Eclipse ProR

by the RMF Community

June 24, 2014

Chapter 1

Introduction

ProR is the user interface of the Eclipse Requirements Modeling Framework. This handbook strives to become a complete reference for ProR. In addition, it will provide a tutorial, making it as easy as possible for new users to get started with requirements engineering.

1.1 Conventions

In this book, we use the following conventions:

Information. Additional, useful information or tips are marked like this.

Warning. Warnings and marked like this.

Examples are marked like this.

1.2 Acknowledgements

Many parties were involved in creating RMF we would like to thank the core team that made it possible.

The roots of this project were created by Andreas Graf, Michael Jastram and Nirmal Sasidharan, who joined their various projects together. Their efforts were financed by the research projects itea Verde and FP7 Deploy. Together, they brought the project to the Eclipse Foundation, where it has been an active project ever since. Figure 1.1 shows four of the five RMF Committers at a joint coding session (missing is Andreas Graf).

Figure 1.1: The RMF team during a Sprint in April 2012 in Düsseldorf, Germany

Chapter 2

Tutorial

2.1 Basic ReqIF Concepts

A ReqIF model is a structured collection of natural language requirements. However, it comes with its own terminology. For instance, a "Requirement" is called "SpecObject" in ReqIF:

A ReqIF model has some similarities to a Excel Spreadsheet, although there are some notable differences. It's simply meant as a starting point to get a feel for ReqIF.

The following table compares a Spreadsheet model with a ReqIF model, introduces the terminology and explains it:

Specification. (Excel-equivalent: Sheet) A ReqIF model consists of an arbitrary number of Specifications. The Specification is the "Container" for the Requirements. Think of an Excel Document, that allows you to create an arbitrary number of Sheets. There are two differences to Excel: (1) The requirements in the Specification are references (which means that the same requirement can appear in multiple places); (2) The content of Specification is hierarchical.

SpecObject. (Excel-equivalent: Row) A SpecObject represents the actual Requirement. A requirement typically has a number Attributes. Again compared with Excel, each row in a Sheet represents a requirement. In contrast to Excel, the ReqIF model may contain SpecObjects that do not appear in any Specification (whether this is useful is a different question).

Attribute. (Excel-equivalent: Cell) Besides the actual text of the requirement, typical Attributes include ID, Status, etc. Note that there are no "standard" Attributes. The ReqIF model contains the definitions of the Attributes. Here the Excel analogy starts to break down. In Excel, each row has the same columns. Different SpecObjects may have different sets of Attributes.

SpecType. (Excel-equivalent: Column configuration) Each SpecObject has a SpecObjectType. The SpecObjectType contains a list of Attributes for the SpecObject. For instance, the SpecObjectType "Headline" may have only one Attribute "HeadlineText". Another SpecObjectType "Requirement" may have three Attributes, "ID", "Description" and "Status". A Specification may then contain a mixture of SpecObjects with different types.

There are many more concepts, but this is enough to get us started.

Let's look at a concrete example to understand this. Here is a snippet of a Specification:

The table shows the first four SpecObjects, as visualized in a Specification. The tree-like structure is recognizable: INF-1 is a node with three children, REQ-1, REQ-2 and REQ-3 (this can be seen by the indentation). Let's look at INF-1 and REQ-1. If they would be selected in the GUI, the Property pane would show their Attributes, as it is shown to the right.

INF-1 has two attributes, "Description" and "ID". The SpecObjectType is "Information Type" (shown as the root in the properties view).

REQ-1 has three attributes, "Description", "ID" and "Status". The SpecObjectType is called "Requirements Type". Let's have a closer look at that one.

The "Requirements Type" SpecObjectType is shown in the picture, with arrows indicating how the Attributes relate to the SpecObjectType (a simple one-to-one relationship). A SpecObjectType has one entry for each Attribute, consisting of a name and a Datatype. For instance, the first entry has the name "ID" and the datatype "T_ID_REQ". Note that multiple Attributes may have the same Datatype: "Description" and "Status" both have the Datatype "T_String32k".

Last, The Datatypes must be defined as well. In this example, there are two Datatypes, "T_String32" and "T_ID_REQ". These are finally based on a number of standard types that ReqIF supports.

NOTE: As of this writing, we only implemented the simple and the enumeration ReqIF types. This leaves out the complex ones (rich text, attachments, embedded files, etc.) for now.

2.2 Tutorial 1: Creating a basic ReqIF Model

In this Section, we will build a ReqIF model from scratch, step by step.

2.2.1 Install ProR

You can download ProR from the Download page. Alternatively, you can install ProR in any Eclipse-Installation via its update site (also listed on the Download page). Using the update site allows you to use the Eclipse Update mechanism, something that is currently not yet enabled for the standalone version.

2.2.2 Create the Model

- Make sure that you have at least one Project in the Workspace;
- Select File > New > Reqif10 Model;
- Select a Project and name the file (ending in .reqif), then click "Finish";
- In the Window named "file.reqif", double-click on "Requirements Document" button in the "Specifications" panel.

After this, your window should look more or less as follows:

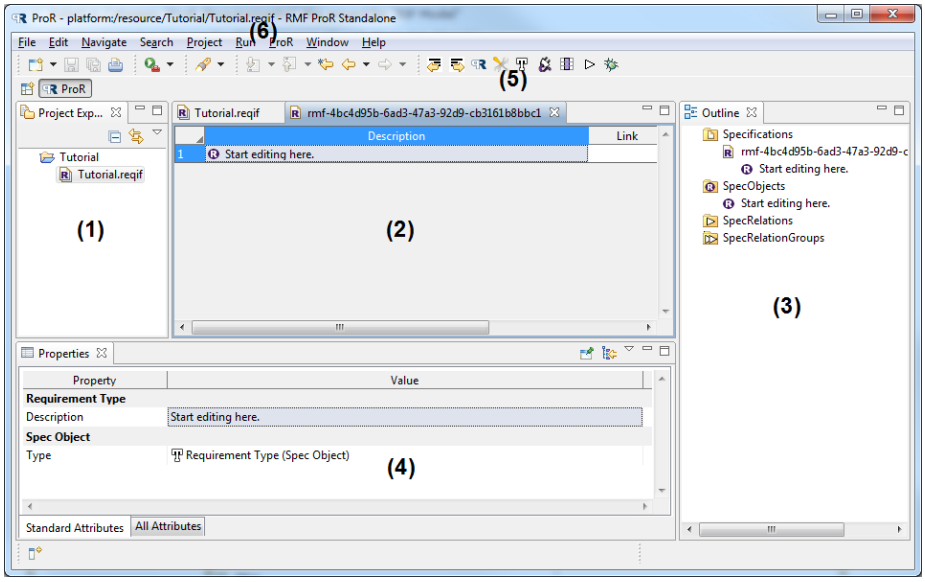

Figure 2.1: The ProR user interface

You will see your ReqIF file in the (1) Project Explorer or the regular Navigator.

