An Introduction to Metamodelling and Graph Transformations

with eMoflon

Part V: Miscellaneous

For eMoflon Version 2.16.0

File built on 21st September, 2016

For further information contact us at contact@emoflon.org.

The eMoflon team
Darmstadt, Germany (September 2016)

Contents

1 Grokking Enterprise Architect 2
2 Using existing EMF projects in eMoflon 13
3 eMoflon in a Jar 20
4 Useful shortcuts 23
5 Legacy support for CodeGen2 25
6 Creating and Using Enumerations in Enterprise Architect . . 27
7 Convert your TGG to textual syntax 29
8 Glossary 30
9 GNU General public license 34

Part VI:

And all that (eMoflon) jazz

URL of this document: https://emoflon.github.io/eclipse-plugin/beta/handbook/part5.pdf

Welcome to the miscellaneous part of our eMoflon handbook. You can consider this Part to be the 'bonus' or appendix area of the entire handbook series. Here we have collected and documented a series of advanced topics related to our tool. These include some tips and tricks you may find helpful while using the tool with Enterprise Architect (EA) and information about the protocol file generated with every Triple Graph Grammar (TGG) transformation which we were never able to explain in Parts IV or V. This entire part is kept rather compact, intended to be used mainly as a reference and consulted on demand.

Please note that if you're looking for instructions on how to properly export and import separate metamodels into the same project for work with TGG transformations, please refer to Part IV, Section 2.2, where we included detailed steps in the context of an example.

If you feel anything is missing from this part, or if you have any other comments or suggestions about the handbook series and our tool, feel free to contact us anytime at contact@emoflon.org.

1 Grokking Enterprise Architect

> Grok: "...to understand so thoroughly that the observer becomes a part of the observed."
>
> - Robert A. Heinlein, *Stranger in a Strange Land*

This section is a collection of a few of what we feel are the most important tips and tricks for working productively with Enterprise Architect (EA). We truly believe that spending the time to learn and practice these is necessary for a pleasant modelling experience.

1.1 Positioning elements

Layout is always an important factor when using a visual language: A well laid-out diagram is easiest to understand and, by centralizing important elements or clustering related elements, you can actually impart additional information.

- To select a group of elements, either drag a selection box around the items or hold `Ctrl` and select each element one-by-one.
- In the top right corner of the last selected element, a small colon-styled symbol will appear (Figure 1.1). Click on this for a context list of different options you can simultaneously apply to all active elements. The same list appears on the toolbar above the diagram.
- Experiment to find out what effect each option has. The last symbol in the list opens a further drop-down menu with standard layout algorithms to organize your diagram automatically.
- Right-clicking any of the selected elements opens a different menu with a further set of layout options and their descriptions (Figure 1.2). `Align Centers` or `Same Height and Width` can be especially useful.

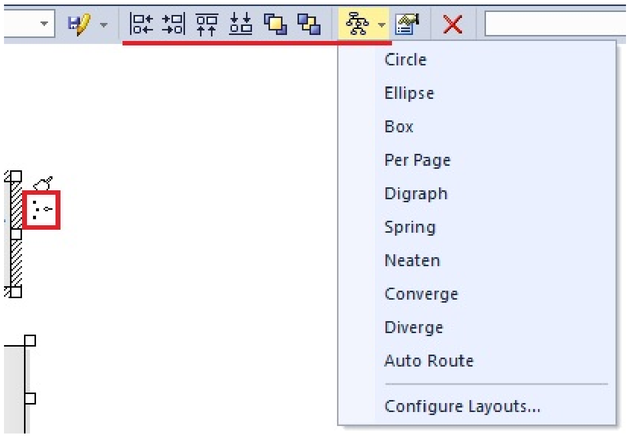

Figure 1.1: Setting the layout of multiple elements

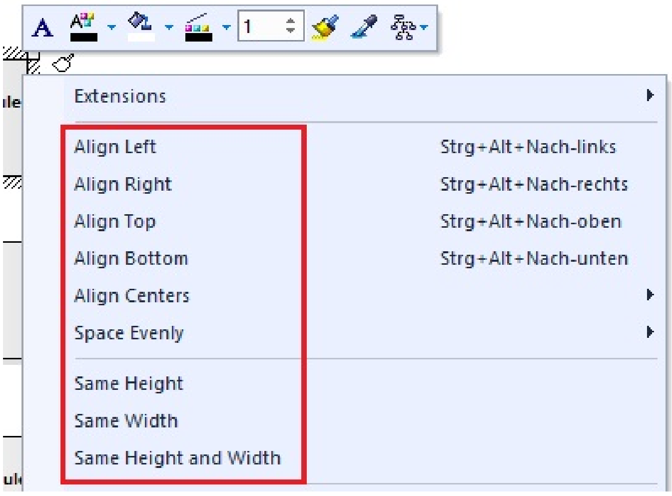

Figure 1.2: Further layout options

1.2 Bending lines to your will

Another important part of a good layout is getting lines to be just the way you want them to be. In EA you can add and remove bending points which can be used to control the appearance of a line.

- Hold down `Ctrl` and click on a line to create a bending point (Figure 1.3). You can now pull the bending point and shift the line as you wish.

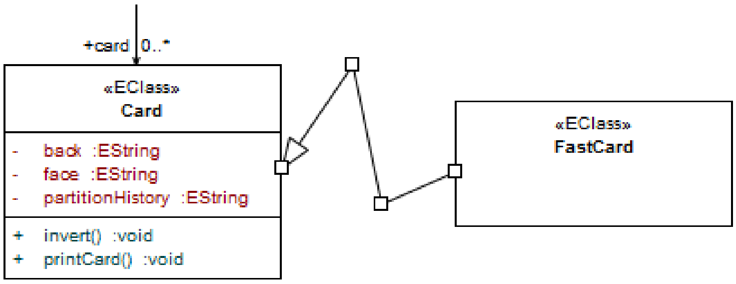

Figure 1.3: Adding bending points to a line

- You can create as many bending points as you wish, and you can *remove* them by holding down `Ctrl` and clicking once on the unwanted point.

1.3 Deleting vs. removing elements from diagrams

A central feature that new users should understand as soon as possible is the way EA handles diagrams. *A diagram is simply treated as a* view *of the complete model.* The complete model can always be browsed in its entirety via a tree view in the package browser. This space contains all elements that will be exported. The driving reason behind this setup is that diagrams typically do not contain all elements and one usually uses multiple (possibly redundant) diagrams to show the different parts of the model. Thinking in this frame is crucial and provides a pragmatic solution to the problem of having huge, unmaintainable diagrams.

A tricky consequence one must get used to is that removing an element from a diagram does *not* delete it from the model. We have added some support with the validation in the eMoflon add-in control panel, which can prompt a warning when an element cannot be found in any diagram,[1] but there's

[1] Review Part II, Section 2.8 for an example

currently no way to recover a deleted element.

A common mistake new users make is to remove an element by pressing `del`, and expecting the element to be deleted from the model. As you can probably guess, this is not the case as evidenced in the package browser (Figure 1.4).

