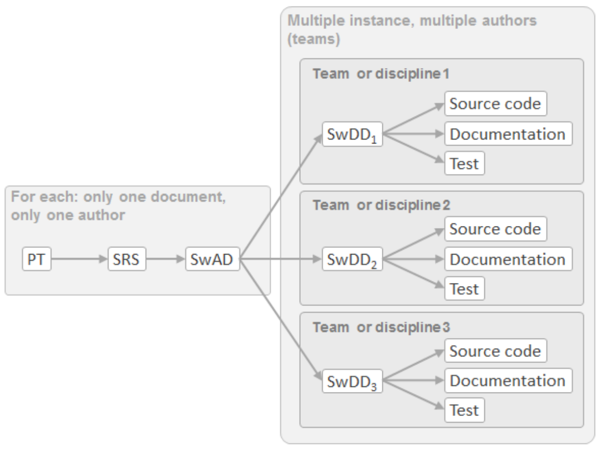


Figure 5 Different cardinalities for artefacts on different levels of requirements tracing hierarchy

That means to identify or measure the tracing quality for a particular discipline or team it is not sufficient to specify the doctype but additionally you need to specify the discipline or team.

From that we derive the requirement to support an additional filter criterion: discipline or team.

2.7.2 Different doctypes

On higher levels of requirements tracing hierarchy there is usually no need to create additional doctypes. On lower levels developers usually have a detailed design and also a detailed architecture for each component they create and maintain:

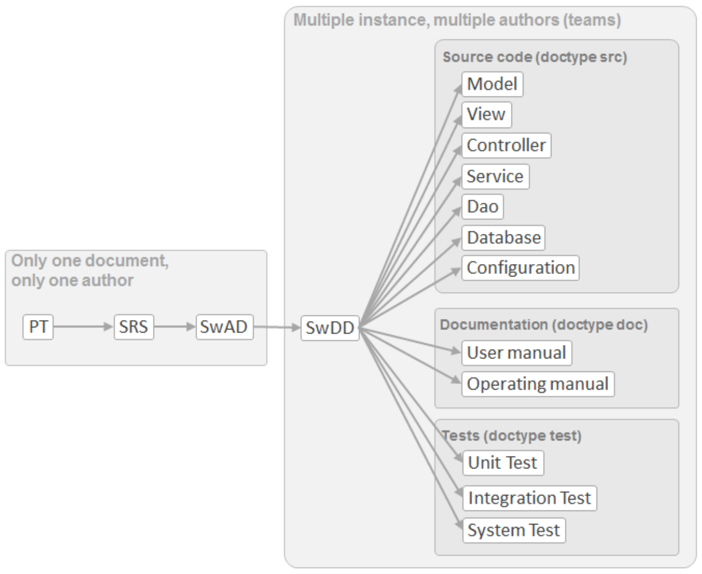


Figure 6 Additional doctypes on lower levels of requirements tracing hierarchy

A software developer might create a single requirement in SwDD, hopefully covering one or more requirements from SwAD, but also requiring coverage in several different implementation layers, e.g. in model, view and controller.

Additionally the developers might need to maintain one or more different documentation artefacts.

Additionally developers should create and maintain automated tests. Also these tests can (and usually do) address different levels of the software: unit tests, integration tests, system tests.

2.7.3 Abstract high-level overview

For a high level overview project management might prefer an abstract view. On this level it might be required to aggregate various doctypes on source code level to a single doctype "src" for "source code". The same can be imagined for documentation and also for tests.

In the current understanding it usually should be enough to just

- hide detailed doctypes on top level
- sum up the statistics for all detailed doctypes into their common category

For actual tracing the original detailed doctypes are required to be kept. Otherwise tracing results would be incorrect:

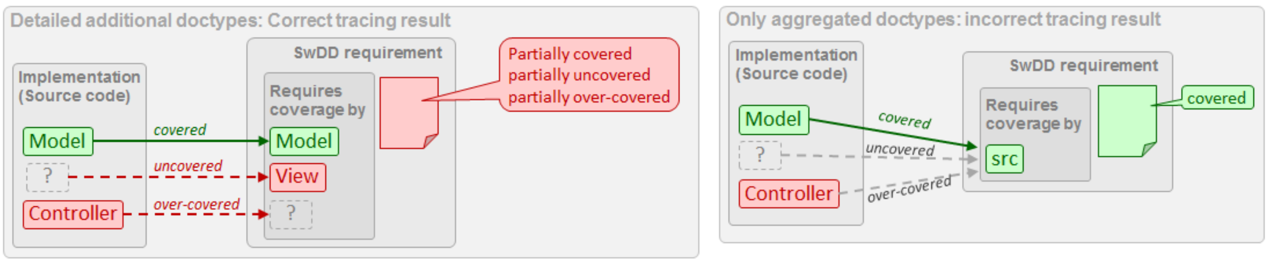

Detailed additional doctypes: Correct tracing result
Implementation (Source code)
Model
?
Controller
covered
uncovered
over-covered
SwDD requirement
Requires coverage by
Model
View
?
Partially covered partially uncovered partially over-covered
Only aggregated doctypes: incorrect tracing result
Implementation (Source code)
Model
?
Controller
covered
uncovered
over-covered
SwDD requirement
Requires coverage by
src
covered

Figure 7 Incorrect tracing results if using aggregated detailed doctypes too naively

As detailed doctypes are defined by the developers, the developers might also introduce new doctypes from time to time.

For that the developers also need to maintain the relation between the detailed doctypes and the aggregated doctypes. Each version of the artefacts maintained by the developers needs to come with an updated relation. The relation should be part of each delivery by the developers.

detailed doctype	aggregated doctype
model	src
view	src
controller	src
user manual	doc
operating manual	doc
utest	test
itest	test
stest	test

Table 5 Example for the relation of detailed doctypes to aggregated doctypes

3 Next steps

- Collect valid input files
- get explanation of all fields currently visible in ReqM2 output
- get explanation for each xml tag in oreqm
- How is milestone mapping encoded in SRS?

4 Organisation and budgeting of development

Currently Open Fast Trace is hosted as an open source project. Initial efforts were made during private time.

- EB or project RVD make a donation to open source project Open Fast Trace
- EB or project RVD sponsor additional implementation efforts on Open Fast Trace
- Developers continue development in private time
- Open source project Open Fast Trace sells current implementation to EB or project RVD

List of figures

List of tables