The Editor (2) shows your Specifications.

In the Editor, you see the SpecObjects that exist in this Specification. There is currently only one, with the description "Start Editing".

The Outline (3) has three folders:

- "Specifications" shows the Specifications in the ReqIF. You can expand the tree to expose the hierarchy of SpecObjects in the Specification.
- "SpecObjects" shows all SpecObjects in the ReqIF as a flat list. Keep in mind that SpecObjects in Specifications are references. In contrast, this folder shows all SpecObjects in the ReqIF model, no matter whether they appear in any Specification.

- "SpecRelations" shows all SpecRelations in the ReqIF as a flat list. For now, we will ignore SpecRelations.

The properties of a selected Element are shown in the Properties view (4). As the only Requirement in the model is selected, we see its SpecObjectType ("Requirements Type") and its only Attribute ("Description") with the value "Start editing here.". There are two tabs "Standard Attributes" and "All Attributes" at the bottom of the Properties view. The "Standard Attributes" tab shows you all standard attributes of the selected element. The "All Attributes" shows all existing ReqIF attributes of the selected element.

When a ReqIF Editor is active, there are also additional tool bar items (5) and an additional menu (6).

2.2.3 Customizing the SpecType

To get familiar with the system, next we will:

- Add two more attributes to the SpecObjectType called "ID" and "Owner"
- We will show those Attributes in the Specification

To add new attributes, we open the Datatype Configuration dialog with ProR > Datatype Configuration.

The resulting Dialog has a folder for SpecTypes and Datatypes. Currently, there is only one Datatype (T_String32k) and two SpecTypes, one called "Requirements Type" with one Attribute "Description" and one called "Specification Type" with one Attribute "Description".

We add more Attributes to "Requirements Type" by right-clicking "Requirements Type" and selecting "New Child > Attribute Definition String". This will create a new element. Upon selecting, we can rename it in the lower pane. We do this twice for "ID" and "Owner". We assign both Attributes the same Datatype, "T_String32k". In the end, the dialog should look as follows:

Upon closing the dialog, little will have changed - the Specification still shows just two columns, Description and Link. However, if you select the requirement, you will see the new Properties (ID and Owner) in the Property view.

2.2.4 Showing the new Attributes in the Specification

To show the new Attributes in the Specification, we have to configure the Specification columns. We do this by selecting ProR > Column Configuration.

The resulting Dialog shows one entry for the one and only Column of the Specification. By clicking on the "Add Column" icon at the top of the dialog, we can add two more columns. We do this and name them "ID" and "Description" (in the lower pane). Feel free to rearrange the order with drag and drop, as you see fit. When all is done, the dialog should look as follows:

Note that you have to provide Strings for the columns for the same reason that we used Strings for the Labels earlier: This way we can easily match multiple SpecObjects of different types.

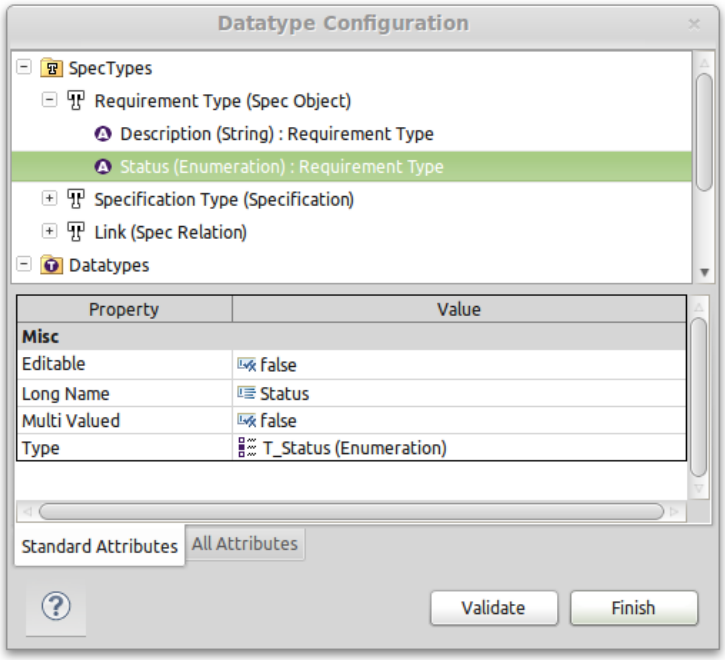

Figure 2.2: Datatype Configuration Dialog

You can actually adjust the width of the columns simply by dragging the column headers.

2.2.5 Adding new SpecObjects

Now we can finally add SpecObjects by right-clicking on a row in the Specification. In the context-menu, there are two submenus: "New Child" and "New Sibling". "New Child" is the only way to produce a hierarchical structure.

In both menus, there are three entries "Spec Hierarchy", "Adding SpecObjects" and "SpecObject (Requirement Type)". Some background is needed here:

We said before that Specifications contain references to SpecObjects. A SpecHierarchy is the "Wrapper" that allows the hierarchical structure and that points to the referred SpecObject. Usually, we don't have to be concerned with them. Therefore the second option: If selected, a new SpecHierarchy is created and associated with a new SpecObject, which in turn is set immediately to the given SpecObjectType. If we had more than just one SpecObjectType (besides "Requirements Type"), there would be an entry for each type in the context menu.

