Instance Based Meta-Model Generation

Simon Van Laerhoven

Abstract

Model-driven engineering aims to improve the efficiency of the software development process. Domain-specific modelling languages (DSMLs) are used to develop models and are specified through meta-models. Domain experts, who have the knowledge about the domain usually don't have the skills needed to develop a correct meta-model, while software engineers, who do have the needed skills, don't have enough knowledge about the specific domain. This makes close interaction between software engineers and domain experts necessary which slows the development process. This is where the need for instance based meta-model generation rises. In this paper we propose a way to create sample models using a generic meta-model in AToMPM. With model transformations these samples will be used to automatically generate a meta-model to which all the samples conform.

Keywords: Meta-Model, Model, Example-driven modelling, Exploratory modelling

1. Introduction

Model-driven engineering (MDE) has gained a lot of populairty over the years. It is a higher level form of software development which allows domain experts, with very limited knowledge of programming, to create software applications. Before a domain expert can start modelling, a domain specific modelling language (DSML) has to be defined to match the problem domain and constrain the user. A DSML is defined by a meta-model. Domain experts, who have the knowledge about the domain usually don't have the skills needed to develop a correct meta-model, while software engineers, who do have the needed skills, don't have enough knowledge about the specific domain. This makes close interaction between software engineers and domain experts necessary which slows the development process. Instance based meta-model generation is a possible solution for this problem. The

idea is that with multiple sample models, created by the domain expert, a meta-model is automatically generated. All the sample models will conform to the created meta-model. With this the need for close interaction with the software engineer decreases drastically, although the software engineer might still be usefull for some of the steps in the meta-model generation process as we will discuss later.
In this paper we will propose a method for meta-model generation in AToMPM (A Tool for Multi-Paradigm Modelling) (Syriani et al., 2013).

2. Related Work

The following papers are somewhat related to the problem of Instance based meta-model generation. But they don't provide much help except that they give context to the problem at hand.

2.1. Generating meta-model-based freehand editors

Minas (2007) talks about freehand editing of meta-model-based models as opposed to structured editing. Most tools only provide structured editing of models, where all the allowed operations transform a correct model into another correct model. Freehand editing allows editing of a model without any restrictions, giving more freedom to the user. They combine meta-model-based models with freehand editing in the tool DiaMeta.

2.2. Example-driven meta-model development

Example-driven meta-model development by López-Fernández et al. (2015) is the most relevant paper in our case. They propose an approach where a meta-model is generated from model fragments which are specified using the tools Dia or yED. The meta-model is iteratively updated using the model fragments and refactoring by the user. A virtual assistant proposes some refactorings to the user during the process. In the end the meta-model is validated by the domain expert by again checking example models against the meta-model.

2.3. Automated Model-to-Metamodel Transformations Based on the Concepts of Deep Instantiation

In Kainz et al. (2011) they propose an automated model to meta-model transformation in the Eclipse Modelling Framework (EMF). The tranformation is divided in phases and based on the concept of deep instantiation.

Deep instantiation allows elements in a model to be both a class and an objects (clabjects). The result of the different processing phases will be a complete definition of the meta-model.

2.4. Supporting constructive and exploratory modes of modeling in multi-level ontologies

Atkinson et al. (2011) talks about the differences between constructive and exploratory modelling. Constructive modelling aims to create a complete, definitive description of all the types in a system. Building a meta-model can be seen as constructive modelling as it completely describes all types system. Exploratory modelling, on the other hand, aims to develop the types that characterize the objects in a domain. Instance Based Meta-Model Generation can then be seen as a form of exploratory modelling.

3. Model to Meta-Model Transformations in AToMPM

In this section we will propose a method to transform a collection of sample models into a meta-model which all sample models conform to. The tool we work with is AToMPM (Syriani et al., 2013). It allows us to create meta-models, models and model tranformations with a visual syntax. The process will be executed in 7 steps.

