The tree-based Henshin Transformation Model Editor

Gregor Bonifer

The tree-based Henshin Transformation Model Editor

by Gregor Bonifer

Table of Contents

List of Figures

Chapter 1. Overview

This part of the documentation covers the tree-based Editor for Henshin Transformation Models.

The editor is a customization of the default editor generated by EMF. It follows the free-editing paradigm that is commonly expected from EMF editors. To alleviate creating, editing and reading transformation models, it brings a variety of display customizations, like context-sensitive icons, selection-based coloring of semantically associated elements and dynamic filtering. The default editing actions are complemented by a suite of advanced editing actions leveraging the EMF Command Framework. To facilitate the testing of transformations during the development, a wizard allows to invoke transformations to preview their effect using the EMF Compare services and visualization. Syntactic analysis may be performed by invoking the EMF Model Validation on selected elements.

Chapter 2. Advanced Editing

The editor provides a variety of advanced editing commands. Based on the currently selected model-elements, the context-menu offers all applicable commands in a submenu named *Advanced Editing* . The following sections describe each command in detail, for which selection it will be available and how it alters the transformation model.

2. Create Mapped Node

Selection: Single Rule

Effect: Creates two nodes in the selected rule. One in the LHS and one in the RHS. Creates a mapping from the LHS-node to the RHS-node.

3. Create Edge

Selection: Two Nodes in same graph.

Effect: The menu will provide an actions for every known reference between the types of the selected nodes. If one is chosen, an edge will be created with the selected type. The edges source and target will reference the two selected nodes.

The command checks for multiplicity and uniqueness of an edge. Therefore there may not be two equally-typed(and equally directed) edges between the two nodes. If a reference is not of a collection type(i.e. upperBound<2) there must not be an outgoing edge of the same type from the node whose class contains the reference.

Edges that may not be created for reasons of multiplicity or uniqueness violations are shown as disabled menu items.

4. Remove Origin/Image from ..

Selection: Single Node in a rule's LHS or RHS. A LHS node must be mapped to a RHS node. A RHS node must be mapped from a LHS node.

Effect: Removes the corresponding other node from the model.

5. Copy to ..

Selection: Multiple nodes contained in the same graph.

Effect: Establishes a complete image of the selected subgraph structure in the target graph. The subgraph is defined by all selected nodes and all edges between them. If a node of the subgraph does not correspond to node in the target graph, the command adds a copy of the node to the target graph and creates a mapping between the two. If a node already corresponds to a node in the target graph, it is skipped. The same is done for all edges of the selected subgraph. This command can be used to create or to incrementally update a certain target graph.

6. Define as Nested Condition

Selection: Multiple nodes contained in the same LHS or NestedCondition graph.

Effect: This creates a new NestedCondition inside the source graph and copies the selected subgraph to the NestedCondition's conclusion graph. In case there already is a formula present in the source graph, the menu provides two alternatives regarding the handling of the existing formula:

- **Replace current Formula** removes the existing formula and inserts the newly created NestedCondition in its place.
- **Wrap with current Formula** creates an And which holds the existing formula and the newly created NestedCondition as child elements.

7. Formula Commands

7.1. Negate Formula

Selection: Single Formula

Effect: Negates the formula by either inserting or removing a containing Not element. If a Not is negated it is replaced by its child element in its containing element.

7.2. Toggle Junction Type

Selection: Single BinaryFormula

Effect: Replace an And formula with an Or formula and vice versa. The child elements of the original formula are moved to the new formula.

7.3. Wrap in AND/OR

Selection: Single Formula

Effect: Replaces a formula by a newly created And or Or formula. The selected formula is added as child.

7.4. Replace by child Left/Right

Selection: Single BinaryFormula

Effect: Replaces a formula by its left or right child in its container. The selected formula and its other child are removed from the model.

