ProR Rodin/Event-B Integration

Content

User Story

At the beginning the user starts the ProR tool. His first task is to create a new project and a corresponding ReqIF file in order to create new requirements. Typically, a rough set of requirements R1..Rn already exists, which has to be transferred to ProR. In practice this is often a mixture of requirements, domain properties and even specification elements. We refer to these as artifacts.

Next, a subset of the artifacts is modelled in Rodin. The user creates a new Event-B component. He starts with a suited artifact of interest Ri and assigns it the type “traced artifact”. He then models the domain properties that are necessary to express Ri, and proceeds to model Ri itself. This may result in a revision of Ri, as well as the addition of new artifacts. He continues with other suited artifacts and restructures artifacts and models as he sees fit. During this process, he creates a traceability between artifacts and their corresponding model elements. This is done via drag and drop, and results in SpecRelations. Source and target of the link are marked as “validated”.

As the user continues to model, traces that point to or from elements that are changed will be marked as “invalidated”. This means that the ProR tool will inform the user that he has to re-verify the corresponding traces. We define verification as a process for assuring that the model elements satisfy the requirements.

Names of model elements are picked up by the tool and allow marking and color highlighting in the requirements text.

The user will eventually have a set of artifacts and a model that are connected by traces, and that have the following properties:

- All “traced artifact”s have at least one outgoing trace
- All traces are marked as “validated”

The user runs an analysis tool that reports all violations of these properties.

Glossary

Artifact	A requirement, domain property or other piece of information, represented as a SpecObject.
Model Element	Any Event-B element. A model element may have sub-model elements, e.g. both an event and a guard are both model elements.
Model Element Name	The model element name is the identifier label of a model element. For instance, the identifier of a variable is the variable name. In case of an invariant the identifier is the label directly to the left of the invariant (a typical identifier of an invariant is “inv01”).
Proxy Model Element	A SpecObject that represents a model element.
Link, Trace	Synonyms for a SpecRelations object.

Architecture Outline

This section outlines the architecture of the Rodin Integration.

Idea: Predefined Event-B specific ReqIF types and presentation configurations. The user can then create a new ReqIF file with these types and presentation configurations already predefined.

The Rodin Integration is subdivided into four sub projects. Three sub projects are implemented as presentation plugins for ProR. The fourth sub project comes in the form of a wizard which creates a ReqIF model file with the predefined Event-B specific ReqIF types. The following sections outline these sub project as well describe the needed Event-B specific ReqIF types.

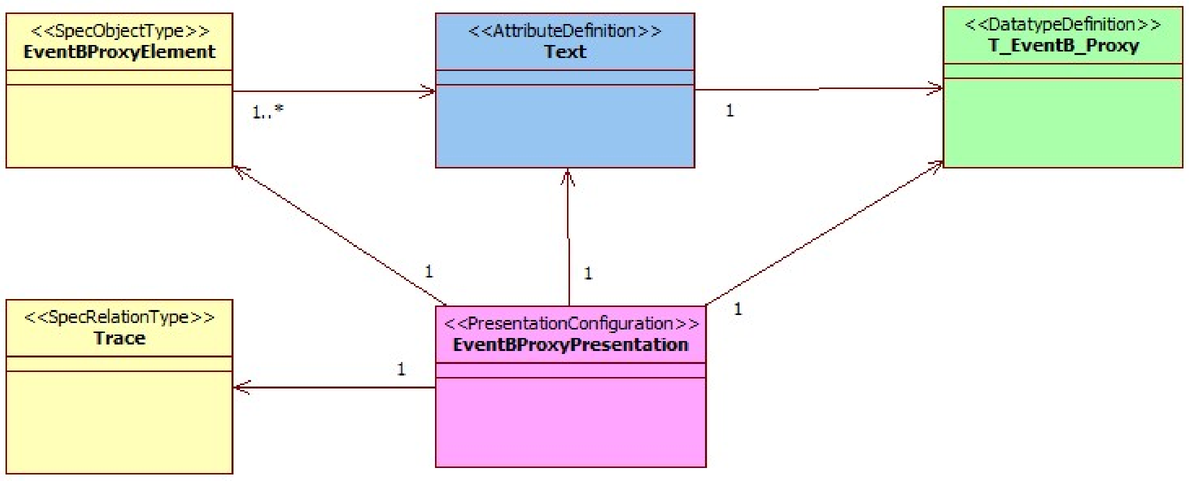
Drag & Drop + Model Element Cell Renderer