To continue the exercise, select the child "SpecObject (Requirement Type)". Now we have two SpecObjects. Create a few more values and enter some values. You can edit directly in the Specification, as you would in Excel. Play around

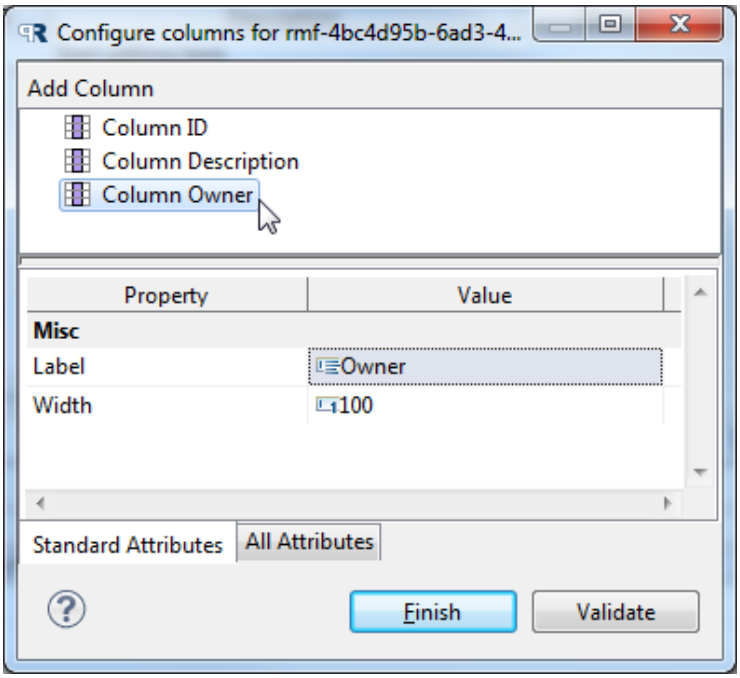

Figure 2.3: Column Configuration

with Hierarchies.

You can drag and drop SpecObjects as a sibling or as a children. The highlighting feedback helps you to find out whenever you shifting a SpecObject as sibling or as a children. For instance, if you are dragging a SpecObject over another one, the entire cell will be highlighted. This means, that the SpecObject will be assigned as a children to the dropped SpecObject.

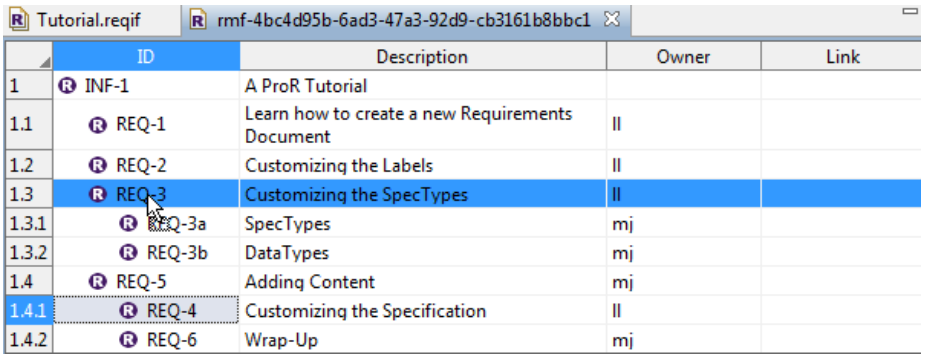

Figure 2.4: Drag and Drop as Child Object

If you are dragging a SpecObject between two rows, you get also a visual feedback an the SpecObject will be assigned as sibling:

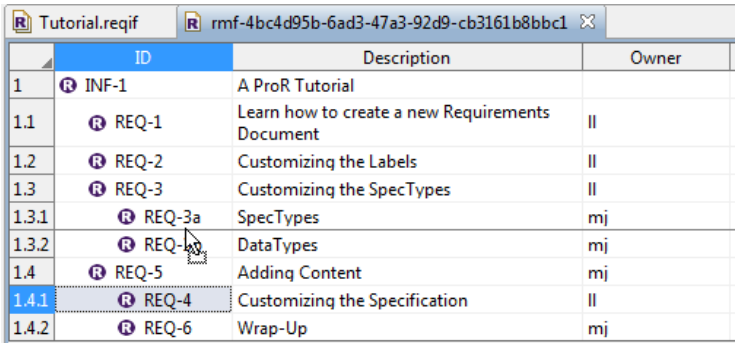

Figure 2.5: Drag And Drop as Sibling Object

After some playing around, our Specification may look like this:

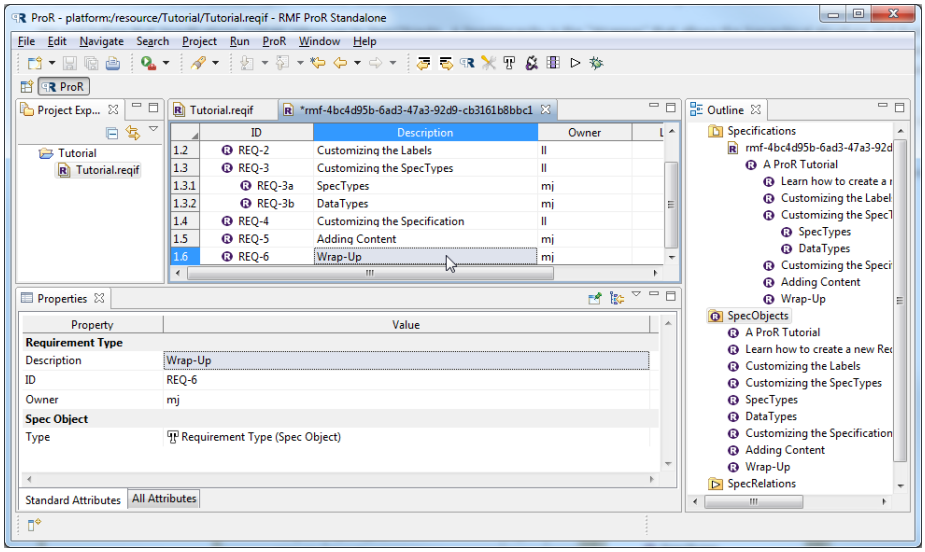

Figure 2.6: Improved Specification

2.2.6 Export Specification as HTML

If you want to export your Specification as HTML, go to “File” in the main menu and click “Print...”. The HTML representation is generated and automatically opened in your default browser.