3.1. Generic Meta-Model

First we need a generic meta model which we will use to model the example models with. Figure 1 represents the abstract syntac of our generic meta-model. There are four classes: Model, ModelElement, Class and Association. A Model object represents one sample model and contains zero or more ModelElements. A ModelElement is an abstract class which has as attributes *type* and *attributes*. Both Class and Association are instances of ModelElement. An Association object has exactly one incoming and one outgoing edge to and from a Class object. Figure 2 shows how we will represent our generic meta-model.

3.2. Create Example Models

With our new generic meta-model a domain expert can create a number of sample models for the domain they desire. These models should represent the most important, if not all, use cases of the desired DSML. An example of two very simple use cases in a DSML similar to that of Petri Nets is shown in Figure 3.

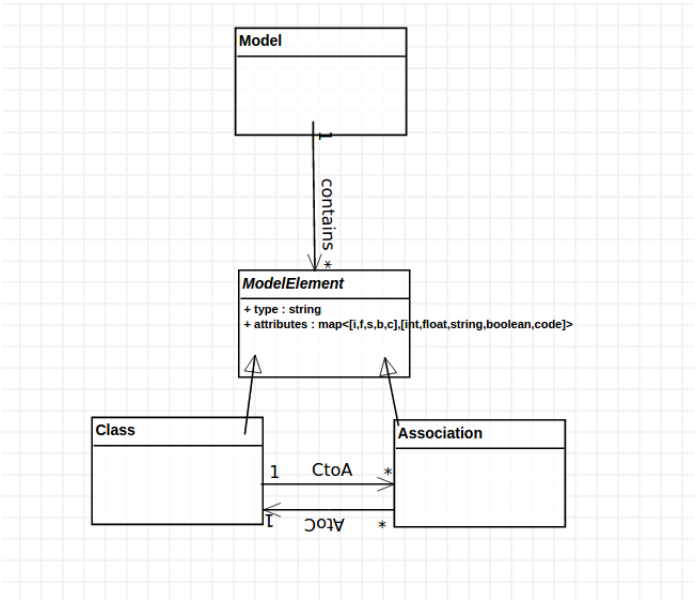

Figure 1: Generic Abstract Syntax

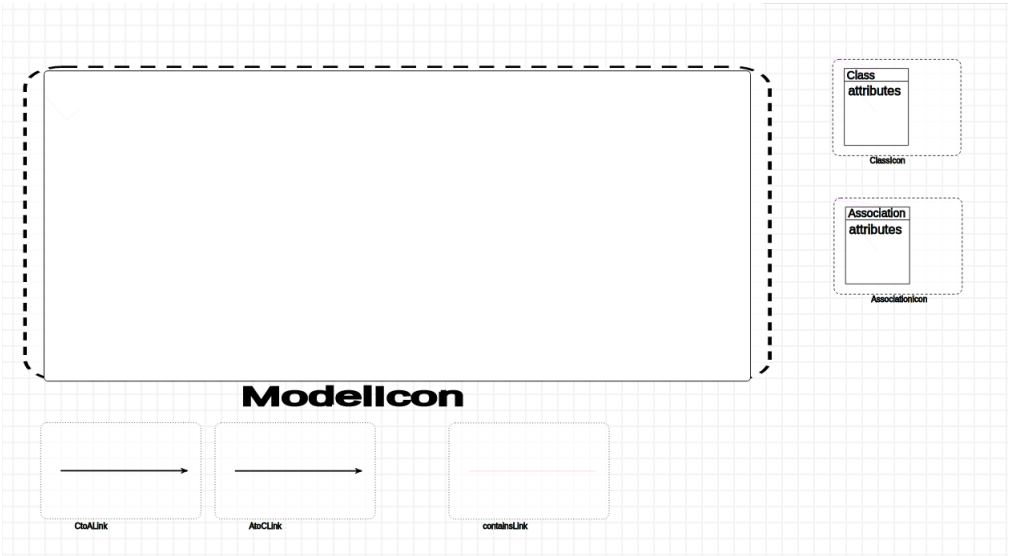

Figure 2: Generic Concrete Syntax

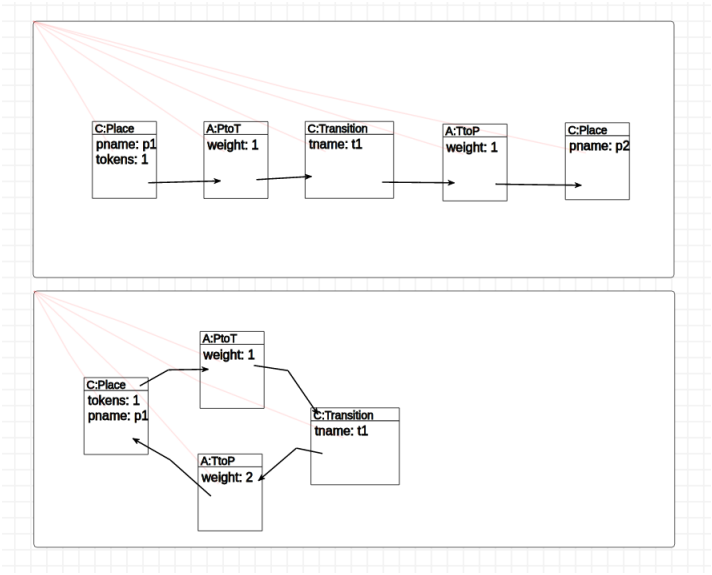

Figure 3: Two simple examples in a PN-like language

3.3. Model transformations into abstract syntax

Through the AToMPM model transformations we should be able to automatically generate a meta-model describing the language of the sample models.

3.4. Manually check abstract syntax

At this stage the meta-model could be checked to see if it looks correct. This can be done by a software engineer checking the meta-model. For this again a short interaction between a software engineer and domain expert is needed. This step is optional because later in the process the meta-model can be checked by creating new models, which the domain expert considers correct, and trying to verify if they conform to the meta-model.

3.5. Default concrete syntax generation

From our newly generated meta-model, we will generate a default concrete syntax. A rectangle to represent an element and an arrow to connect two elements representing an association.