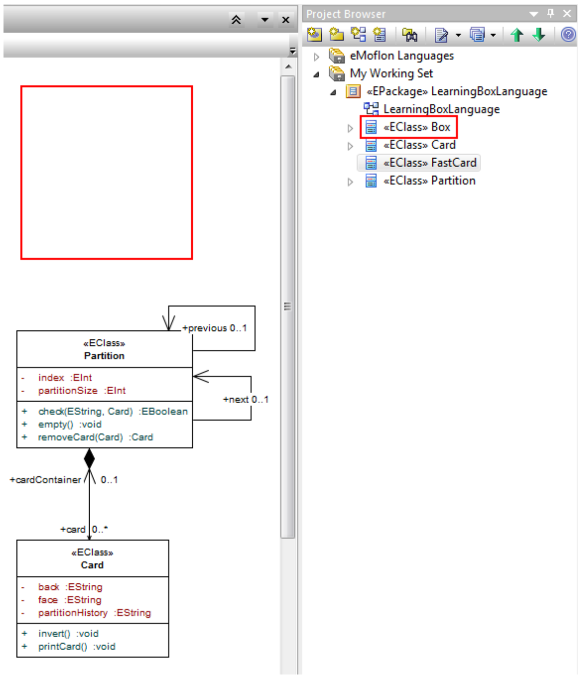

Figure 1.4: Removing an element from a diagram via pressing `Del` does not delete it from the model and it is still present in the package browser

- To fully delete an element from a model (not just a diagram), select it in the diagram and press `Ctrl + Del`. Confirm the action in the warning dialogue (Figure 1.5), and the element should no longer be in the project browser.

- Alternatively, elements can be deleted directly from project browser by right-clicking the item and navigating to the large red 'x' at the bottom of the context menu

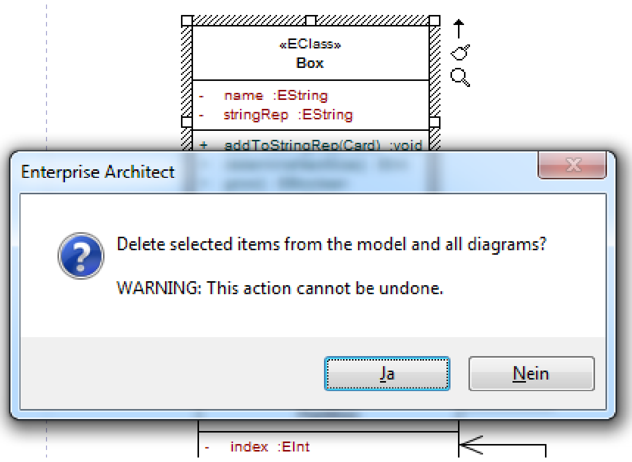

Figure 1.5: Deleting an element from a diagram and the model

1.4 Excluding certain projects from the export

You may find it sometimes necessary to exclude certain projects from your diagram export (such as the `MocaTree` model used in Part V). Some reasons for this could be (i) because the project is still a work in progress and simply not ready to be exported, (ii) because the complete project is present in the Eclipse workspace but has not been modelled completely in EA, and you wish to do this gradually on-demand, (iii) because the project is not meant to be present in your Eclipse workspace as generated code and is instead provided via a plugin (this is usually the case for standard metamodels like Ecore, UML etc.), or (iv) because the project is rather large and stable and you do not want to wait for EA to process a known, unchanging model. Whatever the reason, you can prevent unnecessary exports by setting a certain *tagged value* of the project.

- ▶ Open your project in EA, and navigate to "View/Tagged Values" from the menu bar (Figure 1.6).
- ▶ The tagged value, *Moflon::Export*, should already be present and be set to a default `true` value (Figure 1.7). If you want the project to be ignored by the eMoflon's validation and/or export functions, change the value to `false` (and conversely back to `true` to export it again).

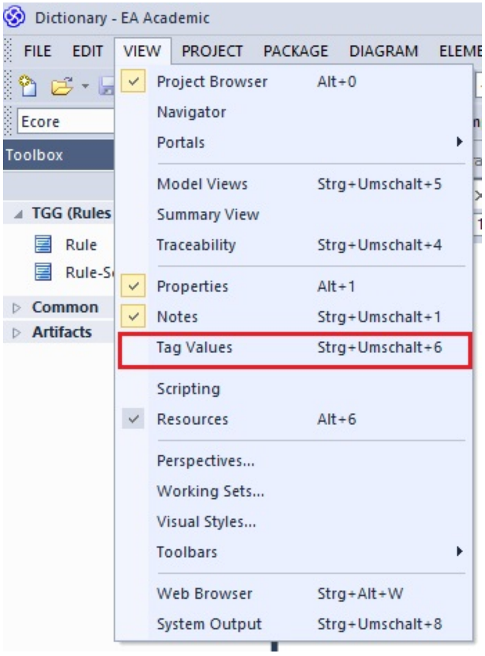

Figure 1.6: Opening the tagged values view

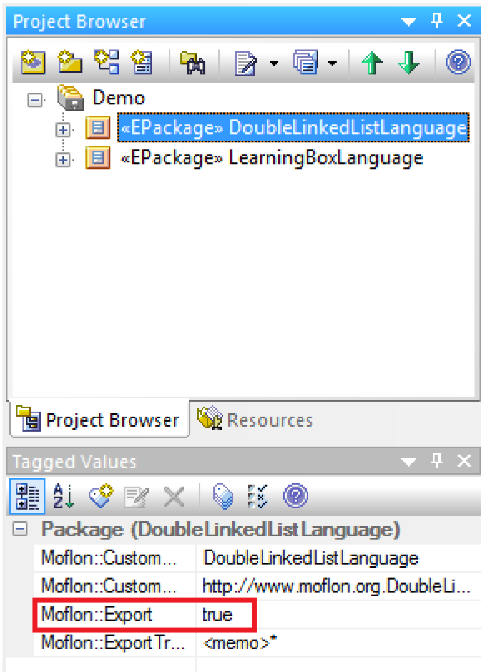

Figure 1.7: The `Moflon::Export` setting determines ignored projects

1.5 Getting verbose!

Although we use colours in SDMs to indicate when an element is to be matched (black), created (green), or destroyed (red), it sometimes makes sense to indicate these binding operators via explicit stereotypes (i.e., for black-and-white printouts of a model).

- ▶ Open the relevant diagram in the EA editor window and, depending on what type it is, press the `Verbose` button in either the `eMoflon SDM Functions` or `eMoflon TGG Functions` panel (Figure 1.8).

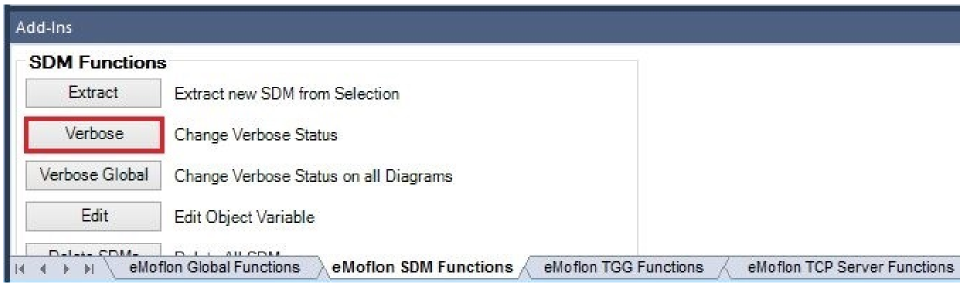

Figure 1.8: Add extra markup to colored links and objects in the current diagram

- ▶ This will add small `++` or `- -` symbols next to deleted and created elements in the current diagram (Figure 1.9). Press the button again to deactivate these indicators.

