An Introduction to Metamodelling and Graph Transformations

with eMoflon

Contents

1 Introduction **1**

2 Installation **3**

2.1 Install Our Plugin for Enterprise Architect (EA) 3

2.2 Install Our Plugin for Eclipse 4

2.3 Get a Simple Demo Running 5

2.4 Validate Your Installation with JUnit 7

2.5 Project Structure and Setup 8

3 Modelling a Memory Box **13**

Chapter 1

Introduction

This tutorial has been engineered to be *fun*.

If you work through it and, for some reason, do *not* have a resounding "I-Rule" feeling afterwards, please send us an email and tell us how to improve it: `contact@moflon.org`

Figure 1.1: How you should feel when you're done.

To enjoy the experience, you should be fairly comfortable with Java or a comparable object-oriented langauge, and know how to perform basic tasks in Eclipse. Although we assume this, we give references to help bring you up to speed as necessary. Last but not least, very basic knowledge of common UML notation would be helpful.

Our goal is to give a *hands-on* introduction to metamodelling and graph transformations using our tool *eMoflon*. The idea is to *learn by doing* and all concepts are introduced while working on a concrete example. The language and style used throughout is intentionally relaxed and non-academic. For those of you interested in further details and the mature formalism of graph transformations, we give relevant references throughout the tutorial.

The tutorial is divided into two main parts: In the first part (Chapter 2), we provide a very simple example and a few JUnit tests to test the installation and configuration of eMoflon.

After working through this chapter, you should have an installed and tested eMoflon working for a trivial example. We also explain the general workflow and the different workspaces involved.

In the second part of the tutorial (Chapter 3), we go, step-by-step, through a more realistic example that showcases most of the features we currently support.

Working through this chapter should serve as a basic introduction to model-driven engineering, metamodelling and graph transformations.

One last thing – at the moment we unfortunately only support Windows. This should hopefully change in future releases.

That's it – sit back, relax, grab a coffee and enjoy the ride!

Chapter 2

Installation

2.1 Install Our Plugin for Enterprise Architect (EA)

- Download and install EA (Fig. 2.1)

  Go to `http://www.sparxsystems.com.au/` to get a free 30 day trial.

Figure 2.1: Download Enterprise Architect

- Install our EA-Plugin (Fig. 2.2)

  Download zip from `http://www.moflon.org/fileadmin/download/moflon-ide/eclipse-plugin/ea-ecore-addin/ea-ecore-addin.zip`, unpack, and run setup.exe

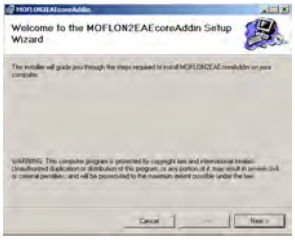

Figure 2.2: Install our plugin for EA.

2.2 Install Our Plugin for Eclipse

- Download and install Eclipse for Modeling "Eclipse Modeling Tools (includes Incubating components)" (Fig. 2.3) from: `http://www.eclipse.org/downloads/packages/`

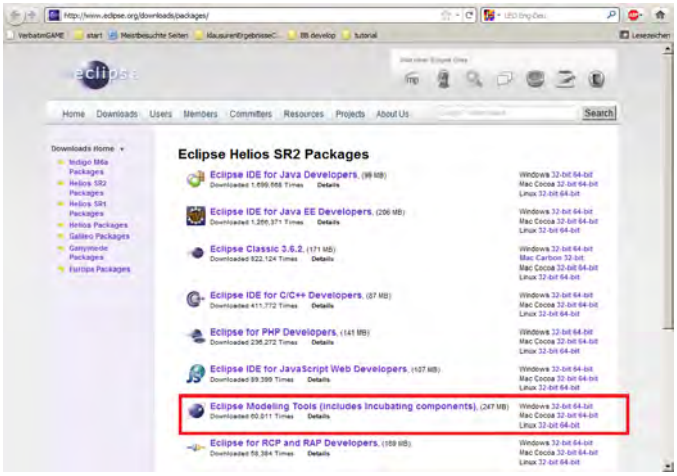

Figure 2.3: Download Eclipse Modeling Tools.