2.2.7 Conclusion

Quite an achievement - but also quite a bit of work to get started. Also, entering information requires a lot of clicking, and some of the descriptions are not completely visible. In the next part of the tutorial, we will address these issues.

2.3 Tutorial 2: Use Presentations

We will continue where we left off with Tutorial 1 and we assume that you have ProR in front of you with a similar model.

In this tutorial we will introduce Presentations. Presentations are Eclipse Plug-Ins that extend the functionality of ProR. Specifically:

- Presentations can change the way how Attributes are rendered in the Specification
- Presentations can change the way how Attributes are edited in the Specification
- Presentations can perform tasks in the background.

ProR comes with a number of standard presentations that we will introduce in this tutorial.

2.3.1 Linewrap Presentation

By default, text is not wrapped in cells. We will enable the Linewrap Presentation for the Description column.

To do this, we select ProR > Presentation Configuration.

The dialog shown has no entries. The drop-down menu "Select Action..." at the top allows us to create new Presentations. We select the "Linewrap" Presentation. This will create a new entry in the upper pane. Upon selecting it, we can configure it in the lower pane.

A Linewrap Presentation has only one configuration element, the Datatype. We select "T_String32k". This means that from now on, all Attributes of this type will be rendered with the Linewrap Presentation. Upon completion, the dialog should look like this:

Upon closing the dialog, the lines that are too long should be wrapped automatically. Also, upon clicking on a cell, the content is now wrapped in the editor.

2.3.2 ID Presentation

It would be nice if every SpecObject had its own unique ID. Actually, it does (it is shown in the Property View, if a SpecObject is selected in the Outline View). But that ID is meant for machines and is not practical.

The ID Presentation allows the automatic creation of IDs. Let's create one.

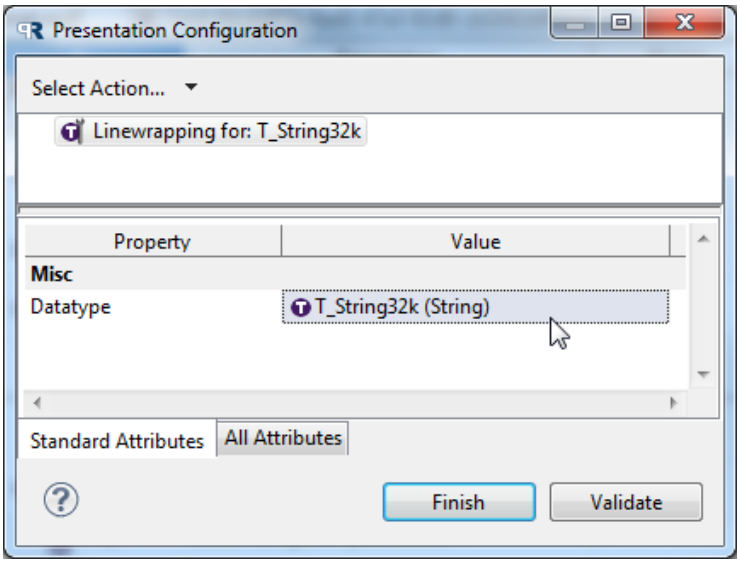

Figure 2.7: Presentation Configuration Dialog

Remember that Presentations are associated with Datatypes, not Attributes. Thus, we first have to create a new Datatype. We then associate that Datatype with the Attribute "ID". We described this process in the first tutorial. Here is a screen-shot of the configuration dialog, when all is done:

The last step is, like before, to associate that Datatype with the Presentation.

We open the Presentation Configuration and create a new Presentation from the dropdown menu "Select Action...", this time of type "Id" Presentation. We associate it with the newly created Datatype. After configuration, the dialog should look as follows:

Note that you could adjust the prefix and the Count.

NOTE: At this point, the Presentation doesn't check yet for duplicates. It simply grabs a new value from count, increments it and uses it. Also, existing values are not overwritten.

2.4 Tutorial 3: Advanced SpecTypes

So far, we have a model with only one SpecObjectType. In this tutorial, we will show how we can work with multiple SpecTypes, and we will introduce other SpecTypes.

2.4.1 Multiple SpecTypes

The first entry in our Specification ("A ProR Tutorial") isn't really a requirement. Thus, it doesn't need an ID or an owner, and it would be nice to Highlight

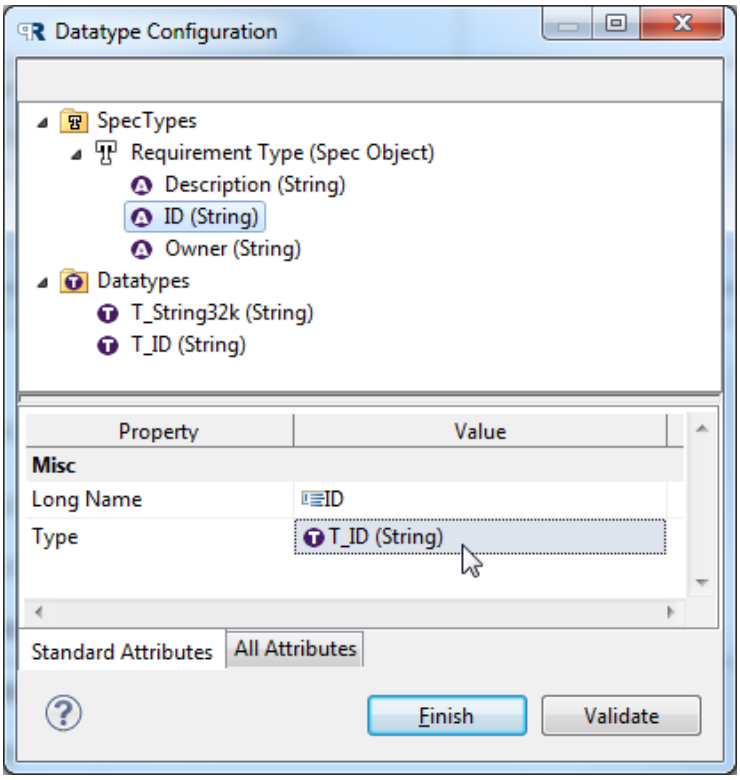

Figure 2.8: Datatype Configuration Dialog

it somehow. For Highlighting, we have the Headline Presentation. We will:

- Create a new "Headline" SpecObjectType
- Create a new Datatype that will be used for the Headline content
- Associate that Datatype with the Headline Presentation

With ProR > Datatype Configuration we get the Dialog where we can create new SpecTypes and Datatypes. For the first time, we create a new SpecObject-Type by right-clicking on "SpecTypes". One of the entries in the child menu is "SpecObject Type".

Once created, we select it and rename to "Headline Type" it in the properties.

Then we give it a new Attribute called "Description".

NOTE: It is important that we call it description. This matches the Attribute from "Requirement Type". By using the same name, we ensure that the Attributes appear in the same column, even though they are different Attributes.

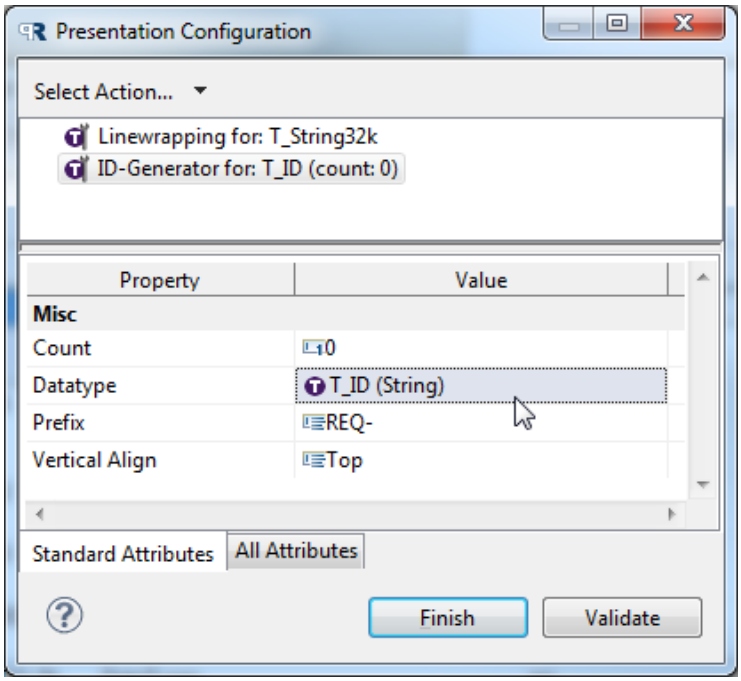

Figure 2.9: ID Configuration Detail

We do not set the type yet, as we need to create a new one. We do this by right-clicking on "Datatypes". There we create a new "Datatype Definition String" and call it "T_Headline". Now we can go back to the Description Attribute and set the type to T_Headline.

When all this is done, the configuration should look like this:

You can change the type of a SpecObject by selecting it an changing it in the Properties view. Please note that currently all existing values are lost when changing the type.

After the changes, the GUI should look as follows:

Note the following:

- The columns "ID" and "Owner" are now empty and cannot be edited for the Headline
- Note how the Property View changes, as you select SpecObjects of different types
- Right-clicking on a row now shows one more option for child/sibling creation: A new entry of type "Headline Type"

Last, we will use the Headline Presentation for the type T_Headline. This is done via the Presentation Configuration, and should result in the following. In addition, you can change the font size of the headline in the "Size" Attribute:

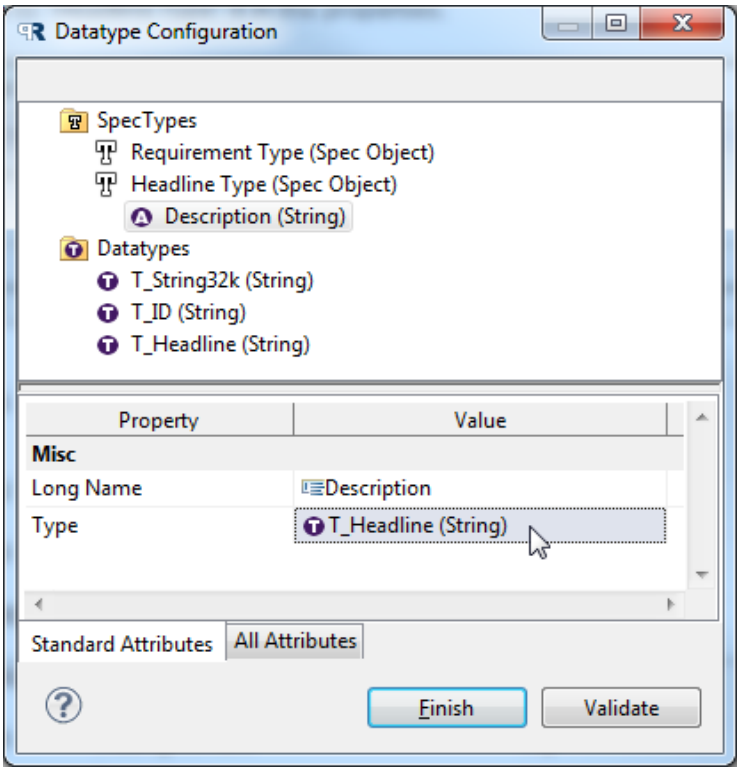

Figure 2.10: Datatype Configuration for the Headline Presentation

2.4.2 Other SpecTypes

You may have noticed in the Datatype Configuration Dialog, that right-clicking on "SpecTypes" offered more options, besides "Spec Object Type". A number of ReqIF-Elements can have Attributes.

We will now create a "Specification Type" and assign it to our Specification.