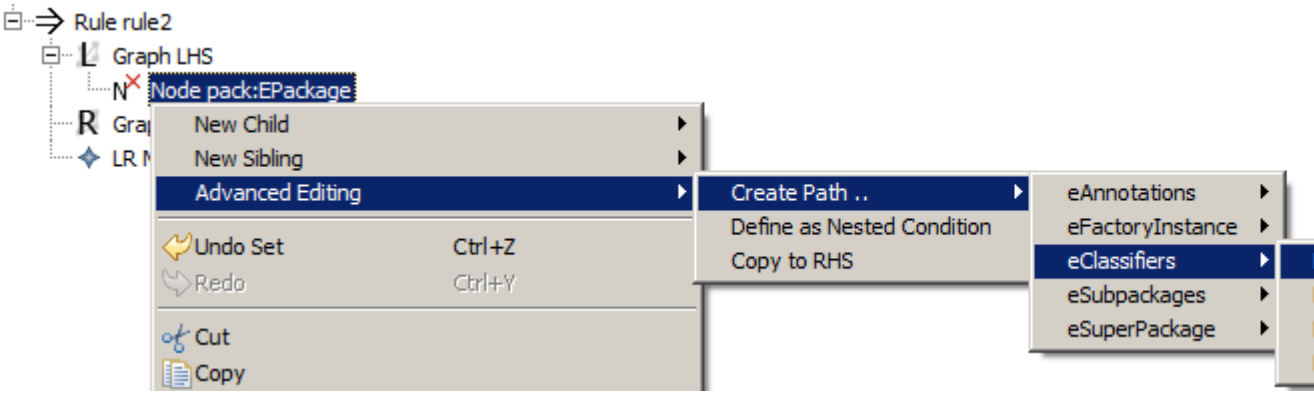
8. Create Path

Selection: Single Graph or single Node

Effect: Allows to create a graph path of arbitrary length by navigating the classes and references of the imported packages. This is done by unfolding a dynamic menu structure on demand. There are several different menu types: The **Reference Selection** offers all known references owned by the selected type. If the menu is shown for a node, this is done for the node's type. The **Subclass Selection** shows all classes that are compatible with the type of a selected reference. If the graph contains nodes whose type is one of those classes the menu will contain a section to **connect** the path to an existing node. If either a subclass or an existing node are selected the path may be created by selecting "Create" or "Connect". Also the path may be continued by selecting the corresponding ".. and expand" item. If the menu is opened for a graph an initial element has to be selected first. For this purpose there is an additional menu that allows selecting

an imported package and one of its contained classes. The expansion of the path can now be done in the same manner as if the menu was initialized for an existing node.

Figure 2.1. Create Path



Chapter 3. Display Options

1. Color Associated Elements

There is a lot of information that requires to be inferred from the usage context of a model element. E.g. a node contained in a rule's LHS may be preserved or might be deleted. This depends on the existence of a mapping from the node to a node in the RHS. Inferred information is presented by the different icons. This way the inferred information can be presented in concise manner that does not produce visual noise.

Knowing an element's semantics is quite helpful for reading and understanding a transformation. For editing a transformation this is not sufficient. One usually requires not only the information on how an element is affected but which associated elements have an impact on the element's semantics. Therefore the editor provides an option to *color associated model elements* (Figure 3.1, “Color associated Model Elements”) .

Figure 3.1. Color associated Model Elements

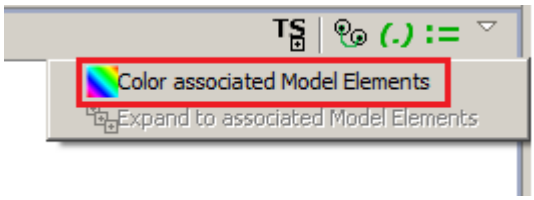
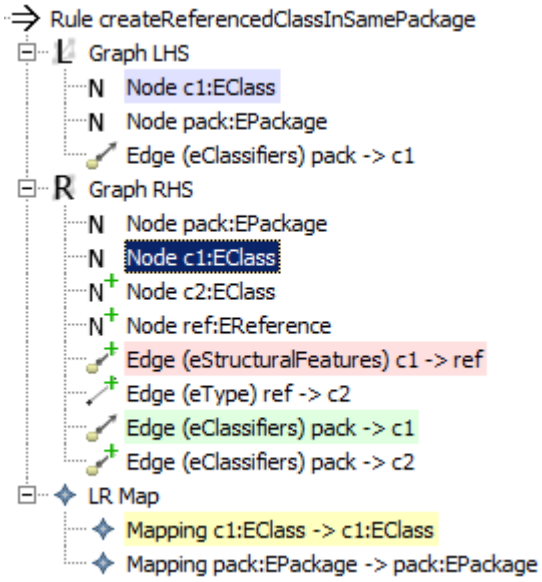


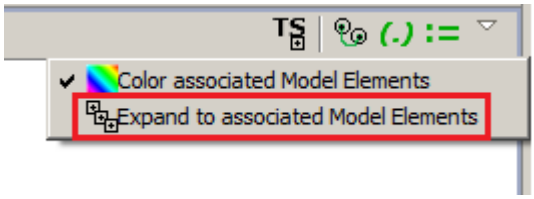
Figure 3.2, “Colored Model Elements” shows the effect of this option. The selected Node has one outgoing reference(marked red) and one incoming reference(marked green). As reflected by the icon, the node is preserved and therefore is the image of a mapping(marked yellow). The corresponding node in the LHS is colored blue.

