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

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

Model-to-Text Transformations

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

1 Model-to-Text Transformation with Acceleo

After exploring ways to create and edit models in the last two exercises, we will now look at ways to process our models. The purpose of this exercise is to understand how to realize a *Model-to-Text* (M2T) transformation. *Model-to-Text* transformations are often used to generate code fragments of different programming languages. Other usages are the generation of documentation or the transfer of models into another technical space (like the XML or grammar space)

This exercise focuses on template-based code generation utilizing *Acceleo*[1] [1] to generate *Java* source code from our *Statechart* models, defined in the previous exercises.

1.1 Acceleo

- Install *Acceleo 3.7* from the *Eclipse Marketplace*.
- Open a eclipse workspace with your statechart metamodel projects (Metamodel, Edit, and Editor) or import them in a new workspace. *Run* the Metamodel Project *as* an *Eclipse Application* to register your metamodel in Eclipse. In the new Eclipse instance you can start using Acceleo.
- Create a new *Acceleo Project* like it is described in this tutorial[2]. The same tutorial also explains how to write an Acceleo template that can be used to transform models into code. Use your *Statechart Metamodel* from Exercise 1 for the project creation.

[1] https://www.eclipse.org/acceleo/
[2] https://wiki.eclipse.org/Acceleo/Getting_Started

1.2 Statechart-to-Java Transformation

Write an *Acceleo* template to generate *Java* code from your *Statechart* models. You can, for example, use the models created in Exercise 1 or 3.

Please follow these hints.

- Use the *State Design Pattern* with inner classes to translate your statecharts into java code.
- You do not need to translate the *guard* or *action* of a *Transition* to valid java code. You can just use them like they are in the statechart models or ignore them. The same goes for the *Activities* in a *State*.

If you wish a review of your solution, please hand in the *Acceleo* project as a `*.zip` file on the day before the next exercise (E-Mail: `Markus.Hamann1@tu-dresden.de`). If you did not hand in your metamodel after Exercise 1, you need to hand it in too. As always, on the day of the next exercise, an example solution will be published.

1.3 (Optional) A Code/Documentation Generator Based on Your Project

Like in the last exercise, we want to encourage you to create your own model-driven toolchain. If you created a metamodel from one of your projects, try to create a Model to Text generator for it.

1.4 Additional Information

- *Acceleo*,[1] is a pragmatic implementation of the Object Management Group (OMG) MOF Model-to-Text Language (MTL) standard.
- *Acceleo Getting Started*,[2] is a basic tutorial on the use of Acceleo.

References

[1] Jonathan Musset, Étienne Juliot, Stéphane Lacrampe, William Piers, Cédric Brun, Laurent Goubet, Yvan Lussaud, and Freddy Allilaire. Acceleo user guide. *Acceleo*, 2, 2006.