Try to create a "Specification Type" and configure it as shown in the screenshot:

Next, we will assign this type to the one Specification that we have. To do this we select the Specification in the Outline View. That will show the Specification's Properties in the Properties View. The "Type" Property is empty. We select "Specification Type" from the drop down. As soon as it is selected, the Attribute "Description" will appear in the properties view.

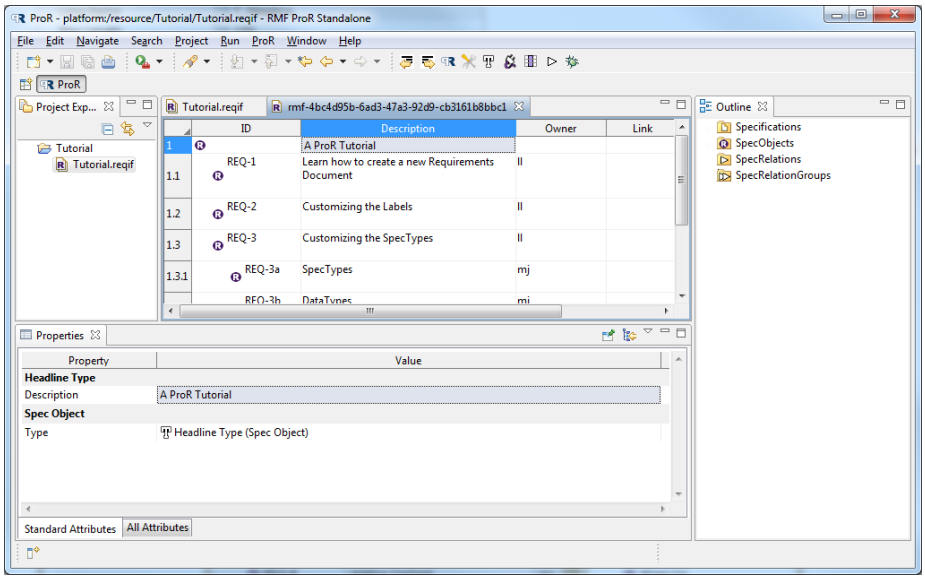

Figure 2.11: Activated Headline Configuration

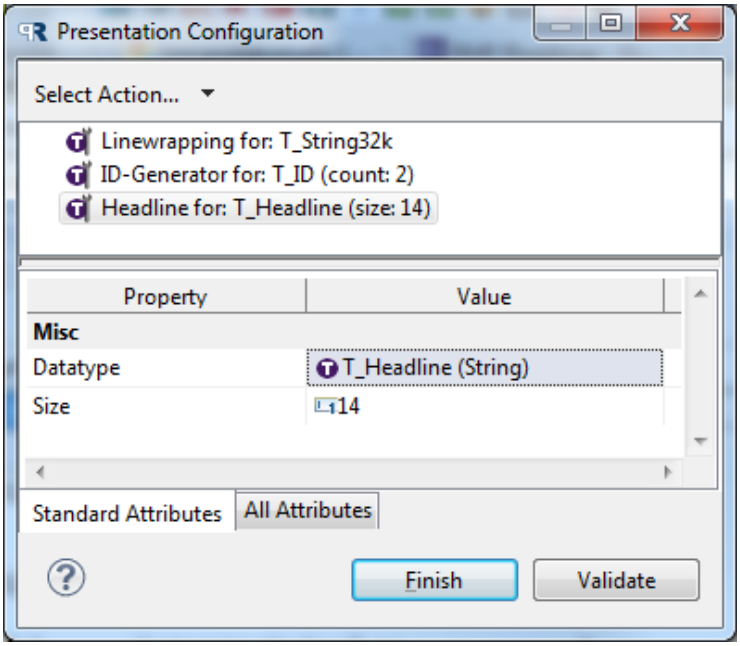

Figure 2.12: Presentation Configuration for Headline

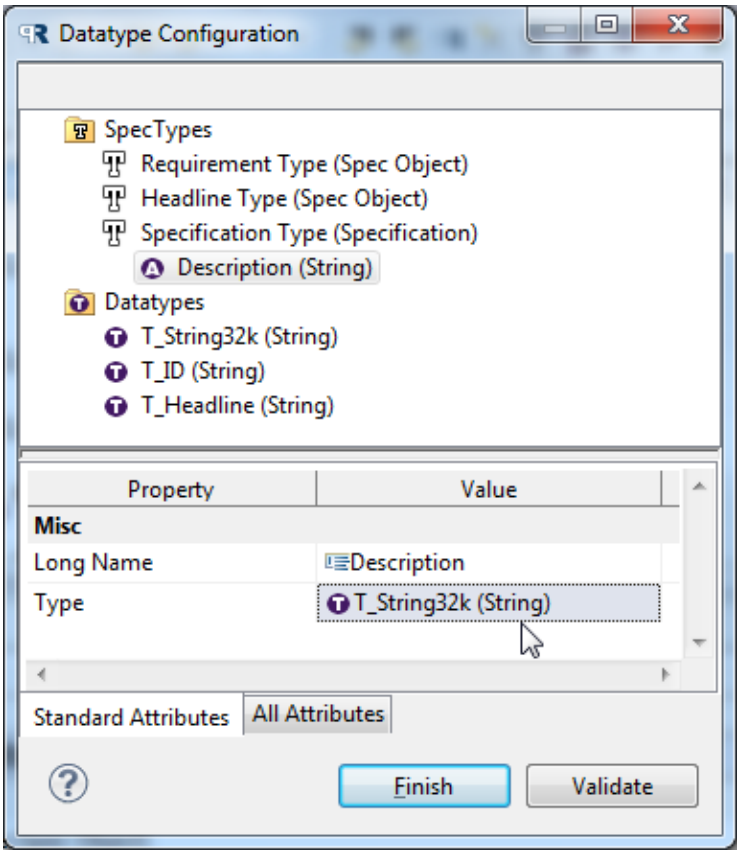

Figure 2.13: Creating a New SpecType

2.5 Tutorial 4: Links between SpecObjects (SpecRelations)

The implementation of SpecRelations is not complete yet. Still here are a few pointers to show what is planned.