Figure 3.2. Colored Model Elements



Optionally the tree can be expanded automatically to reveal those elements.

Figure 3.3. Expand to associated Model Elements



2. Filtering

Creating bigger and more complex transformations usually means creating a bunch of mappings, parameters and parameterMappings. Those tend to clutter the tree and often make it much less comprehensible. To reduce some of the noise, these elements may be excluded from the visualization. They remain part of the model and may be shown on demand by toggling the respective filter option.

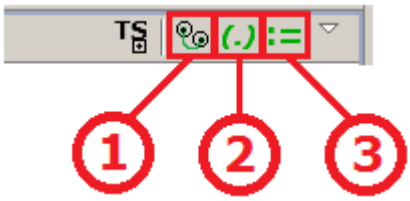
Figure 3.4, “Filter Model Elements” shows the filter buttons. Toggle filtering of..

1. Mappings

2. Parameters

3. ParameterMappings

Figure 3.4. Filter Model Elements

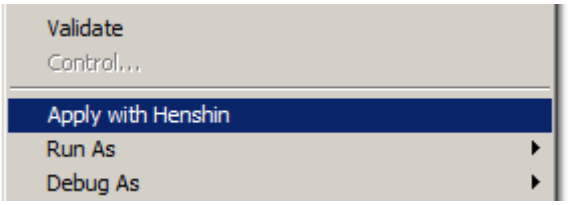


Chapter 4. Transformation Wizard

1. Opening the wizard

The transformation wizard provides a UI for applying and testing transformations. An action for opening the wizard is provided in the context of Rules, TransformationUnits, TransformationSystems and henshin-files. Inside the henshin-editor the popup menu provides an action "Apply with Henshin" as shown in Figure 4.1, “Wizard menu entry” . The popup menu of the Package Explorer lists the action in the Henshin submenu.