3.6. Manually change concrete syntax icons

The domain expert then needs to change to concrete symbols to the layout that they had in mind.

3.7. Create models using newly generated MM

The final step is using the newly generated meta-model to create models. These models should be used to verify the meta-model. If a created model is correct according to the domain expert but is not accepted as a valid instance of the meta-model, then the meta-model needs to be adapted. A simple way of adapting the meta-model is by adding this correct but invalid model as a sample in the set of samples used to generate the meta-model. The newest generated meta-model should then, by definition, see this model as valid.

4. Conclusion

We have stated the need for Instance Based Meta-Model Generation. We aim to further improve the efficiency of the development process by decreasing the work for software engineers. We presented some related work about the topic, which was never directly related to our work, but helped us to put it into context.

5. References

Atkinson, C., Kennel, B., Goß, B., 2011. Supporting constructive and exploratory modes of modeling in multi-level ontologies. In: Procs. 7th Int. Workshop on Semantic Web Enabled Software Engineering, Bonn (October 24, 2011).

Kainz, G., Buckl, C., Knoll, A., 2011. Automated Model-to-Metamodel Transformations Based on the Concepts of Deep Instantiation. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 17–31.
URL http://dx.doi.org/10.1007/978-3-642-24485-8$_3$

López-Fernández, J. J., Cuadrado, J. S., Guerra, E., de Lara, J., 2015. Example-driven meta-model development. Software & Systems Modeling 14 (4), 1323–1347.
URL http://dx.doi.org/10.1007/s10270-013-0392-y

Minas, M., 2007. Generating meta-model-based freehand editors. Electronic Communications of the EASST 1.

Syriani, E., Vangheluwe, H., Mannadiar, R., Hansen, C., Van Mierlo, S., Ergin, H., 2013. Atompm: A web-based modeling environment. In: Demos/Posters/StudentResearch@ MoDELS. Citeseer, pp. 21–25.