Name: EventB_Proxy_Presentation (PresentationConfiguration)

Description: This presentation plugin is responsible for the drag and drop functionality of model elements from the Event-B editor to the ProR specification editor. More precisely: ProR locates the proxy model element for the model element, or creates one if it does not exist yet. ProR creates a trace/link between the proxy model element and the artifact. Furthermore the presentation plugin is responsible to render the cell containing the proxy model element (i.e. showing the variable name).

The following Event-B specific ReqIf types are needed:

- **Trace (SpecRelationType):** Represents the trace between two artifacts (i.e. between a requirement and a model element).
- **Event-B Proxy Model Element (SpecObjectType):** Represents an Event-B model element as an "proxy" spec object.
- **Description (AttributeDefinition):** The description attribute contains the id of the Event-B model element and the source file (machine or context). Furthermore, the description attribute contains the model element in plain text. This is useful to synchronize with Event-B and to detect changes whenever the specification editor is closed.
- **T_EventB_Provy (DatatypeDefinition/String):** This DataType in connection with the *EventB_Proxy_Presentation* presentation configuration is responsible for rendering the the id + source and for displaying the model element (i.e. an invariant).



Use Cases

Prefix: UC-DND
Actors: User, ProR

Creating a trace

UC-DND-CREATE-TRACE	Creating a trace
Primary Actor	User, ProR
Precondition	At least one artifact and one model element exists.
Guarantee	A proxy model element is created (if does not exist yet) and a trace between the artifact and the model element is established.
Main Success Scenario	1. The user selects the model element in the outline view of the Event-B editor. 2. The user drags the model element (target) and drops it to the corresponding artifact (source). 3. ProR locates the proxy model element for the model element, or creates one if it does not exist yet. 4. ProR creates a trace/link between the proxy model element and the artifact.
Variation	-
Exceptions	Trace already exists. In this case nothing happens.

Editing a model element

UC-DND-EDIT-MODEL-ELEMENT	Editing a model element
Primary Actor	User, ProR
Precondition	1. A corresponding proxy model element of the edited model element exists. 2. ProR is open.
Guarantee	The proxy model element is updated.
Main Success Scenario	1. The user edits a model element (i.e. an invariant). 2. ProR updates the corresponding proxy model element upon save of the Event-B editor.
Variation	-
Exceptions	Note that ProR can only react to model changes if the ReqIF model is open, while the Event-B model is being edited. See: UC-DND-OPEN-REQIF-MODEL

Deleting a model element

UC-DND-DELETE-	Deleting a model element

MODEL-FILE	
Primary Actor	User, ProR
Precondition	ProR is open
Guarantee	-
Main Success Scenario	1. The user deletes a model element. 2. ProR marks the proxy model element as “missing”.
Variation	The user deletes a model file (i.e. machine or context file): In this case all linked model elements of the model file are marked as “missing”.
Exceptions	Note that ProR can only react to model changes if the ReqIF model is open, while the Event-B model is being edited. See: UC-DND-OPEN-REQIF-MODEL

Renaming a model file

UC-RENAME-MODEL-FILE	Renaming a model file
Primary Actor	User, ProR
Precondition	-
Guarantee	-
Main Success Scenario	1. The user renames a model file (i.e. machine or context file). 2. ProR updates the references of the proxy model elements.
Variation	-
Exceptions	Note that ProR can only react to model changes if the ReqIF model is open, while the Event-B model is being edited. See: UC-DND-OPEN-REQIF-MODEL

Opening a ReqIF model

UC-DND-OPEN-REQIF-MODEL	Opening a ReqIF model
Primary Actor	User, ProR
Precondition	The model files exists under the same names as referenced in the proxy model elements and the ProR editor is closed.
Guarantee	Changes in the model have been detected and acted upon.