- Install our Eclipse Plugin from the following update site[1] [2]: `http://www.moflon.org/fileadmin/download/moflon-ide/eclipse-plugin/update-site2`

[1] For a detailed tutorial on how to install Eclipse and Eclipse Plugins please refer to `http://www.vogella.de/articles/Eclipse/article.html`

[2] Please note: Calculating requirements and dependencies when installing the plugin might take quite a while depending on your internet connection.

2.3 Get a Simple Demo Running

- Go to "Window/Open Perspective/Other..." and choose Moflon (Fig. 2.4).

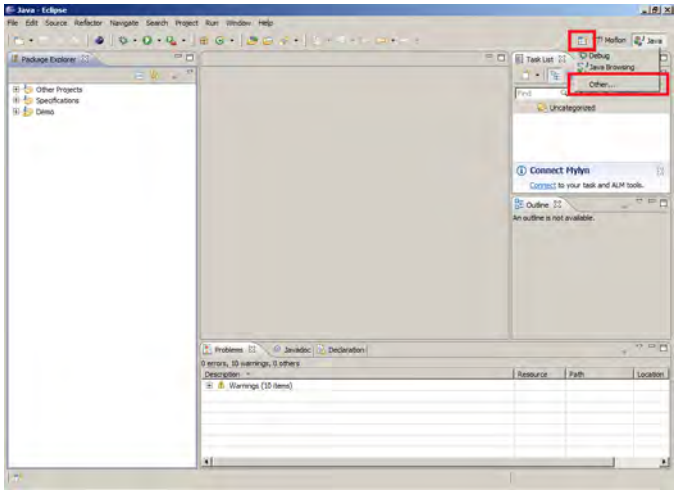

Figure 2.4: Choose the Moflon Perspective.

- In the toolbar a new action set should have appeared. Choose "New Metamodel" (Fig. 2.5). The button with an "L" shows you our logfile (important input for us if something goes wrong!).

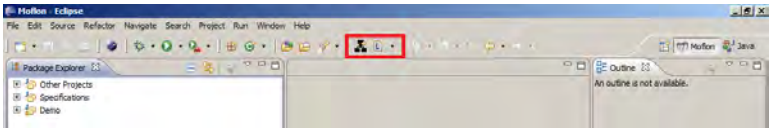

Figure 2.5: Eclipse New Metamodel

- Enter "Demo" as the name of the new metamodel project and confirm. An empty EA project file "Demo.eap" will be created in a new project with a certain project structure according to our conventions.

- Choose working sets as your top level element in the package explorer (Fig. 2.6). We work a lot with working sets and use them to structure the workspace in Eclipse.

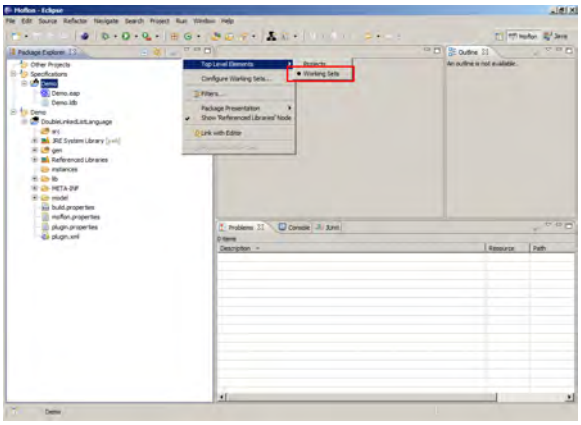

Figure 2.6: Choose Working Sets as Top Level Elements.