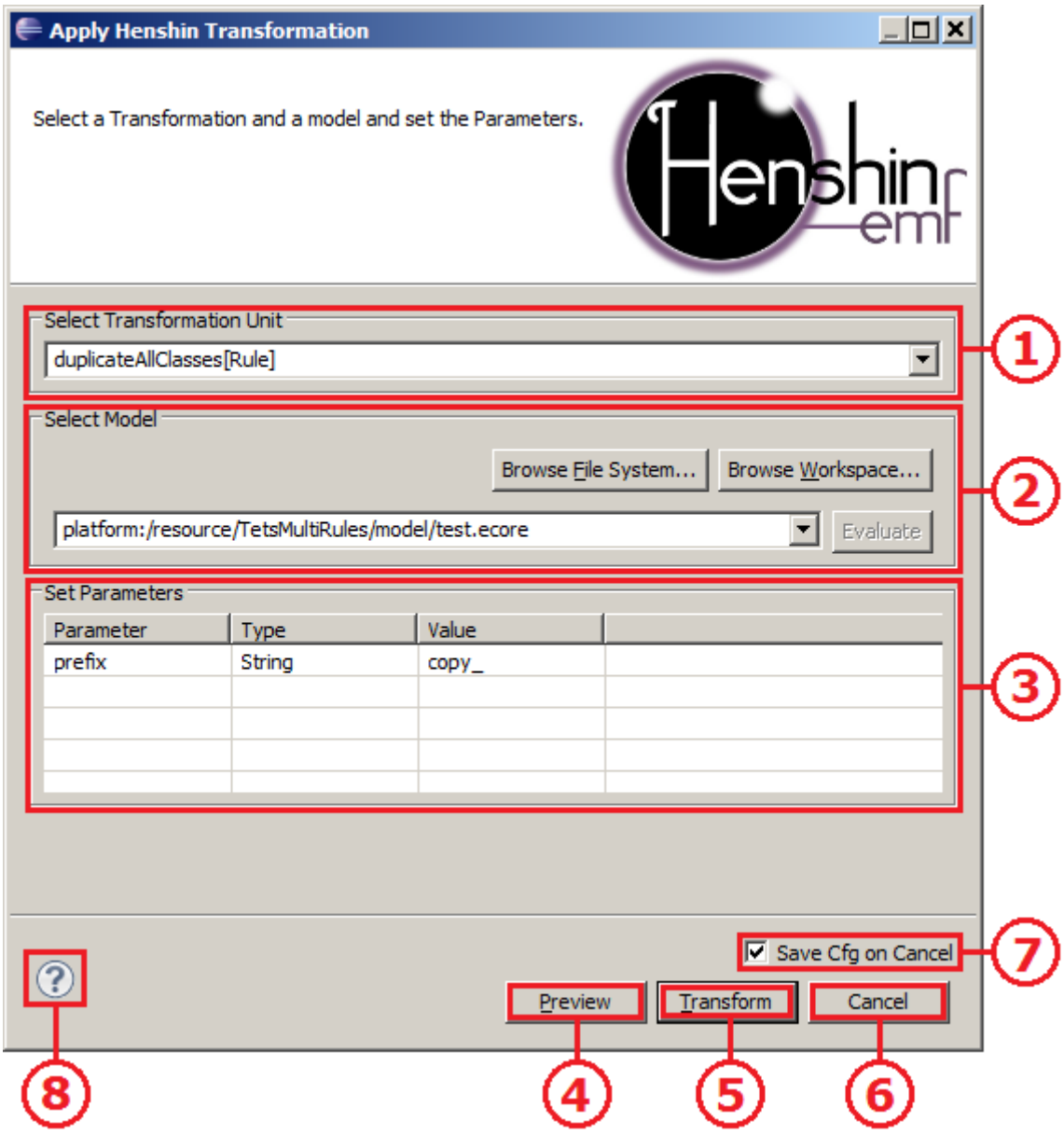
Figure 4.1. Wizard menu entry



2. Wizard Elements

The wizard's interface is made of a single page containing several sections that allow to configure the different aspects of a transformation's application. The following illustrates the UI elements in detail.

Figure 4.2. Wizard

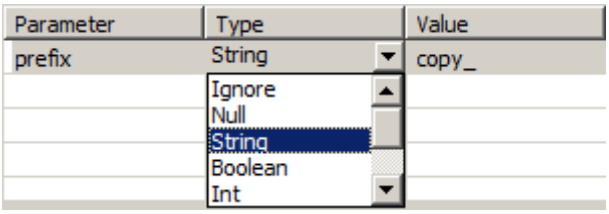


1. Select the Rule or TransformationUnit that shall be executed. This selector is only enabled if the wizard is opened in the context of a TransforamtionSystem or a henshin-file. Otherwise the Rule or TransforamtionUnit for which the wizard was opened will be fixed.

2. Select the resource you wish to transform. Only the first content element of the resource will be used as the graph.The element is inserted as root. This means that the graph will consist of the element itself and all transitively contained objects.

3. Set values for the parameters of the chosen unit. Since all values are entered as strings, it is necessary to also select a type. The given string-value is parsed as the defined type, which is then passed to the transformation.

Figure 4.3. Selecting a parameter's type



There are two types which do not allow a value to be entered:

- The type *Ignore* prevent a parameter from being initialized. This option must be used for parameters that pre-match a node in the LHS.

- The type *Null* initializes a parameter with null. If the parameter is used as an attribute value in the LHS, the attribute value of a match must be null. In case the parameter defines a prematch-node, setting the Null-type will prevent the unit from being applicable.

4. This allows to see a preview of the transformation. The transformation will be applied to a copy of the selected model. The differences are detected and visualized using EMF Compare.

5. Applies the transformation to the selected model and writes the result back to the resource. This action is encapsulated as an undoable operation using the Eclipse Command Framework. Note that this does not place an undoable command on the EMF CommandStack inside the editor. To undo a transformation you have to focus the Package Explorer. This will switch the context of the edit-menu to the platform's operation history.

6. Cancel closes the dialog without applying the transformation.

7. This flag indicates whether the settings shall be saved even if the transformation was not applied. The current settings are automatically saved when the dialog is closed via the *Transform* button. If you are in the process of developing a transformation, usually you will only preview your transformation's effects without applying them to the selected model. To avoid having to select a model and entering parameter values each time, those settings may are saved if the *Save Cfg on Cancel* is checked, even if the dialog is canceled.