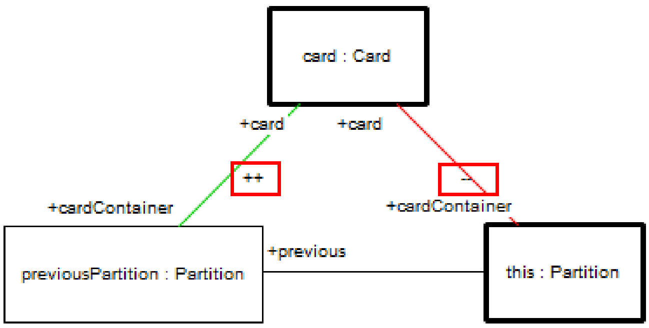

Figure 1.9: Diagram in verbose mode

1.6 Duplicating elements via drag-and-drop

Sometimes you'll have an element (or many) that are nearly identical, and life would be *so* much easier if you could copy and paste an existing one already. Suppose you want a copy of a `this` element, so you press `Ctrl + C`, followed by `Ctrl + V`. An error dialogue preventing the action will immediately raise, stating that the "…diagram already contains an instance of the element you are trying to paste." EA can only support unique objects, so you'll need to use the following process.

- In either a diagram or in the project browser, hold `Ctrl`, then drag the element you wish to duplicate. A confirmation-style dialogue will appear (Figure 1.10), and a properties window will follow. You must assign a unique name to the new element, or else you'll receive an error when you try to export the project later.

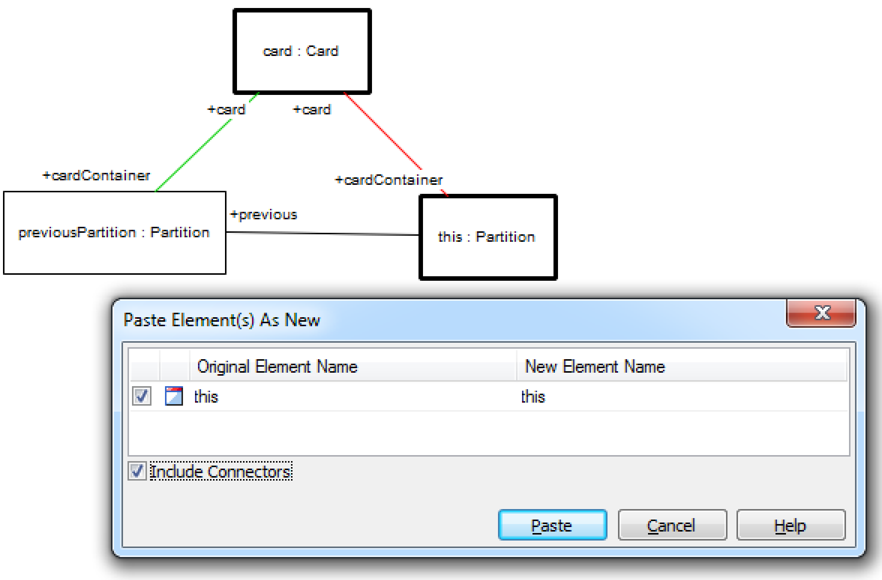

Figure 1.10: Copying elements

1.7 Seek, and ye shall find ...

EA has a model search function that can be quite handy for large models with thousands of elements and a brain that can't *quite* remember where something is.

- Select `Model Search Window` in the toolbar and enter the name of an element you wish to find (Figure 1.11).[2]

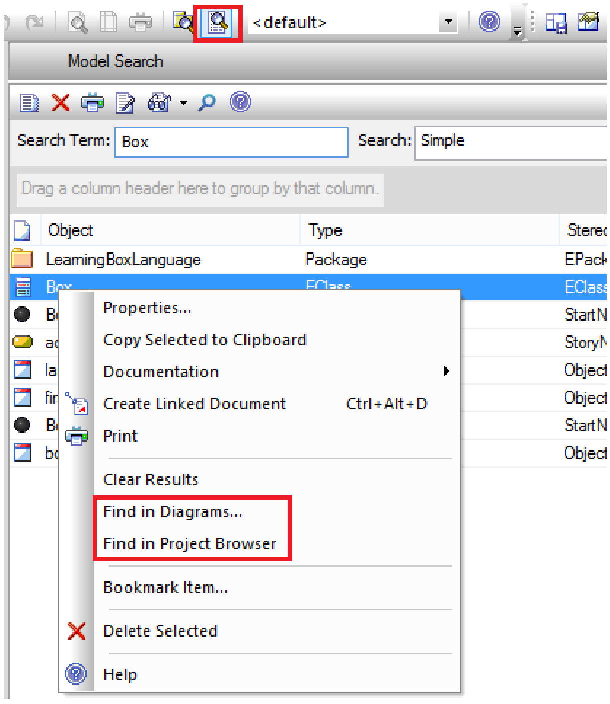

Figure 1.11: Model Search Window

- All elements that meet the search criteria are listed and you can right-click on each of the items and select one of the options above to locate the element.
- In a similar way, you can locate the corresponding class of an object by right clicking and selecting "Find/Locate Classifier in Project Browser."

[2]You can also access this window by pressing `Ctrl+Alt+A`

1.8 Advanced search

EA offers an even more advanced search capability using SQL.[3]

- To use this, first open the model search window via either the menu bar or by pressing `Ctrl + Alt + A`.
- Click the "Builder" button, and switch to the `SQL` tab (Figure 1.12).

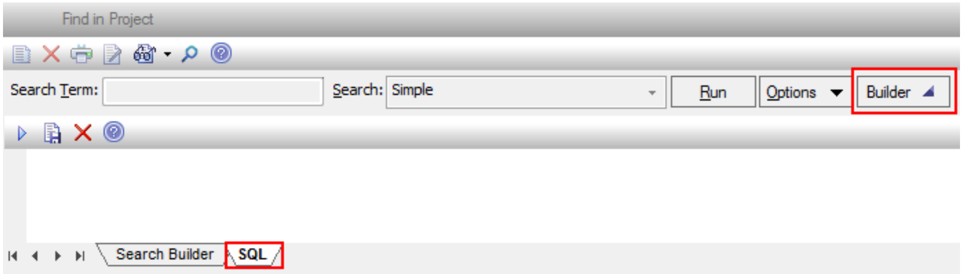

Figure 1.12: Advanced project search window

Here you can formulate any query on the underlying database. The SQL-editor helps you with syntax-highlighting and auto-completion. Here are some basic examples to get you started:

- To find all eClasses

```sql
SELECT * FROM t_object
WHERE Object_Type='Class' AND Stereotype='eclass';
```

- To find all associations

```sql
SELECT * FROM t_connector
WHERE Connector_Type='Association';
```

- To find all inheritance relations

```sql
SELECT * FROM t_connector
WHERE Connector_Type='Generalization';
```

- To find all connectors attaching a note to an element

```sql
SELECT * FROM t_connector
WHERE Connector_Type='NoteLink';
```

[3] For some detailed insights to the general database schema used by EA cf.
`http://www.sparxsystems.com.au/downloads/corp/scripts/SQLServer_EASchema.sql`

- To find all control flow edges (used in SDMs)

```
SELECT * FROM t_connector
WHERE Connector_Type='ControlFlow';
```

- To find all associations connected to a class named “EClass”

```
SELECT t_object.Name, t_connector.* FROM t_connector,t_object
WHERE t_connector.Connector_Type='Association'
  AND (t_connector.Start_Object_ID=t_object.Object_ID
    OR t_connector.End_Object_ID=t_object.Object_ID)
  AND t_object.Name='EClass';
```

- To determine all subtypes of “EClassifier”

```
SELECT a.Name FROM t_connector,t_object a,t_object b
WHERE t_connector.Connector_Type='Generalization'
  AND t_connector.Start_Object_ID=a.Object_ID
  AND t_connector.End_Object_ID=b.Object_ID
  AND b.Name = 'EClassifier';
```

- To determine all supertypes of “EClassifier” (cf. above)

```
...
  AND t_connector.Start_Object_ID=b.Object_ID
  AND t_connector.End_Object_ID=a.Object_ID
...
```

To run the search, either hit the `Run SQL` button in the upper left corner of the editor toolbar (it shows a triangular shaped “play” icon), or press `F5` on your keyboard.