- Open the newly created project and replace the "Demo.eap" file with the Demo.eap that you will find in the same folder as this tutorial. This EA file already contains our simple demo project.
- Double click "Demo.eap" to start EA. Please choose "Ultimate" when starting EA for the first time.
- In EA, choose "Add-Ins/MOFLON::Ecore Addin/Export all to Workspace" (Fig. 2.7).

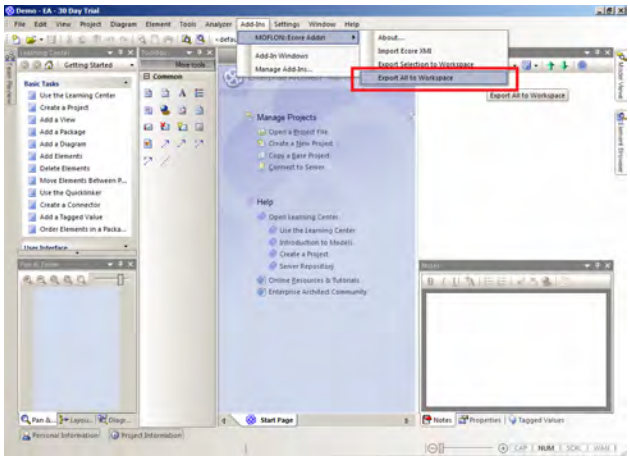

Figure 2.7: Export from EA with our plugin.

- Switch back to Eclipse, choose your Metamodel project and press F5 to refresh. The export from EA places all required files in a hidden folder in the project, and refreshing triggers a build process that invokes our different code generators automatically. You should be able to monitor the progress in the lower right corner (Fig. 2.8). Pressing the symbol opens a monitor view that gives more details of the build process.

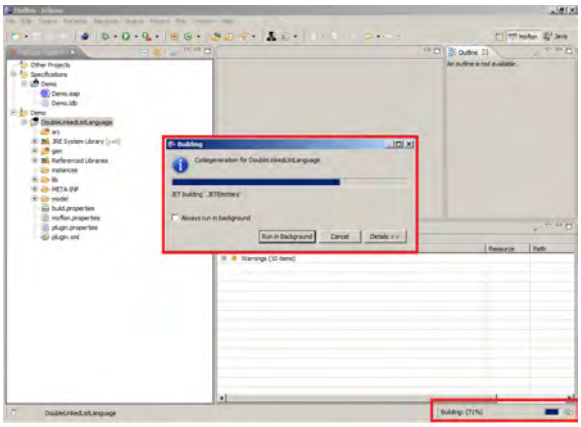

Figure 2.8: Automatically building the workspace after a refresh.

2.4 Validate Your Installation with JUnit

- Go to "File/Import/General/Existing Projects into Workspace" (Fig. 2.9) and choose the Testsuite project that is also in the same folder as this tutorial.

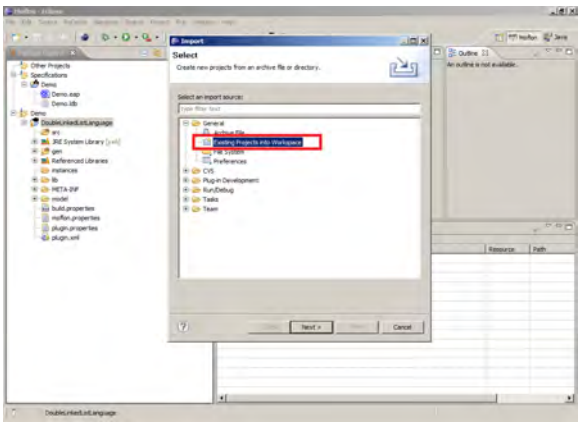

Figure 2.9: Import our Testsuite as an existing project.

At this point, your workspace should resemble Fig. 2.10.

Figure 2.10: Workspace in Eclipse.

- Right-click on the Testsuite project and select "Run as/JUnit Test". Congratulations! If you see a green bar (Fig. 2.11), then everything has been set-up correctly and you are now ready to start metamodelling!