8. Provides access to *this document* .

3. Transformation Preview

EMF Compare is used to preview a transformation's effect on a given model. As shown in Figure 4.4, “Wizard Preview” , a transformed copy is compared to the original model. To inspect the changes of the parameter values, the preview window provides a second page, that lists the parameters of the applied unit together with their initial values and their values after the transformation was applied. Figure 4.5, “Wizard Parameter Preview” shows a changed parameter value that is used to externalize an object that was matched by a node named "package". Note that here the type of the parameters has to be *Ignore* . This guarantees that the parameter does not affect the match-finding. The parameter will hold a reference to the object matched by the "package"-node.

Figure 4.4. Wizard Preview

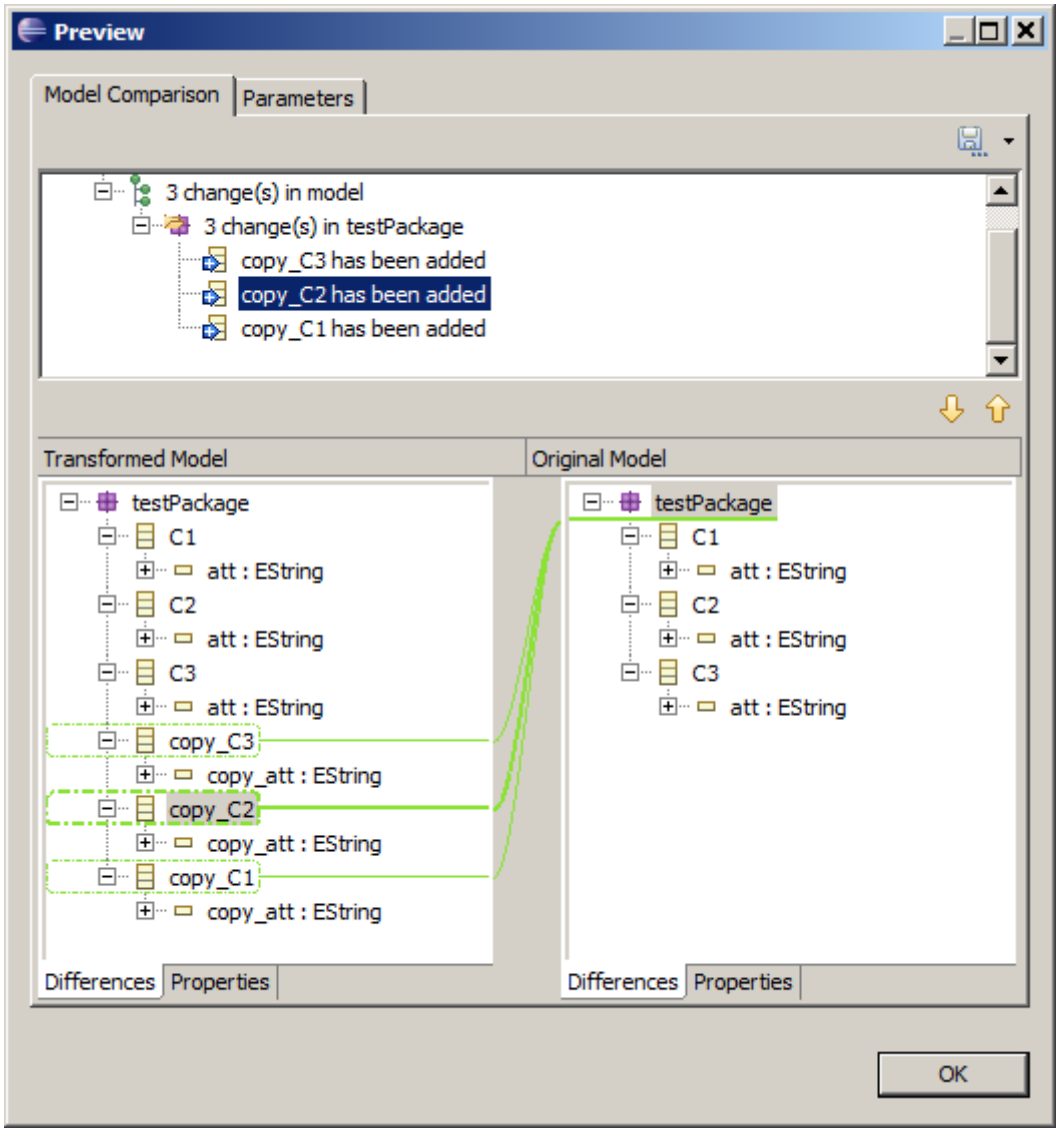


Figure 4.5. Wizard Parameter Preview