OETCS/WP7/07.3.5

openETCS

ITEA2 Project
Call 6 11025
2012 – 2015

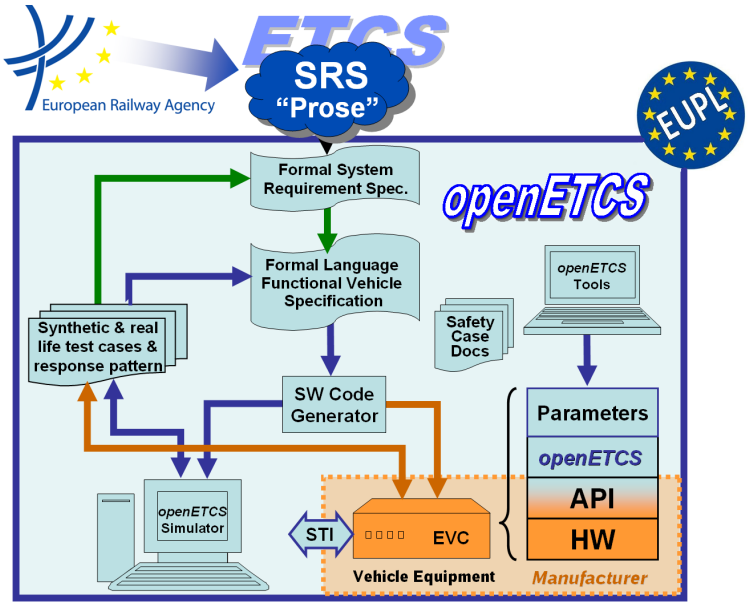
Work-Package 7: “Toolchain”

Traceability Architecture in OpenETCS

WP7 Proposition

Cecile Braunstein and Moritz Dorka

December 2014



Funded by:

Federal Ministry of Education and Research

Région de Bruxelles-Capitale

GOBIERNO DE ESPAÑA MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

This page is intentionally left blank

Work-Package 7: “Toolchain” **OETCS/WP7/07.3.5**
December 2014

Traceability Architecture in OpenETCS

WP7 Proposition

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature Cécile Braunstein (University Bremen)	signature ()	signature ()	signature ()

Cecile Braunstein
University Bremen
Moritz Dorka
DB

OpenETCS : Position Paper on traceability

Prepared for openETCS@ITEA2 Project

Abstract: This document presents apropostion to the tool chain traceability architecture.

Table of Contents

Document Information

Document information	
Work Package	WP7
Deliverable ID or doc. ref.	O7.3.5
Document title	Traceability Architecture in OpenETCS
Document version	00.02
Document authors (org.)	Cécile Braunstein (Uni.Bremen)

Review information	
Last version reviewed	
Main reviewers	

Approbation			
	Name	Role	Date
Written by	Cécile Braunstein	WP7-T7.3 Sub-Task Leader	06.02.2014
Approved by			

Document evolution			
Version	Date	Author(s)	Justification
00.00	17.12.2014	C. Braunstein	Document creation

1 Traceability Definition and OpenETCS tool chain Proposition

1.1 Traceability Process

From the Subset026 SRS specification we want to be able to trace all artifacts that implement it.