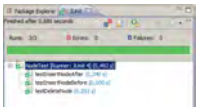

Figure 2.11: All's well that ends well...

2.5 Project Structure and Setup

Now that everything is installed and setup properly, let's take a closer look at the different workspaces and workflow. Before we continue, please make a few slight adjustments to EA so you can easily compare your current workspace to our screenshots:

- Select "Tools/Options/Standard Colors" in EA, and set your colours to reflect Fig. 2.12.
- In the same dialogue, select "Diagram/Appearance" and reflect the settings in Fig. 2.13.
- Last but not least, and still in the same dialogue, select "Source Code Engineering" and be sure to choose "Ecore" as the default language for code generation (Fig. 2.14).

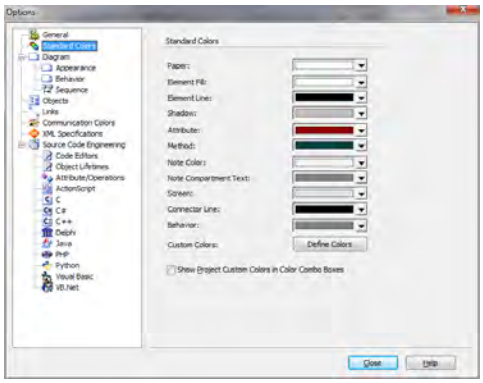

Figure 2.12: Our choice of standard colours for diagrams in EA.

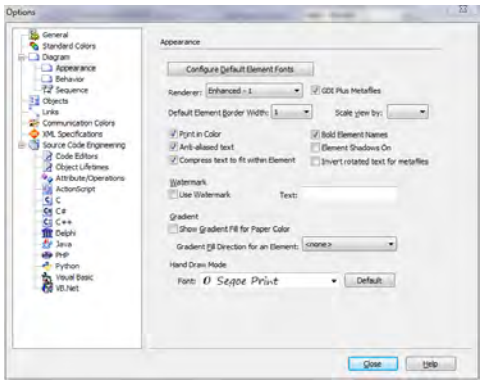

Figure 2.13: Our choice of the standard appearance for model elements in EA.

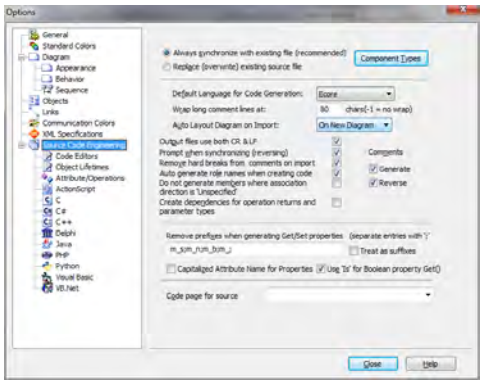

Figure 2.14: Make sure you set the standard language to Ecore.

In your EA "workspace", actually referred to as an *EA project*[3], take a careful look at the project structure: The root node `Demo`[4] is called a *model* in EA lingo and is used as a container to group a set of related *packages*. In our case, `Demo` consists of a single package DoubleLinkedListLanguage. An EA project can however consist of numerous models that in turn group numerous packages.

Now switch to your *Eclipse workspace* and note the two nodes named `Specifications` and `Demo`. These nodes, used to group related *projects* in an Eclipse workspace, are called *working sets*. The working set `Specifications` contains all *metamodel projects* in a workspace. A metamodel project contains a single EAP (EA project) file and is used to communicate with EA and initiate codegeneration by pressing F5 choosing "refresh" from the context menu. In our case, `Specifications` should contain a single metamodel project `Demo` containing our EA project file `Demo.eap`.