2 Using existing EMF projects in eMoflon

This chapter contains stepwise instructions on how to use existing EMF/Ecore projects with an eMoflon project. We will present an example of an existing metamodel which must be integrated with eMoflon before, for example, its transformation using SDMs can be specified. The basic workflow for using an existing EMF project in eMoflon is described in the following.

We will begin by implementing a small subset of the `Ecore -> GenModel` transformation, where `GenModel` is part of the EMF/Ecore standard. The *GenModel* for a given Ecore model can be viewed as a *wrapper* that contains additional generation-specific Java code details. These details are separated from the Ecore model to keep it free of such "low-level" information and settings.

2.1 Modelling relevant aspects in EA

The first step is to load an existing metamodel into EA. A complete and automatic import of existing Ecore files in EA is currently not possible and therefore, *relevant parts* of the existing metamodel (`GenModel`) have to be modelled manually. Although this might sound frightening (especially for large, complex metamodels), the emphasis here on *relevant* indicates that only elements that are needed for the transformation have to be present in EA, where more can be added iteratively as the transformation grows.

If you find this section challenging or unclear, refer to Part II: Ecore for a detailed review of metamodel construction.

- ▶ Open Eclipse and create a new metamodel project named `EcoreTo-GenModel`, do not select the `Add Demo Specification` option in the project wizard window.
- ▶ A new specifications folder with the project name should have been loaded into the workspace.
- ▶ Double-click the generated `EcoreToGenModel.eap` file to open your project in EA. Explore the project browser and make note of the packages already present in EA under `eMoflon Languages`, especially `Ecore` which we shall use in this transformation.
- ▶ Select the root note `My Working Set` and create a new package named `GenModelLanguage`.
- ▶ Add a new Ecore diagram and model the elements as depicted in Figure 2.1. You'll need to create the three EClasses on the left, but

Ecore::EPackage and Ecore::EClass are to be drag-and-dropped and pasted as links from the project browser.

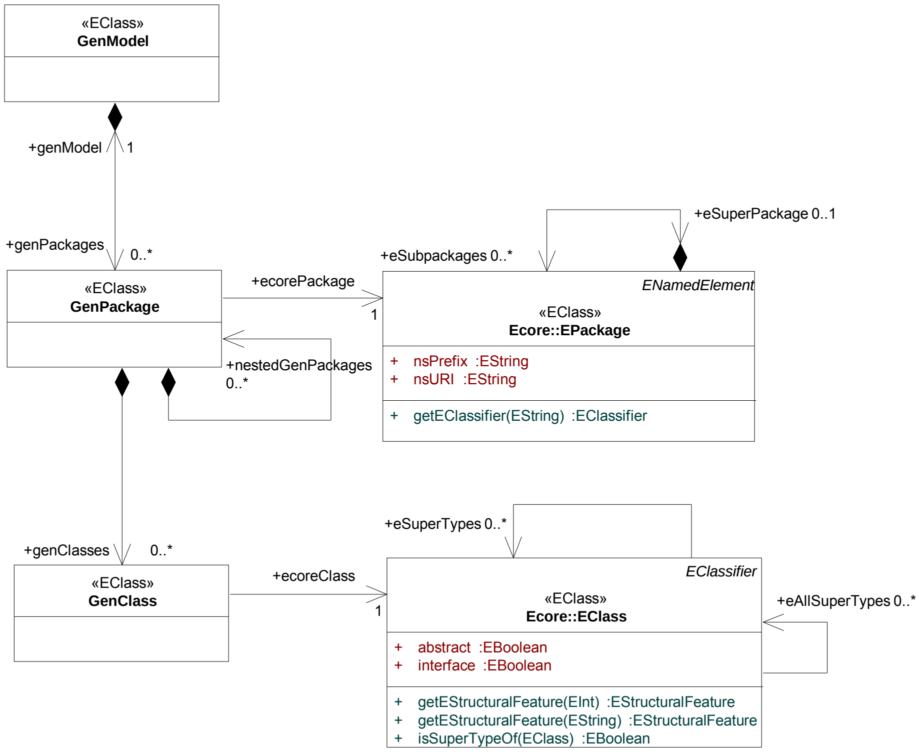

Figure 2.1: Metamodel of GenModel

- ▶ Please note that the actual GenModel metamodel contains many more elements, but this subset is sufficient for our task. Although this subset can be incomplete, it must be correct and not contradict the actual GenModel metamodel in any way!
- ▶ Navigate to the project browser again and create another package named Ecore2GenModel. This will contain the Transformer class; Create and complete its Ecore diagram as depicted in Figure 2.2.
- ▶ Carefully double-click each method to create and implement their SDMs as depicted in Figures 2.3 and 2.4.