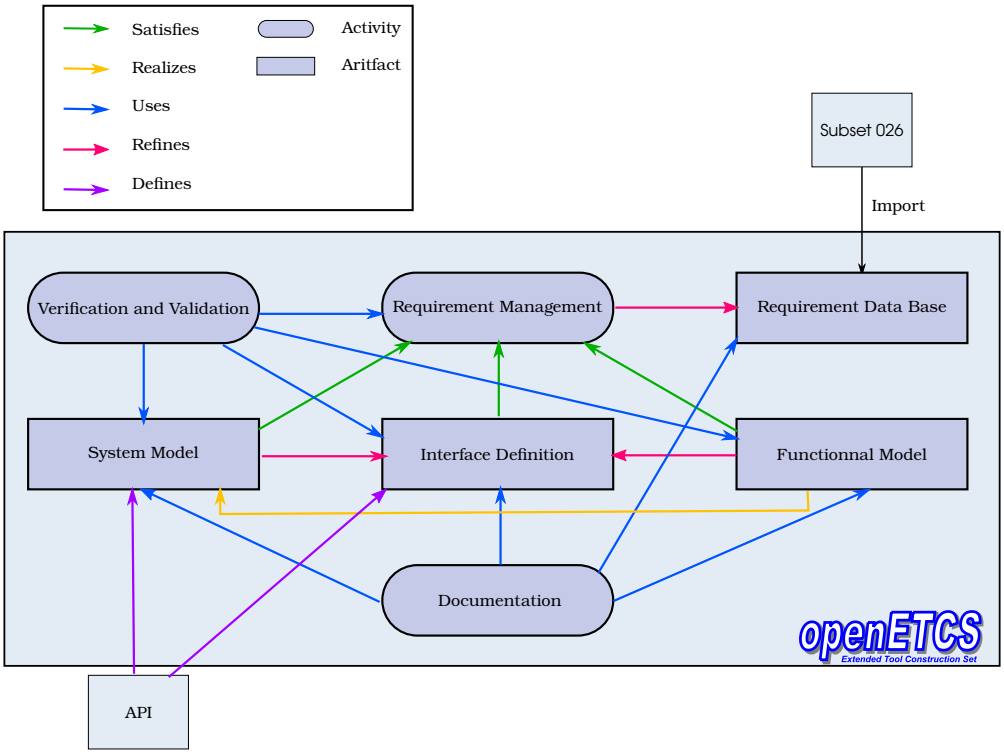


Figure 1. Traceability architecture between artifacts

Figure 1 highlights the different artifacts or activities and their links. Traceability activities should allow to trace a requirement within all the artifacts and/or activities that are using it.

The goal of traceability is that, from any artifacts produced, we are able to track which requirement it realizes, implements or refers to. These links can take different form and be done by different tools. Two traceability tools will be evaluate before being integrated in the OpenETCS tool chain: ReqCycle and Reqtify.

The present solution proposes a first "by hand" approach, to deal with traceability without having all the proper tools already integrated into the tool chain.

1.2 First Solution

Figure 2 proposes a first solution to handle traceability. The basic idea is to all refers to the same requirements data base. This data base is itself directly imported from the subset-026 word document. The data base provides the list of requirements as well as an unique identifiers for

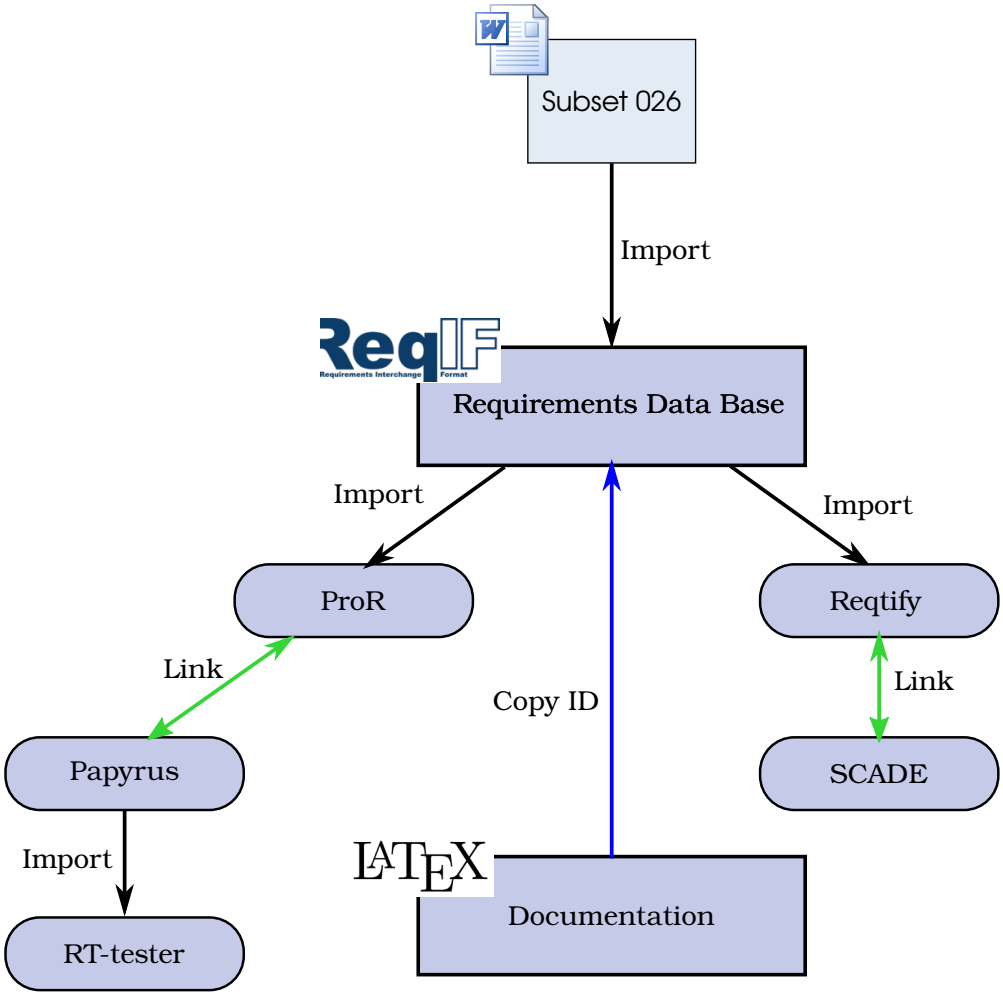


Figure 2. Traceability architecture first solution

each requirements. These identifiers are fixed and cannot be modified by any other tools. But it may be possible for requirement management tools to refine these requirements but they should guarantee the following rules:

1. The requirement's structure is not modified.
2. No requirement should be deleted.
3. Newly added requirements should be either refinement or a decomposition of an existing one.
4. The identifier pattern should be respected.
5. Identifier should stay unique.

1.2.1 TODO ReqCycle Evaluation

1.2.2 TODO Reqtify Evaluation

2 Activities Details

2.1 Subset-026 import

The Subset-026 import is realized by a script transforming the Word document into a req-IF format file.

2.1.1 Script description

The script generates a hierarchical tree of all traceworthy artifacts in each chapter of subset-026. Each artifact shall be uniquely addressable via a tracestring.

The script need a name

2.1.2 Unique ID definition

Take the following example:

3.5.3.7 If the establishment of a communication session is initiated by the on-board, it shall be performed according to the following steps:

a) The on-board shall request the set-up of a safe radio connection with the trackside. If this request is part of an on-going Start of Mission procedure, it shall be repeated until successful or a defined number of times (see Appendix A3.1).

If this request is not part of an on-going Start of Mission procedure, it shall be repeated until at least one of the following conditions is met:

- Safe radio connection is set up
- End of Mission is performed
- Order to terminate communication session is received from trackside

Figure 3. Traceability architecture between artifacts

2.1.2.1 Guideline

The scope of a single requirement ID is a paragraph of text (there are six such paragraphs in the above example). requirement IDs are hierarchical. The hierarchy is a direct mapping of the hierarchy in the original subset026 text. Levels are separated by a dot. There is a requirement at each level (i.e. you may truncate the requirement ID to any level and it stays valid).

2.1.2.2 How to

Suppose we want to trace the fifth paragraph in the above example i.e

- `End of mission is performed`

1. Let traceString be the variable to store the result.
2. Find the current running number of the base list. That is the list which includes the chapter number. In this example this number equals `3.5.3.7.` Set traceString to this number.

3. Count the number of paragraphs in this list item starting with 1 and append this number in square brackets to the traceString if it is greater than 1.

Note: For the first iteration in the example there is only one such paragraph (`If the establishment...`). Hence, we do not append anything. In the second iteration there are two such paragraphs (`The on-board shall...` and `If this request is not ...`). Hence, the second one will receive an `[2]` appendix.

1. Until you arrived at your target paragraph: Append any running number of sub-lists and remove leading or trailing characters (such as braces). If the current sub-list is bulleted then the level string always becomes `[*][n]` (with n being the running number of that bullet starting at 1). Prefix this new level with a dot (`.`) and append it to the traceString.

Note: `a)` is the identifier of one such sub-list item. The trailing brace will be removed. The bullet points form another (less significant) sub-list.

1. Do step 3.
2. Do step 4 or break.
3. traceString is now the fully qualified requirementID.

This will result in the following requirement ID:

```
3.5.3.7.a[2].[*][2]
```

2.2 Requirement Database

The requirement data base is a ReqIF file-based.

2.3 Requirement management

The requirement management allows us to have a look at the set of requirements and performs some actions. The set of requirements can be automatically imported from a Req-IF file.

The requirement management should be use to refine the high-level requirements from the import. It can be use to give more precise information about :

- Which requirements are implementable
- Who is assigned to implement a set of requirements
- How and if the requirement is decomposed.
- TODO (list to be refined)

In the OpenETCS tool chain this tasks is done with ProR. Alternatively to deal with the closed source path the requirement management may be done by Reqtify Gateway of SCADE.

2.4 System Model

The system model defines block and interfaces between those blocks that realizes the specification. This is done with Papyrus.

The link with the requirement may be included via requirements diagram with a direct link of requirement in ProR The explanation to performs the link may be found here ProR-Papyrus proxy. The links may be viewed through Papyrus an ProR and the requirement may be directly apply from ProR to a Papyrus elements.

2.5 Interface Definition

Should define the interfaces between the architecture artifacts. It is used and set up to facilitate team working together on a big architecture. Its definition comes from the requirements but can also be refines by the modeling team without changing the existing implemented requirements.

2.6 Functional model

The function model is the implementation of the system defines in the system model. It should covers the set of implementable requirements.

2.7 TODO Verification and Validation