An Introduction to Metamodelling and Graph Transformations

with eMoflon

Part 0: Introduction

For eMoflon Version 2.3.0

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

The eMoflon team
Darmstadt, Germany (October 2015)

Part 0:

Introduction

This handbook 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 at contact@emoflon.org.

`URL of this document: http://tiny.cc/emoflon-rel-handbook/part0.pdf`

Figure 0.1: How you should feel when you're done

To enjoy the experience, you should be fairly comfortable with Java or a comparable object-oriented language, 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, 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.

So, what is eMoflon?

eMoflon is a tool for building tools. If you wish, a "meta" tool. This means that if you're interested in building *domain-specific* tools for end users, then eMoflon could be pretty useful for you.

Why should I be interested?

To build a tool, you typically need a way for users to communicate with it, i.e., you must establish a suitable *language* for specifying input and output to and from the tool. You also need a central data structure to represent the "state" of the tool. This data structure or *model* is usually manipulated and appropriately *transformed* in some useful manner by the tool. Many tools also *generate* something useful from their internal models and keep them synchronized with other models in different tools. To achieve these goals you can use *metamodelling* to define your language (your *meta*model), *graph transformations* to transform your models, and parsing/code generation techniques to produce something useful from your models. All this and much more is supported by eMoflon. Take a look at Fig. 0.2 to see how all these tasks fit together.

What does this handbook cover?

On the last page, we've described each of the 6 parts that make up this handbook. You can work through them sequentially and become an *official*[1] eMoflon master or, depending on your interests, decide what you'd like to read and what to skip. We provide all the necessary materials (i.e., a cheat package) so you can jump right in without having to complete the previous parts. For those of you interested in further details and the mature formalism of graph transformations, we give relevant references throughout the handbook.

[1] Certificate not guaranteed

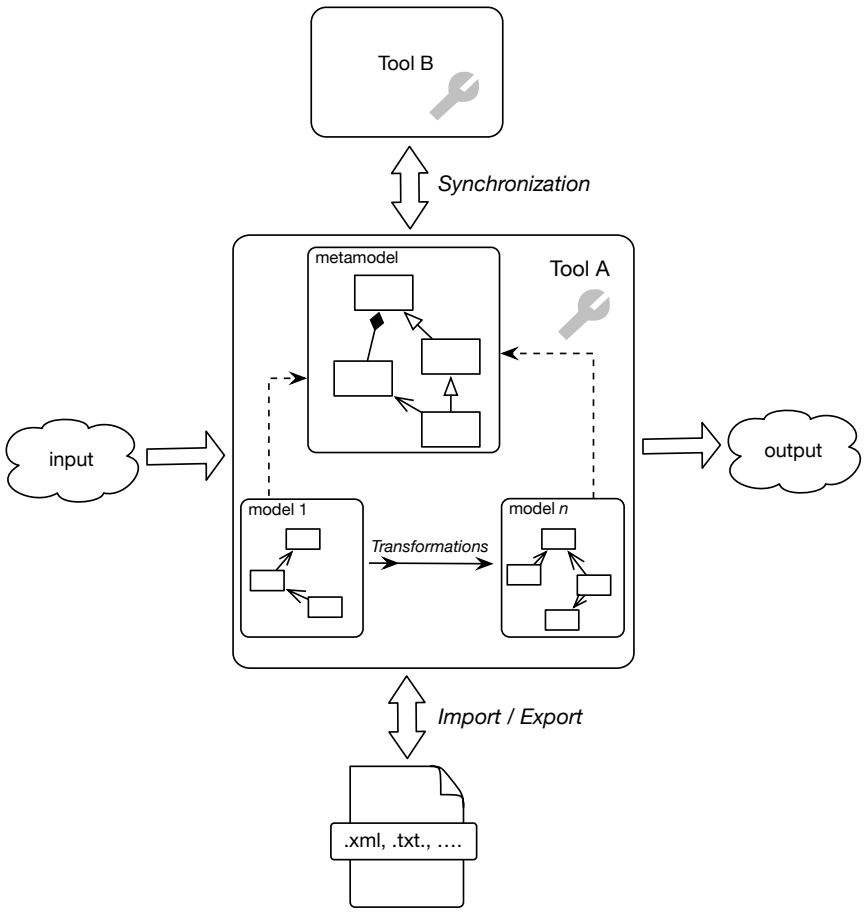

Figure 0.2: Building a tool requires language specification (metamodelling), transformations, synchronization, parsing, and code generation

Part I: Installation and set up provides a very simple example and a few JUnit tests to test the installation and configuration of eMoflon. We also explain the general workflow and the different workspaces involved.

This part can be considered *mandatory* only if you are new to eMoflon, but we recommend working through it anyway.

File download: `http://tiny.cc/emoflon-rel-handbook/part1.pdf`

Part II: Ecore takes you step-by-step through a more realistic example that showcases many of the features we currently support. Working through this part should serve as a basic introduction to model-driven engineering, and is especially recommended if you're new to metamodelling (using Ecore and the Eclipse Modeling Framework (EMF)).

File download: `http://tiny.cc/emoflon-rel-handbook/part2.pdf`

Part III: Story Driven Modelling (SDM) introduces *unidirectional* model transformation via programmed graph transformation using story diagrams.

File download: `http://tiny.cc/emoflon-rel-handbook/part3.pdf`

Part IV: TGGs introduces *bidirectional* model transformation with Triple Graph Grammars (TGGs).

File download: `http://tiny.cc/emoflon-rel-handbook/part4.pdf`

Part V: Model-to-Text Transformations shows how standard parsing and code generation technology can be combined with story diagrams and TGGs.

File download: `http://tiny.cc/emoflon-rel-handbook/part5.pdf`

Part VI: Miscellaneous contains a collection of tips and tricks to keep on hand while using eMoflon. This can be used as a reference to help avoid common mistakes and increase productivity. If you're in a hurry, this part can be skipped and consulted only on demand.

File download: `http://tiny.cc/emoflon-rel-handbook/part6.pdf`

Well, that's it! Download Part I, grab a coffee, and enjoy the ride!