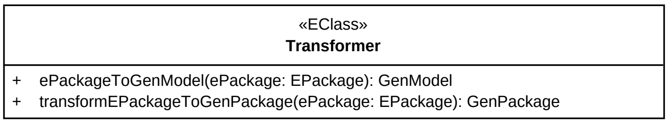

Figure 2.2: Methods in `Transformer`

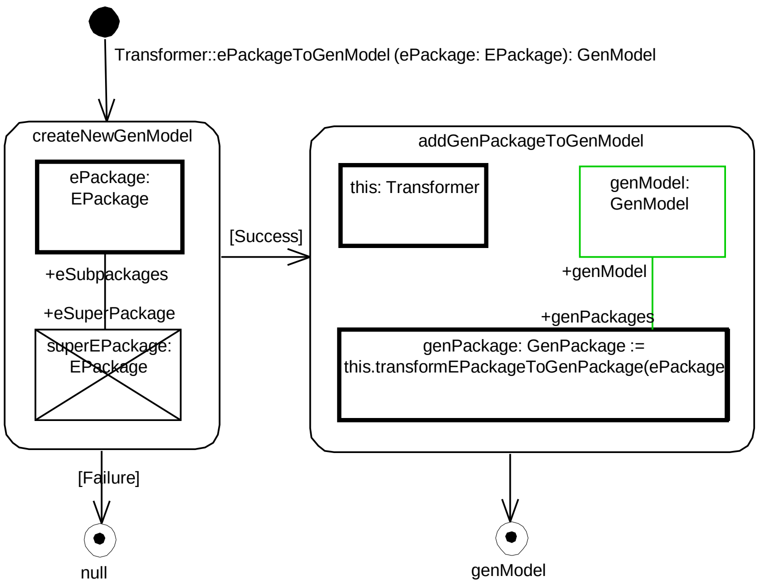

Figure 2.3: Main method for `EPackage` to `GenModel` transformation

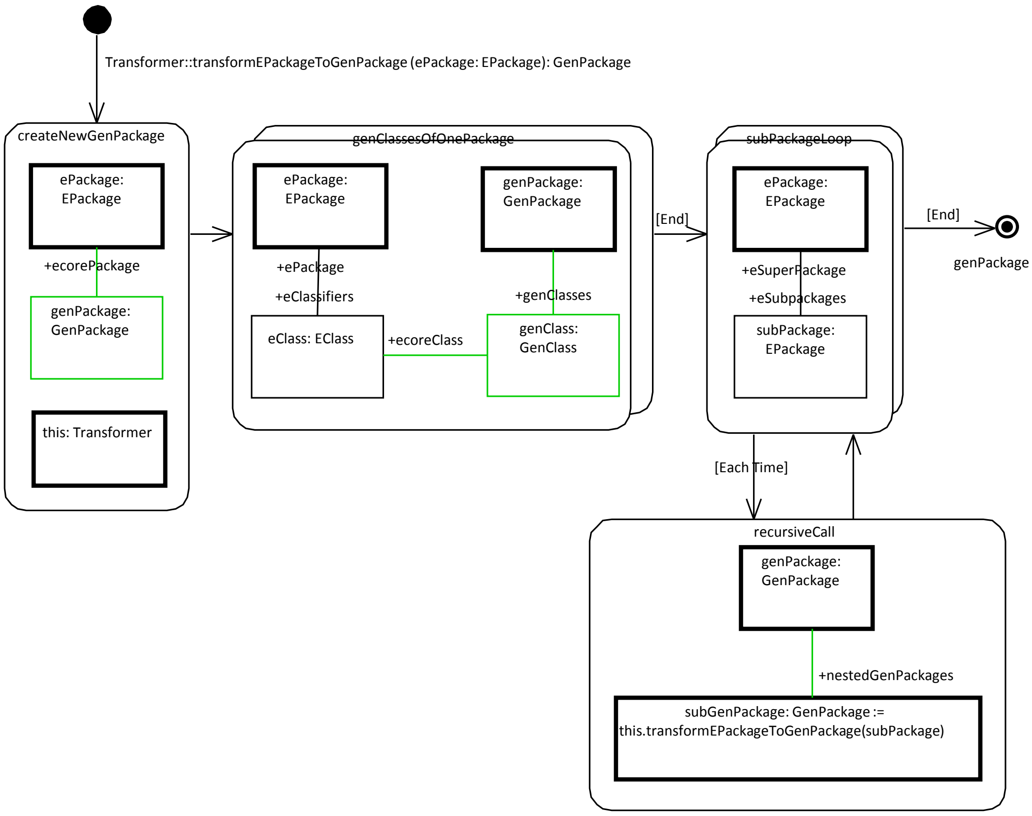

Figure 2.4: Helper function to transform all `EPackages` to `GenPackages`

2.2 Configuration for code generation in Eclipse

Since there is already generated code for the existing `GenModel` metamodel (provided via the Eclipse plugin), we do *not* want to export our incomplete subset of `GenModel` from EA. Instead, we need to configure Eclipse to access the elements specified in our partial metamodel from the complete metamodel.

- ▶ In EA, right-click your `GenModelLanguage` package and select "Properties..."
- ▶ Navigate to "Properties/Moflon" in the dialogue window and update the tagged `Moflon::Export` value to `false` (Figure 2.5).

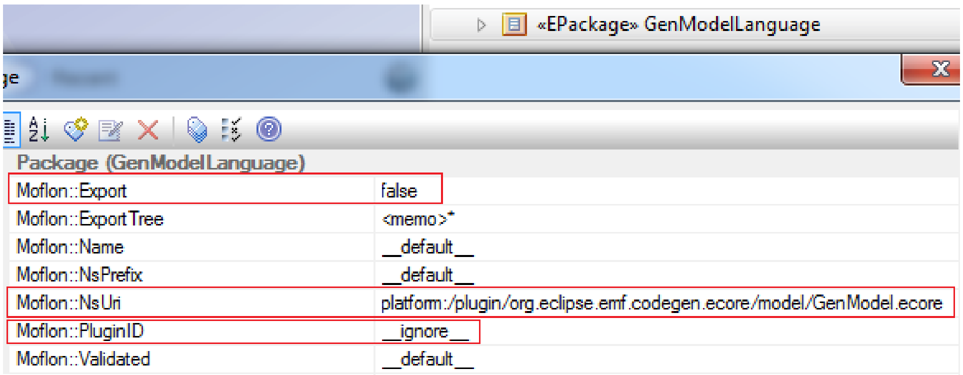

Figure 2.5: Update the `GenModel` export option and other tagged values

- ▶ Next we have to set the "real" URI of the project to be used in Eclipse so that the relevant references are exported properly. Set the value of `Moflon::NsUri` to `platform:/plugin/org.eclipse.emf.codegen.ecore/model/GenModel.ecore`. As the default plugin ID generation provided by eMoflon is also not valid here, set the value of `Moflon::PluginID` to `__ignore__` (two underscores before and after!). The three relevant values to be set are shown in Figure 2.5.
- ▶ Validate and export all projects as usual to your Eclipse workspace, and update the metamodel project by pressing `F5` in the package explorer.
- ▶ Right-click `Ecore2GenModel` once more and navigate to "Plug-in Tools/Open Manifest." The plug-in manager should have opened in the editor with a series of tabs at the bottom.

- ▶ Switch to the `Dependencies` tab. Press `Add` and enter `org.eclipse.-emf.codegen.ecore`. This plug-in includes both the `Ecore` and `Gen-Model` libraries we require for successful compilation of the transformation code.

Although we have already specified the URI of the existing project (in this example, `GenModel`) as tagged project values, we still have to configure a few things for code generation.

- ▶ Expand the `Ecore2GenModel` project folder and open the `moflon.-properties.xmi` file tree. Right-click the properties container, and create a new `Additional Dependencies` child. Double click the element to open its properties tab below the editor, and as shown in Figure 2.6, update its `Value` to:

`platform:/plugin/org.eclipse.emf.codegen.ecore/model/GenModel.ecore`

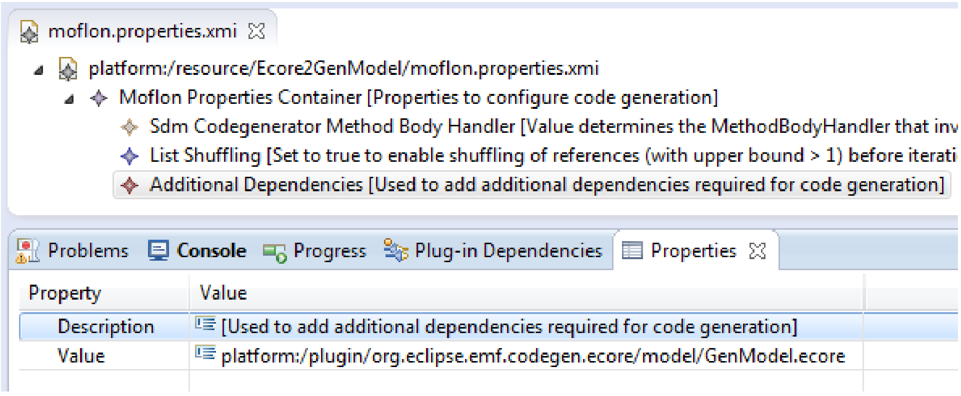

Figure 2.6: Setting properties for code generation

- ▶ Similarly, add a second `Additional Used Gen Packages` child and set its value to:

`platform:/plugin/org.eclipse.emf.codegen.ecore/model/GenModel.genmodel`

Finally, to compenstate for some cases where our naming conventions were violated, analogously add the following mapping as corrections:

- Add an *import mapping* child for correct generation of imports, setting the key as `genmodel` (depicted in Figure 2.7) and value to:
  `org.eclipse.emf.codegen.ecore.genmodel`

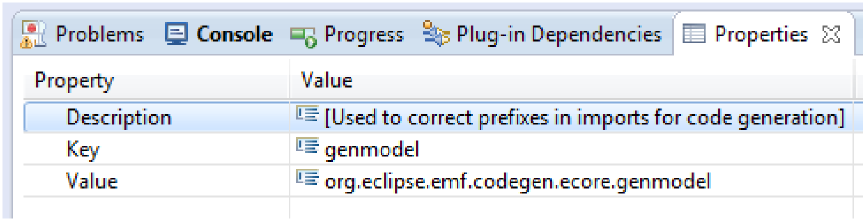

Figure 2.7: Correcting default conventions for generating imports

- Finally, add a *factory mapping* to ensure that `GenModelFactory` is used as the factory for creating elements in the transformation instead of `GenmodelFactory`, which would be the default convention. Set its key as `genmodel`, and its value to: `GenModelFactory`.
- Its now time to generate code for the project. If everything worked out and the generated code compiles, you can ensure that the transformation behaves as expected by invoking the methods and transforming an ecore file to a corresponding genmodel.