Main Success Scenario	1. The user opens a ReqIF model. 2. ProR detects changes made in the model. 3. ProR updates the affected proxy model elements (See also: UC-DND-EDIT-MODEL-ELEMENT and UC-DND-DELETE-MODEL-FILE). 4. ProR marks the proxy model elements which are not found in the Event-B model as “missing”.
Variation	-
Exceptions	-

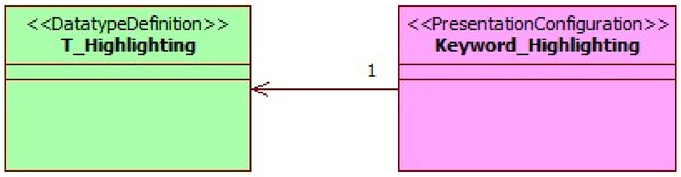
Color Highlighting

Name: Highlighting (PresentationConfiguration)

Description: This presentation plugin is responsible for highlighting predefined key words (i.e. model element names).

The following ReqIF types are needed:

- **T_Highlighting (DatatypeDefinition/String):** This DataType in connection with the *Highlighting* presentation configuration is responsible for highlighting predefined key words (i.e. model element names).



Use Cases

Prefix: UC-COLOR
Actors: User, ProR

Using model element names in artifact attributes

UC-COLOR-HIGHLIGHTING	Using model element names in artifacts attributes
Primary Actor	User, ProR
Precondition	-
Guarantee	The model element names in the artifact attribute (i.e. description) are highlighted.

Main Success Scenario	1. The user enters the name of a model element (in square brackets) in an artifact attribute (i.e. description). 2. ProR highlights the model element name in the artifact attribute (i.e. description).

Scenario	**Renderer**	**Editor**
name not marked	black text	red underline
marked & exists	blue text	blue text
marked & does not exist	red text	red text

Variation	
Exceptions	Note that ProR can only react to artifact changes which are located outside of the ReqIF model, if the ReqIF model is open.

Renaming a model element

UC-COLOR-RENAME-MODEL-ELEMENT	Renaming a model element
Primary Actor	User, ProR
Precondition	-
Guarantee	The highlighted model element names are updated in the artifact attributes (i.e. description).
Main Success Scenario	1. The user renames a model element in the Event-B editor (i.e. a variable). 2. ProR updates the highlighted model element names in the artifact attributes (i.e. description) upon save of the Event-B editor.
Variation	-
Exceptions	- Note that ProR can only react to artifact changes which are located outside of the ReqIF model, if the ReqIF model is open. - We get a problem if a variable and constant have the same name.