2.5.1 Creating SpecRelations

SpecRelations are created by "Link-Dragging". This is platform specific:

- Linux: Dragging with Ctrl-Shift
- Mac: ???
- Windows: Dragging with Alt

Showing links as children can be switched on or off with the little triangle icon in the toolbar:

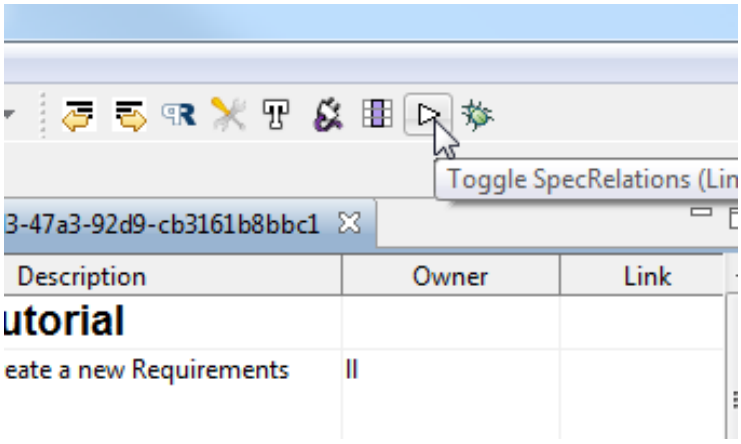

Figure 2.14: Toggle Links on and off

After creating a link:

- The "Link" column shows incoming and outgoing links
- The SpecRelations are shown as children in the Specification view

SpecRelations can have their own Types. Once a type is set, the corresponding columns in the Specification are rendered using Presentations, etc. However. the Type has to be set explicitly (a link created with dragging has currently no Type).

Here is a Screenshot of a Specification with some Links and a SpecRelation-Type that includes a "Description" Attribute:

Note the following:

- The right column shows incoming and outgoing links
- The number to the left of the triangle is incoming links, the other outgoing links
- The outgoing link from REQ-1 is shown
- The link from REQ-1 has a description, and it shows the destination name in the "Link" column

2.5.2 Outline View

Note that the Outline shows a folder with all SpecRelations.

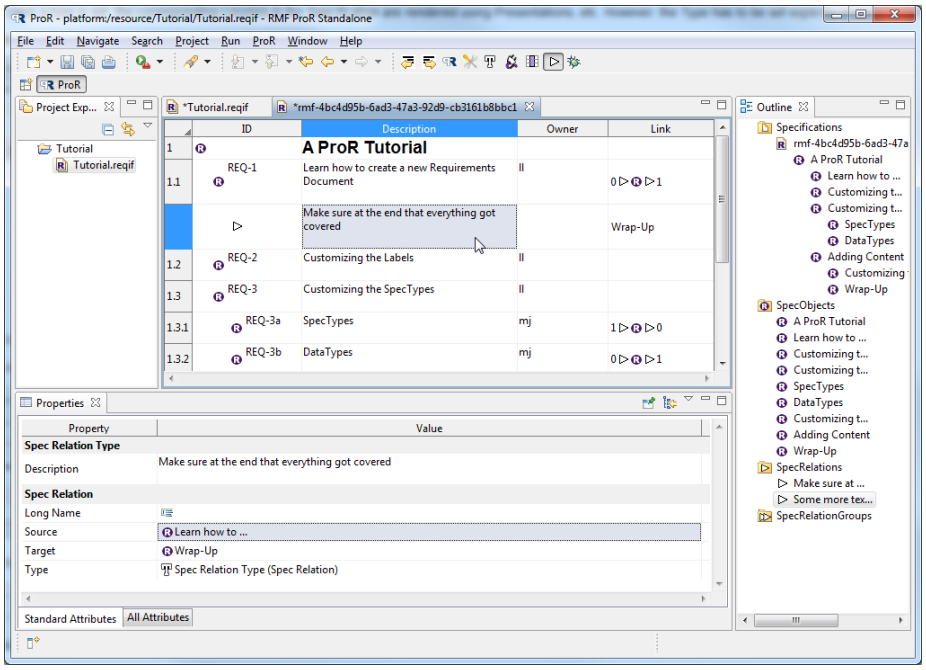

Figure 2.15: Showing Links in the GUI

Chapter 3

Reference

This is the reference manual for ProR.

3.1 Editors

Upon opening a ReqIF Model, the ReqIF Editor opens that provides an overview of the model. A model contains any number of specifications, and each specification is opened in its own editor.

3.1.1 ReqIF Overview Editor

3.1.2 Specification Editor

3.2 Views

By default, ProR shows three views.

3.2.1 Project View

3.2.2 Properties View

3.2.3 Outline View

3.3 Configurations

The ProR menu contains entries to launch a number of configuration dialogs.

3.3.1 General Configuration

3.3.2 Datatype Configuration

This configuration is opened via ProR | Datatype Configuration ...

The dialog shows two folders, one for SpecTypes and one for Datatypes. SpecTypes are created for typing elements that have attributes (SpecObjects, Specifications, SpecRelations). New SpecTypes can be created by right-clicking on the folder and selecting “New Child”. Through the same mechanism, attribute definitions can be added to a SpecType. attribute definitions are typed. Selecting an element shows its properties in the lower pane, where it can be configured.

Attribute definitions must have a name and a datatype. Some attribute definitions allow further customization. The datatype is selected from a dropdown. New datatypes can be created by right-clicking on the folder “Datatypes” and selecting “New Child”. Again, selecting a datatype shows its properties in the lower pane, where it can be configured. A datatype should have at least a long name.