As a final remark, note that import and factory mappings are not always necessary, `GenModel` is in this sense a particularly nasty example as it violates all our default conventions.

3 eMoflon in a Jar

This section describes how to package code generated with eMoflon into runnable Jar files, which is useful if you want to build applications for end-users.

We distinguish between repository, i.e., SDM-based, and integration, i.e., TGG-based, projects.

3.1 Packaging SDM projects into a Jar file

The following explanations use the demo specification that is shipped with eMoflon to explain the workflow of building a runnable Jar file.

- ▶ Open a fresh workspace and add to it the eMoflon Demo specification by selecting the "Install, configure and deploy Moflon" button and open the "Install Workspace" menu bar. Select the "Demo Workspace".
- ▶ Generate code for the demo and verify the result by running the test cases in *DemoTestSuite*.
- ▶ Add a suitable main method to `NodeTest`, for instance:

```
public static void main(String[] args) {
    System.out.println("Begin of test runs");
    new NodeTest().testDeleteNode();
    new NodeTest().testInsertNodeAfter();
    new NodeTest().testInsertNodeBefore();
    System.out.println("End of test runs");
}
```

- ▶ Run `NodeTest` as "Java Application" (*not* as "JUnit Test"). Now you have a new launch configuration named "NodeTest".
- ▶ Now, select the repository project (containing the generated code) and the project *DemoTestSuite*. You do not need to add the project containing the EA project. Right-click and select "Export...". Choose "Runnable JAR file".
- ▶ On the next page, select the launch configuration you just created by running `NodeTest` and an appropriate target location for your Jar file. The libraries should be packaged or extracted into the generated Jar file.
- ▶ Afterwards, open up a console in the folder containing the generated Jar file and execute it as follows:

```
java -jar [GeneratedJarFile.jar]
```

3.2 Packaging TGG projects into a Jar file

In the following, you will create a runnable Jar from a TGG specification. We assume that you have some existing TGG implementation and that you want to execute the `main` method in class `org.moflon.tie.MyIntegrationTrafo`.

Note: The following instructions show how to use Eclipse's built-in facility for generating runnable Jars. There are other build tools such as ant, Maven or Gradle that facilitate this process.

- ▶ Ensure that your TGG rules from within Eclipse. For simplicity, we assume that your main method currently resembles the following snippet:

```
public static void main(String[] args) throws IOException {
    // Set up logging
    BasicConfigurator.configure();

    // Forward Transformation
    MyIntegrationTrafo helper =
        new MyIntegrationTrafo();
    helper.performForward("instances/fwd.src.xmi");
}
```

  The default transformation helper should print a short success message when the forward transformation has finished.

- ▶ Before packaging your project, you have to change the ways how the TGG rules are being loaded (in the constructor of `MyIntegrationTrafo`). Replace this method call

```
loadRulesFromProject("..");
```

  with

```
File jarFile = new File(MyIntegrationTrafo.class
    .getProtectionDomain().getCodeSource()
    .getLocation().toURI().getPath());
loadRulesFromJarArchive(
    jarFile,
    "/MyIntegration.sma.xmi");
```

  This is a tiny trick to find out the name of the Jar file that you are about to build. If you already know the name of your Jar file (e.g., "tggInAJar.jar"), you could simply use the following code:

```
loadRulesFromJarArchive(
    "tggInAJar.jar",
    "/MyIntegrationTrafo.sma.xmi");
```

- Next, make the "model" directory a source folder by right-clicking it and selecting "Build Path/Use as Source Folder". This will make the contents of "model" available in the Jar file to be built.

- Now, your projects are ready to be packaged. Select all projects that are involved in your TGG, that is, the project of the source and target metamodel as well as the actual integration project.

  Right-click the projects and select "Export..." and then "Java/Runnable JAR File".

- Select the appropriate launch configuration (named "MyIntegrationTraf"), choose the export destination, and make sure that the library handling is set to "Extract required libraries".

- After a successful export, locate the generated Jar file. The program expects to find the source model of the transformation at the following path, relative to the folder containing your Jar file: "instances/fwd.src.xmi".

  Now, let's take the transformation for a spin:

  ```
  java -jar [GeneratedJarFile.jar]
  ```

4 Useful shortcuts

This page is a simple list of special hotkeys you might find useful while working with eMoflon in either EA or Eclipse. Please note standard shortcuts, such as `Ctrl + S` and `Ctrl + Z`, are still applicable in most cases.

4.1 In Eclipse (general)

Note: `I` indicates *in Integrator window*, and `GK` indicates *German keyboards only*

`Ctrl + Space`	Auto-type completion
`Ctrl + 1` (problems tab)	Quick-fix menu
`Alt + arrow` (I)	Proceed to next step
`Shift + Alt + arrow` (I)	Fast navigation
`Shift + Ctrl + Alt + arrow` (I)	Proceed to next breakpoint
`Shift + Ctrl + AltGr + arrow` (I, GK)	Proceed to next breakpoint

4.2 In Eclipse (eMoflon Plugin)

`Alt + Shift + E`	Open list of all available eMoflon commands
`Alt + Shift + E, B`	Trigger a build without clean
`Alt + Shift + E, C`	Trigger a clean and build
`Alt + Shift + E, D`	Convert file to visual representation (dot)
`Alt + Shift + E, G`	Start integrator (on correspondence file)
`Alt + Shift + E, I`	Create/update injections
`Alt + Shift + E, M`	Convert project to textual syntax
`Alt + Shift + E, P`	Add ANTLR parser and/or unparser
`Alt + Shift + E, V`	Valide Ecore file
`Alt + Shift + E, X`	Export and build EAP file

4.3 In EA

Note: `D` indicates *in Diagram*, and `PB` indicates *in Project Browser*

`Alt + Enter`	Selected element `Properties` dialogue
`F9 + EClass`	Class `Attribute` editor
`F10 + EClass`	Class `Operations` editor
`Space` (D)	Current toolbar menu
`Del + Ctrl + element` (D/PB)	Delete element from model
`Ctrl + element` (D/PB)	Duplicate and create new element
`Ctrl + Alt + A`	Open Model Search Window
`Alt + G + element` (D)	Highlight Element (PB)

5 Legacy support for CodeGen2

Since eMoflon 1.8, the default code generator is *Democles*. The previous code generator *CodeGen2* is available, but no longer officially supported.

This sections describes how to configure your project to use CodeGen2.

- Install the eMoflon CodeGen2 feature: Select "Help/Install new software...", choose the eMoflon update site and tick "eMoflon CodeGen2" (Figure 5.1). Proceed with "Next" and follow the instructions to install the feature.