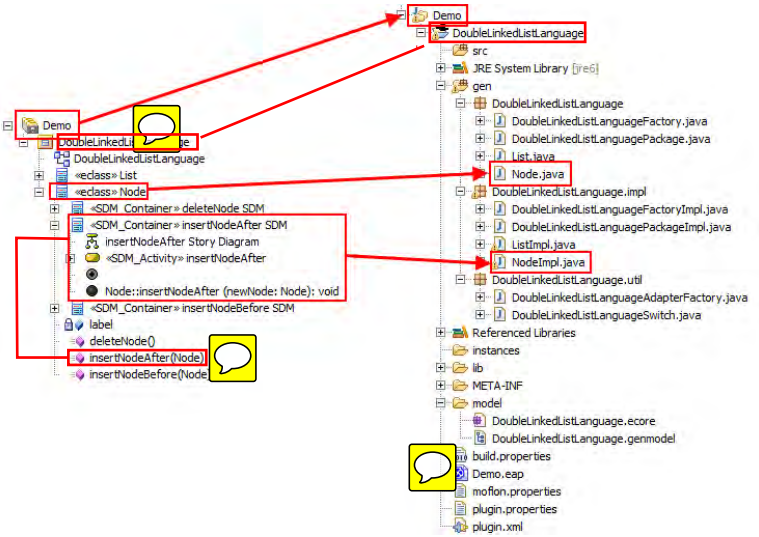

Figure 2.15: From EA to Eclipse

Figure 2.15 depicts how the Eclipse working set `Demo` and its contents were generated from the EA model `Demo`.

[3]Words are set in italics when they represent concepts that are introduced or defined in the corresponding paragraph for the first time.

[4]Words set in a `mono-space font` refer to things that you should find in a tool, dialogue, figure or code.

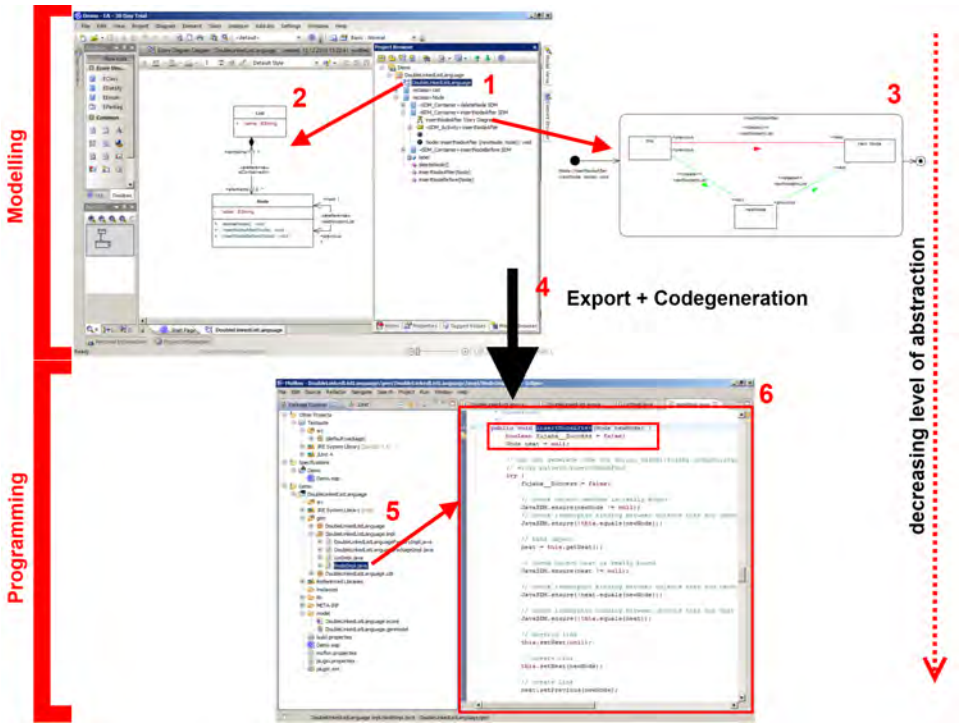

Figure 2.16: Overview

Chapter 3

Modelling a Memory Box