As an example, consider the following Datatype Configuration Dialog:

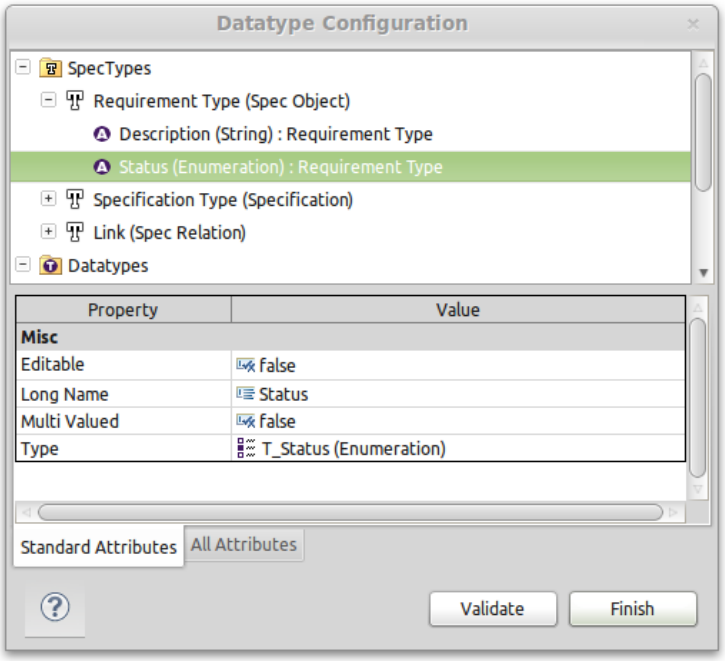

Figure 3.1: Datatype Configuration Dialog

The Spec Type for “Requirements Type”, which is applicable to SpecObjects, is expanded. The Spec Type has two attributes, “Description” (String) and “Status” (Enumeration). Status is selected, and in the pane below the mandatory values, “Long Name” and “Type” have been set. Further customization of the attribute is possible, e.g. by converting it in a Multi-Valued attribute by setting the corresponding flag to “true”.

Enumeration Datatypes

An enumeration datatype must have enumeration values. These are created by right-clicking the enumeration datatype and selecting New Child | Enum Value. You may have to unfold the enum value to select it, so that you can provide it with a Long Name. The following shows a correctly configured enumeration datatype:

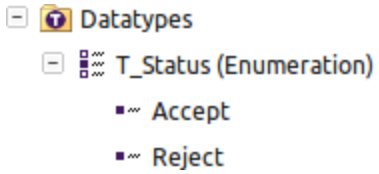

Figure 3.2: Enumerations

3.3.3 Presentation Configuration

3.3.4 Column Configuration

This configuration is specific to the Specification Editor.

The Column Configuration Dialog configures the Columns of a Specification. Columns are identified by name. The width of the column can be adjusted directly by dragging the column separator in the table header.

If the SpecObject has an attribute where the name of the attribute matches the name of the column, then that attribute is shown in that column.

Chapter 4

Presentations

ProR has an extension mechanism that allows custom rendering for certain attributes of a SpecObject. The following Presentations have been implemented:

4.1 ID Generator Presentation

This presentation allows the designation of Attributes to be used as human-readable IDs. The user defines a prefix, and the presentation manages a counter. When a new SpecObject with a designated Attribute is created, then it will get an ID, consisting of prefix and number.

TODO: The plugin doesn't check whether an ID already exists - it simply takes the next number and increments the counter. We should (1) Set the counter correctly upon opening a ReqIF and (2) hook in a Validator.

4.2 Linewrap Presentation

This presentation automatically renders long text with linebreaks and allows editing with linebreaks.

4.3 Headline Presentation

This presentation renders an attribute in bigger font and bold. It allows setting the font size.

Chapter 5

Glossary

5.1 ReqIF

ReqIF stands for Requirements Interchange Format. It is an exchange format for requirements and a data model. ReqIF is the successor of RIF and an OMG standard.

RMF supports ReqIF 1.0.1 (official OMG standard).

The ProR GUI does support ReqIF.

5.1.1 History

For technical and organizational reasons, two companies in the manufacturing industry are rarely able to work on the same requirements repository and sometimes do not work with the same requirements authoring tools. A generic, non-proprietary format for requirements information is required to cross the chasm and to satisfy the urgent industry need for exchanging requirement information between different companies without losing the advantage of requirements management at the organizations' borders.

The Requirements Interchange Format (ReqIF) described in this RFC defines such a tool-independent exchange format. Requirement information is exchanged by transferring XML documents that comply to the ReqIF format.

In 2004, the HIS (Hersteller Initiative Software), a panel of Germany's automotive manufacturers (Daimler, VW, Porsche, Audi and BMW Group) developed the idea of creating the "Requirements Interchange Format". In 2005, the first version of that format was presented at the REConf®, a conference about requirements engineering and management, in Munich. In 2008, the HIS Steering Committee decided that the internationalization and maintenance of the Requirements Interchange Format should be proceeded with the ProSTEP iViP Association. A project was set up and a team was built that includes members of the ProSTEP iViP Association, representatives of manufacturing companies (Audi, BMW Group, Daimler, VW, Bosch and Continental), tool vendors (Atego, IBM, MKS) and development partners (HOOD GmbH, PROSTEP AG).

The ReqIF team expects that making the Requirements Interchange Format an OMG standard increases the number of interoperable exchange tool implementations on the market, fosters the trust of companies exchanging requirement information in the exchange format and provides safety of investments to tool vendors.

5.1.2 Previous Versions of ReqIF

This document is submitted as RFC of the Requirements Interchange Format (ReqIF) to the OMG. Before the submission, the Requirements Interchange Format has been a specification proposed by the HIS and in its latest version, a recommendation of ProSTEP iViP. For these versions, the abbreviation "RIF" has been applied. The HIS released the Requirements Interchange Format as RIF 1.0, RIF1.0a, RIF 1.1; RIF1.1a and the ProSTEP iViP released the recommendation RIF 1.2.

As the acronym RIF has an ambiguous meaning within the OMG, the acronym ReqIF has been introduced to separate it from the W3C's Rule Interchange Format. ReqIF 1.0 is the direct successor of the ProSTEP iViP recommendation RIF 1.2.

5.2 RIF

RIF stands for Requirements Interchange Format. It is an exchange format for requirements and a data model. RIF is the predecessor of ReqIF.

RMF supports the following versions of RIF:

- RIF 1.1a
- RIF 1.2

The ProR GUI does currently not support RIF.