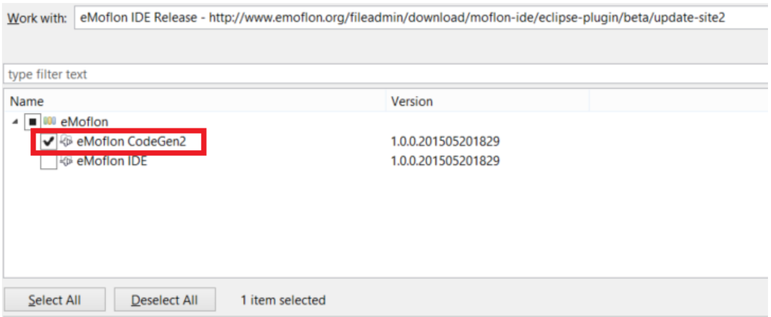

Figure 5.1: Installing the eMoflon CodeGen2 feature

- Open the file "moflon.properties.xmi" in your project(s) and set the code generation strategy to CODEGEN2 (Figure 5.2).

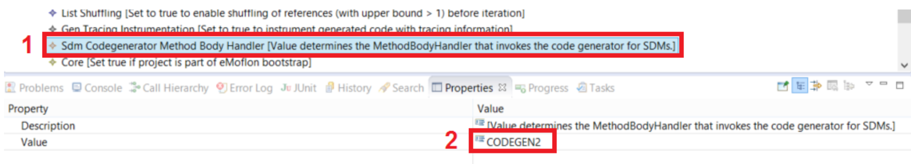

Figure 5.2: Code generation strategy selection in "moflon.properties.xmi"

- Open the file "META-INF/MANIFEST.MF" in your project(s) and add the following dependency: *org.moflon.sdm.codegen2.runtime* (Figure 5.3).
- Clean and build your project using the eMoflon context menu (Alt+Shit+E, B).

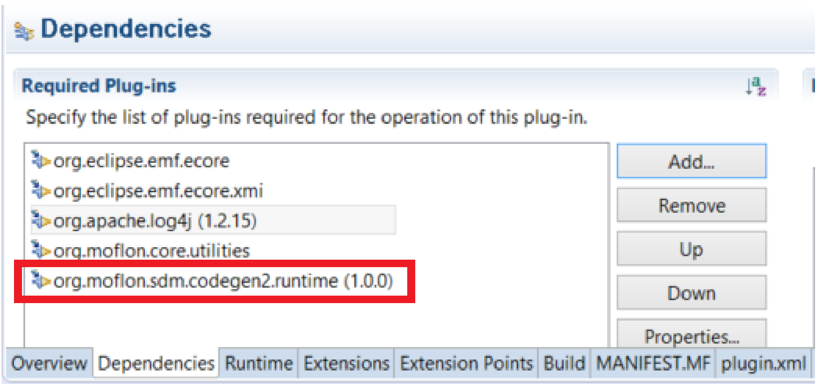

Figure 5.3: Dependency to CodeGen2 runtime in "MANIFEST.MF"

6 Creating and Using Enumerations in Enterprise Architect

This section describes how to create enumeration types in your metamodel. For illustration purposes, we use the linked-list demonstration project from Part I.

Suppose we want to assign one of several predefined colors (e.g., red, green, blue) to each `Node` in a `List`. To represent the colors, we create an enumeration called `Color` with three elements: `Color.RED`, `Color.GREEN` and `Color.BLUE`.

- ▶ Let's start with the double-linked list demonstration specification, which is readily provided with eMoflon: Create a new meta-model project ("File/New/Other...", then "eMoflon/New Metamodel Wizard"), call your project "Demo", and tick "Add Demo Specification".
- ▶ Open the EAP file "Demo.eap" and navigate to the diagram "org.-moflon.demo.doublelinkedlist".
- ▶ Now, add a new "EEnum" type called color. This can be done via the Toolbox ("Diagram/Toolbox") or by pressing space while the cursor is inside the digram area. Your diagram should resemble Figure 6.1.

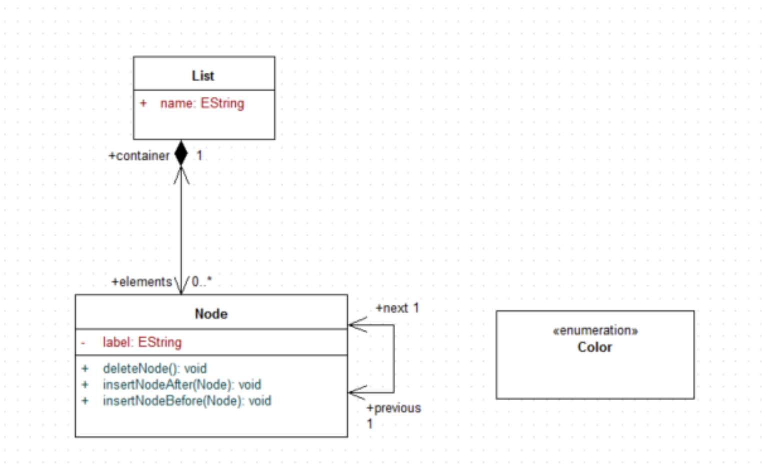

Figure 6.1: Creating a new EEnum

- ▶ We now create the three colors that our list nodes may have. An enum constant is a special attribute. Therefore, open the attriutes view of `Color` ("Right-click/Features & Properties/Attributes...") and add three attributes as shown in Figure 6.2. Make sure that the type of the attributes is `Color` and that each attribute has an “initial value”.

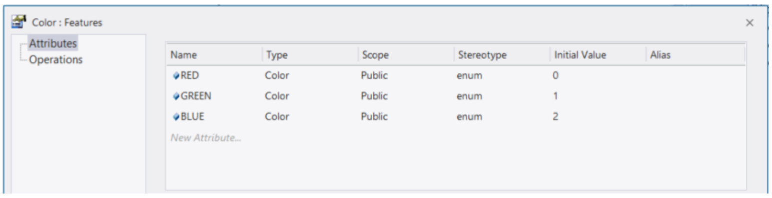

Figure 6.2: Creating the three color attributes `RED`, `GREEN`, and `BLUE`

- Finally, create a `color` attribute of type `Color` in EClass `Node` (Figure 6.3).

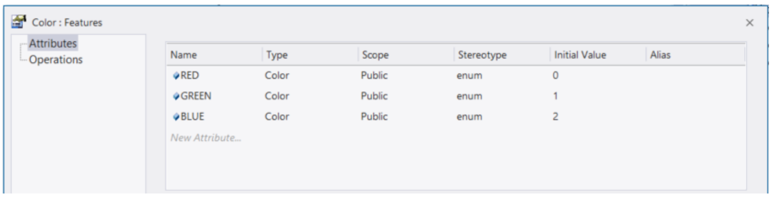

Figure 6.3: `color` attribute of EClass `Node`

- Validate and export and build your metamodel.
- A minimal test for the new feature could be implemented in the class `NodeTest` as follows:

Listing 6.1: Test for coloring nodes

```
@Test
public void testAddColor() throws Exception {
        Node node =
            DoublelinkedlistFactory.eINSTANCE.createNode();
        node.setColor(Color.RED);
}
```

7 Convert your TGG to textual syntax

Your "old" TGG specified with EA (visual syntax) can easily be converted to our new textual syntax:

- ▶ In Eclipse, navigate to the `model` folder of your TGG project and right click on the `pre.tgg.xmi` file (`<your TGG>.pre.tgg.xmi`). Choose `Convert to MOSL` from the eMoflon context menu (Figure 7.1).
- ▶ In `src/org.moflon.tgg.mosl`, check the created `.tgg` file which represents your TGG in textual syntax. (Figure 7.2).