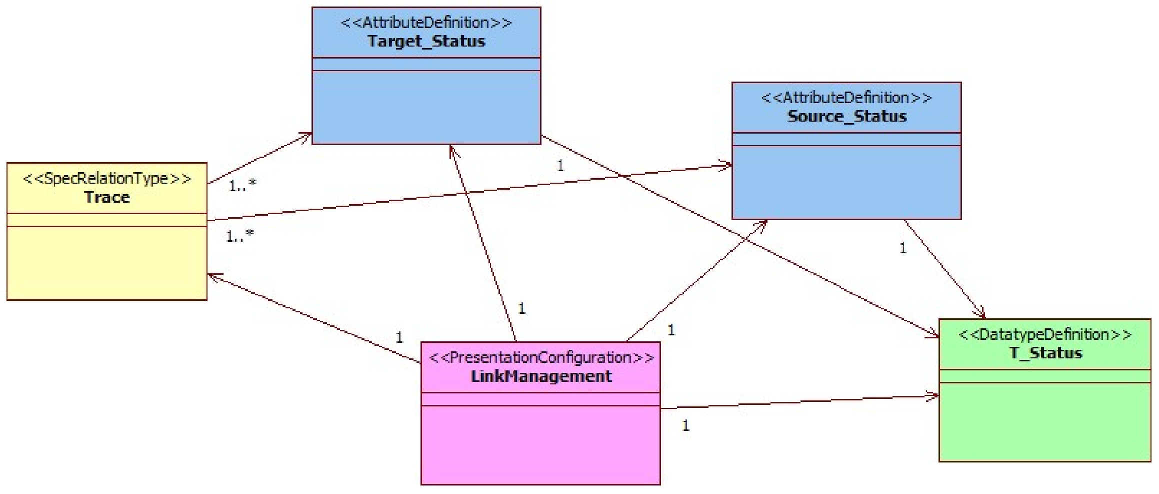
Link Management

Name: LinkManagement

Description: This presentation plugin is responsible for managing the traces between artifacts (i.e. between a requirement and a model element). For instance it manages the status attribute of the trace by setting the right status.

The following ReqIF types are needed:

- **Trace (SpecRelationType):** Represents the trace between two artifacts (i.e. between a requirement and a model element).
- **SourceStatus (AttributeDefinition):** Represents the attribute for the source status with the Status DatypeDefinition (see below).
- **TargetStatus (AttributeDefinition):** Represents the attribute for the target status with the Status DatypeDefinition (see below).
- **T_Status (DatatypeDefinition/Boolean):** The Status attribute is either true or false. For instance true could mean “changed” (Is set whenever a model element or the requirement respectively) and false “not changed” (Means that the model element or requirement is in a “clean” status, where nothing was changed, i.e. after the first creation of the trace).



Use Cases

Prefix: UC-LINK
Actors: User, ProR

Editing an artifact with trace(s)

UC-LINK-EDIT-REQ	Editing an artifact with trace(s)
Primary Actor	User, ProR
Precondition	At least one trace between two artifacts exists.
Guarantee	The source or target status attribute of the corresponding traces are set to “changed”.
Main Success Scenario	1. The user edits an artifact (i.e. a requirement). 2. ProR sets the source or target status attribute of the trace(s) to “changed”.

Variation	If the status attribute of the trace(s) are already set to “changed”, then this use case has no effect.
Exceptions	Note that ProR can only react to artifact changes which are located outside of the ReqIF model, if the ReqIF model is open.

Event-B ReqIF File Wizard (UC-WIZ)

Project Setup

UC-WIZ-PROJECT-SETUP	Project Setup
Primary Actor	User, ProR
Precondition	-
Guarantee	A ReqIF model is created with the predefined Event-B specific ReqIF types and presentation configurations.
Main Success Scenario	1. The user opens the Event-B ReqIF File wizard. 2. The user enters a name for the ReqIF file. 3. The user confirms his entries. 4. ProR creates a new ReqIF file with the predefined Event-B specific ReqIF types and presentation configurations. 5. ProR opens the created ReqIF model.
Variation	-
Exceptions	-

Out of Scope

We should define on which changes this presentation plugin should react. One idea is to define a list of source and target datatypes. I.e.:

- **T_SourceDatatype (DatatypeDefinition):** The LinkManagement presentation configuration listens on changes made in attributes which have this datatype and sets the correct status of the source.
- **T_TargetDatatype (DatatypeDefinition):** The LinkManagement presentation configuration listens on changes made in attributes which have this datatype and sets the correct status of the target.

However, we decided that this feature should not be part of the first iteration. Furthermore, this feature needs more discussion.

After editing a Event-B model element (i.e. the name of the variable) ProR changes automatically all marked keywords in the requirements text. However, this leads to a “change” of the source or target respectively. Should we handle this really as change?

Event-B ReqIF File Wizard

The Event-B ReqIF File Wizard sub project is the “aggregating project”. It is responsible to create a ReqIF file with the predefined Event-B specific ReqIF types (see last sections) and to provide the user a “start-to-work“ specification editor.

In the first version it simply creates this file under a specified name. However, for further development it could be possible to provide more features for customizing the specification.

Further Development

This sections lists some ideas for further development.

- Xtext could be used to provide a DSL which is linked to a model. So, the user can use the code completion for adding model element names in the requirements text.
- Mylyn like features where the user clicks on a requirement to be modelled and mark this requirement as “is modelled now”
- The Highlighting Presentation Configuration Plugin could provide an extension point which defines how to provide a list of key words. This is interesting for other formal languages like Classical-B.