Faculty of Computer Science Institute of Software and Multimedia Technology, Software Technology Group

WS2020/21 – Model-driven Software Development in Technical Spaces

Metamodeling with the Eclipse Modeling Framework

Professor: Prof. Dr. Uwe Aßmann
Tutor: Markus Hamann

1 Model-driven Toolchain - Metamodel

The goal of the exercise is to give you experience in some of the tools used to realize the concepts explained in the lecture. To archive this goal, the exercise is separated into three parts: *Model-driven Toolchains*, *Grammars*, and *Role Modeling*. In the first part, we want to create a model-driven toolchain for the creation and the further processing of simple statecharts. To do that, we will implement Editors (textual and graphical), Code Generators, and Transformation Tools. A free and powerful framework to do that is the **Eclipse Modeling Framework (EMF)** [1] and its rich collection of internal and external tools.

1.1 Eclipse Modeling Framework (EMF)

Most of the tools we will use need a *Metamodel* providing the terms and syntax for our statechart models. This exercise provides a small introduction in metamodeling within the **Eclipse Modeling Framework**.

To start with EMF, take the following steps:

- Download the *Eclipse Modeling Tools*[1], which is a prepackaged version of Eclipse including EMF.
- Familiarize yourself with EMF. You can do that by reading through this comprehensive tutorial on the use of EMF.[2]

[1]https://www.eclipse.org/downloads/packages/release/2020-09/r/eclipse-modeling-tools
[2]http://www.vogella.com/tutorials/EclipseEMF/article.html

1.2 Statechart Metamodel

The main task of the exercise is to create a metamodel of our statecharts using the EMF framework and afterwards creating multiple statechart models to test the metamodel, and prepare for the next exercises.

To do so, complete the following subtasks:

- Create a *Ecore Modeling Project* for your statechart metamodel.
- Model your Metamodel in the `*.ecore` file. Use the editor that should open on project creation.
- Create the *Model*, *Edit*, and *Editor* Implementation Code through the `*.genmodel` file. This will generate a simple Metamodel and Editor Plugin for Eclipse.
- Rightclick on the model project and choose *Run As* and *Eclipse Application.* A new Eclipse application will open. There, create a new general *Project* and a `*.statechart` file. You should now be able to model your statechart with a simple editor.

The statechart is descriped as following:
Each statechart has an identifier and multiple states. There are three types of states. First, the mandatory Initial State, which is the starting point of the statechart. Second, there are multiple Normal States. And finally, there can be multiple optional Final States, which are ending points of the chart. Each state has a name as an identifier. The normal state can also have an optional entry, exit, and/or do Activity defined by a name. Two states can be connected by a directed Transition. These transitions can have an event, a guard expression, and an action. Each of these can be missing.

The metamodel must be created as a `*.ecore` file and instances as `*.statechart` files, respectively. If you wish a review of your solution, please hand in the files till the day before the next exercise (E-Mail: `Markus.Hamann1@tu-dresden.de`). Also, on the day of the next exercise, an example solution will be published.

1.3 (Optional) A Metamodel Based on Your Project

We want to encourage you to create your own model-driven toolchain. To do that, please think of a personal project that could need an individual tool pipeline, like an editor and code generator. Try to create a metamodel based on that project.

References

[1] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: eclipse modeling framework.* Pearson Education, 2008.