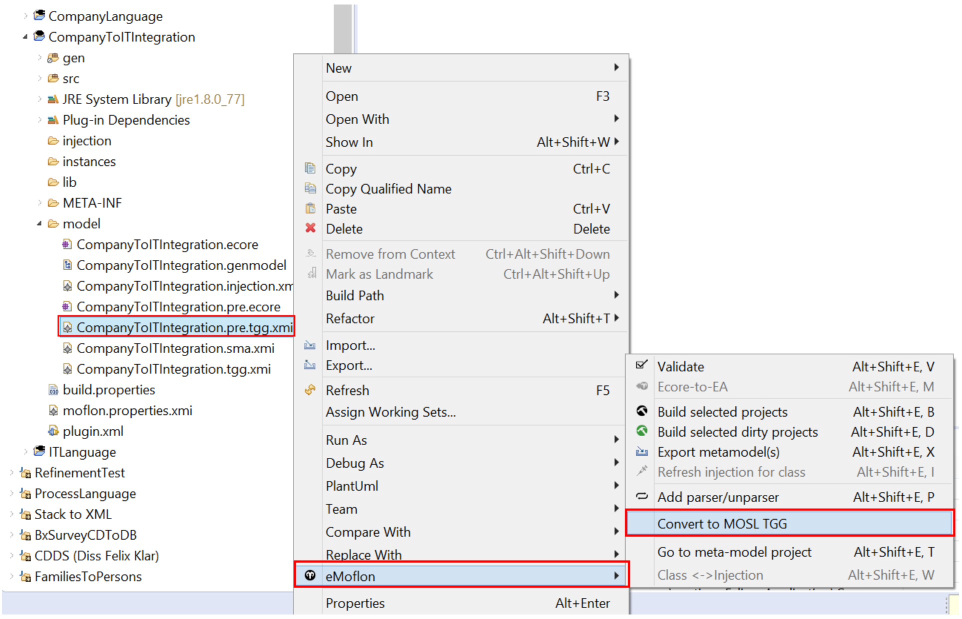

Figure 7.1: Convert your TGG to textual syntax

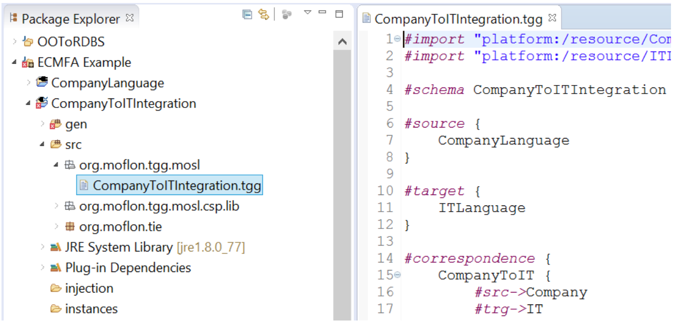

Figure 7.2: The created `.tgg` file representing your TGG in textual syntax

8 Glossary

Abstract Syntax Defines the valid static structure of members of a language.

Activity Top-most element of an SDM.

Activity Edge A directed connection between activity nodes describing the control flow within an activity.

Activity Node Represents atomic steps in the control flow of an SDM. Can be either a story node or statement node.

Assignments Used to set attributes of object variables.

Attribute Constraint A non-structural constraint that must be satisfied for a story pattern to match. Can be either an assertion or assignment.

Bidirectional Model Transformation Consists of two unidirectional model transformations, which are consistent to each other. This requirement of consistency can be defined in many ways, including using a TGG.

Binding State Can be either *bound* or *unbound/free*. See *Bound vs Unbound*.

Binding operator Determine whether a variable is to be *checked*, *created*, or *destroyed* during pattern matching.

Binding Semantics Determines if an object variable *must* exist (*mandatory*), may not exist (*negative*; see *NAC*), or is *optional* during *pattern matching*.

Bound vs Unbound Bound variables are completely determined by the current context, whereas unbound (free) variables have to be determined by the *pattern matcher*. `this` and parameter values are always bound.

Concrete Syntax How members of a language are represented. This is often done textually or visually.

Constraint Language Typically used to specify complex constraints (as part of the static semantics of a language) that cannot be expressed in a metamodel.

Correspondence Types Connect classes of the source and target metamodels.

Dangling Edges An edge with no target or source. Graphs with dangling edges are invalid, which is why dangling edges are avoided and automatically deleted by the pattern matching engine.

Dynamic Semantics Defines the dynamic behaviour for members of a language.

EA Enterprise Architect; The UML visual modeling tool used as our visual frontend.

EBNF Extended Backus-Naur Form; Concrete syntax for specifying context-free string grammars, used to describe the context-free syntax of a string language.

Edge Guards Refine the control flow in an activity by guarding activity edges with a condition that must be satisfied for the activity edge to be taken.

Endogenous Transformations between models in the same language (i.e., same input/output metamodel).

Exogenous Transformations between models in different languages (i.e., unique metamodel instances).

Grammar A set of rules that can be used to generate a language.

Graph Grammar A grammar that describes a graph language. This can be used instead of a metamodel or type graph to define the abstract syntax of a language.

Graph Triples Consist of connected source, correspondence, and target components.

In-place Transformation Performs destructive changes directly to the input model, thus transforming it into the output model. Typically *endogenous*.

Link or correspondence Metamodel Comprised of all correspondence types.

Link Variable Placeholders for links between matched objects.

Literal Expression Represents literals such as true, false, 7, or "foo."

Meta-Language A language that can be used to define another language.

Meta-metamodel A *modeling language* for specifying metamodels.

Metamodel Defines the abstract syntax of a language including some aspects of the static semantics such as multiplicities.

MethodCallExpression Used to invoke any method.

Model Graphs which conform to some metamodel.

Modelling Language Used to specify languages. Typically contains concepts such as classes and connections between classes.

Monotonic In the context of TGGs, a non-deleting characteristic.

NAC Negative Application Condition; Used to specify structures that must not be present for a transformation rule to be applied.

Object Variable Place holders for actual objects in the current model to be determined during pattern matching.

ObjectVariableExpression Used to reference other object variables.

Operationalization The process of deriving step-by-step executable instructions from a declarative specification that just states what the outcome should be but not how to achieve it.

Out-place Transformation Source model is left intact by the transformation which creates the output model. Can be *endogenous* or *exogenous*.

Parameter Expression Used to refer to method parameters.

(Graph) Pattern Matching Process of assigning objects and links in a model to the object and link variables in a pattern in a type conform manner. This is also referred to as finding a match for the pattern in the given model.

Statement Nodes Used to invoke methods as part of the control flow in an activity.

Static Semantics Constraints members of a language must obey in addition to being conform to the abstract syntax of the language.

Story Node *Activity nodes* that contain *story pattern*s.

Story Pattern Specifies a structural change of the model.

Triple Graph Grammars (TGG) Declarative, rule-based technique of specifying the simultaneous evolution of three connected graphs.

Type Graph The graph that defines all types and relations that form a language. Equivalent to a metamodel but without any static semantics.

TGG Schema The metamodel triple consisting of the source, correspondence (link), and target metamodels.

Unification An extension of the object oriented "Everything is an object" principle, where everything is regarded as a model, even the metamodel which defines other models.

9 GNU GENERAL PUBLIC LICENSE

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. `http://fsf.org/` Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see `http://www.gnu.org/licenses/`.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, your program’s commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.