ProR-based Traceability

Specification for openETCS

1 Problem Description

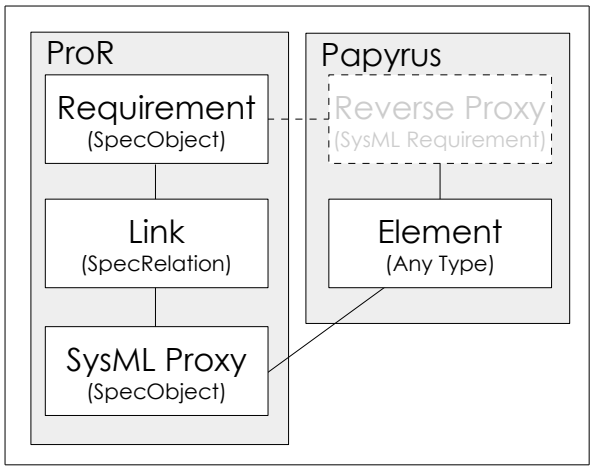
One high-priority need for WP3 and WP4 in openETCS is the traceability between ProR requirements and Papyrus SysML models. Several options have been evaluated, in particular at the Paris traceability workshop on February 7th.

As the most promising approach, ProR-based traceability has been identified. In particular, this kind of traceability has already been realized for Event-B models (link). This implementation convinced the stakeholders that such a solution would provide all the required features, in particular support for change management.

Alternative solutions have been discussed (Reqtify, ReqCycle), but none has all the required features, and none could be provided as quickly as a ProR-based solution.

2 Architecture

For this solution, *SysML Proxy* elements are created that hold (1) a reference to the traced *Element* and (2) a textual representation of that element (cache). The actual trace is realized as a *Link* to the *Requirement*. Now the traceability exists in ProR.



Upon rendering The *SysML Proxy*, the textual representation of the element will be retrieved in real time from *Element* , showing only current content to the user and updating the cache at the same time, if necessary.

Optional: So far, the solution would not allow traceability from SysML to the requirements (no need to even touch the Papyrus model). To allow this, in addition to the above, the tool could, upon linking, do the following: Create a *Reverse Proxy*, in the form of a SysML Requirement, which would be linked (using SysML mechanisms) to *Element*. The ID would hold the corresponding ReqIF ID, the Name field would hold the ReqIF human-readable ID and the Description would hold the requirements text.

3 Use Cases

The following uses cases assume that the *Reverse Proxy* is not created. This could be implemented at a later time.

3.1 Actors

The following Actors exist:

User: The person who creates and uses the traceability through the user interface.

System: The system sometimes needs to perform actions on its own, although these are typically indirectly triggered by the user.

3.2 UC_Create_Link

Precondition:	The SysML Model Explorer is open, the specification editor is open
Steps:	The user drags one or more *Elements* from the Model Explorer onto a *Requirement*.
Result:	A *SysML Proxy* for the dragged model is retrieved (if it already exists) or created (if not). A *Link* is created, if it does not exist yet. The *Requirement* is the Source, the *SysML Proxy* is the Target.
Questions:	• If the *Link* already exists, should another link be created? Technically this is problem and can make sense (e.g. to express different relations between the same source and target), but for now I would recommend against it (to prevent confu-

	sion). • Should the direction of source and target be reversed? This has little to do with traceability, and more a question of presentation in the UI

3.3 UC_Change_Requirement

Precondition:	A *Link* from a *Requirement* to an *Element* has been created.
Steps:	The user edits the requirements text.
Result:	The *Link* is marked with a flag indicating that the Source (the *Requirement*) has changed.

3.4 UC_Change_Model_Element

Precondition:	A *Link* from a *Requirement* to an *Element* has been created.
Steps:	The user edits the *Element* in Papyrus
Result:	The *Link* is marked with a flag indicating that the Target (the *Element*) has changed.
Questions:	The mechanism for observing the linked *Elements* is located in ProR, and therefore, changes will not be noticed if the requirements model is not open. Therefore, upon opening the requirements model, it should initiate a scanning of the SysML model. This could be skipped for a first implementation: The moment the user renders an *Element*, the scanning would take place anyway.

3.5 UC_Deleting_Link

Precondition:	A *Link* from a *Requirement* to an *Element* has been created.
Steps:	The user removes the *Link* from the model
Result:	The *Link* is gone
Questions:	In addition, the *SysML Proxy* could be removed (if no other *Links* point to it), but I see no need for this.

3.6 UC_Deleting_Requirement

Precondition:	A *Link* from a *Requirement* to an *Element* has been created.
Steps:	The user removes the *Requirement* from the model
Result:	The *Requirement* is gone. In ProR, the *Link* and *SysML* Proxy would still exist. The *Link* is marked with a flag indicating that the Source (the *Requirement*) has changed.
Questions:	

3.7 UC_Deleting_Element

Precondition:	A *Link* from a *Requirement* to an *Element* has been created.
Steps:	The user removes the *Element* from the model (in Papyrus)
Result:	The *Element* is gone. In ProR, the *Link* and *SysML Proxy* would still exist. The *Link* would be marked as "Target (*Element*) changed. The link target (the *SysML Proxy*) would show a warning message in red, indicating that the target cannot be found any more. The id and cached information about the target will be intact (and could be shown as well).
Questions:	