FTG+PM: An Integrated Framework for Investigating Model Transformation Chains

Levi Lúcio¹, Sadaf Mustafiz¹, Joachim Denil²,¹, Hans Vangheluwe²,¹, and Maris Jukss¹

¹ School of Computer Science, McGill University, Canada
{levi,sadaf,joachim.denil,hv,mjukss}@cs.mcgill.ca
² University of Antwerp, Belgium
{joachim.denil,hans.vangheluwe}@ua.ac.be

Abstract. In this paper, we describe our ongoing work on model transformation chains. Model transformation chains refer to the sequences of model transformations in Model Driven Engineering (MDE). The transformations represent and formalise typical model/software engineering activities, and their chaining is the natural composition of such activities. Model transformation chains found in industrial practice vary widely, depending on the specific domain they are used in. By explicitly modelling development activities, these activities can be analysed and the MDE process may be improved. As a step towards such analyses, we propose an integrated framework to describe all the artifacts involved in model transformation chains, as well as the means to execute “enact” those chains. We describe the Formalism Transformation Graph + Process Model (FTG+PM) which is at the heart of our framework in detail.

1 Introduction

Model Driven Engineering (MDE) is currently the mainstream top-down approach to software development. The philosophy behind MDE is that software development should start by building domain specific structural and behavioral models of the system under development. By domain specific we mean that initially models of the system should be described in a language close to the domain being tackled. During the software development process those models are then improved, augmented and refined by the application of model transformations – possibly with the automatic or manual injection of additional information.

Model transformations have been called the heart and soul of MDE [1]. Chaining model transformations is a natural step in MDE as such chains allow describing the composition of activities in software construction and provide explicit means for MDE automation. However, to the best of our knowledge little work is devoted to understanding the underlying structure of such chains when they are used in domain specific software development. This work is crucial for the following (non-exhaustive) list of reasons:

– Reuse: Model transformation chains are typically devoted to building software within certain domains. In this paper, we provide an example of the
usage of model transformation chains for building automotive software. As in traditional software development, the modularity and possibility of reuse of such chains is extremely relevant from an engineering viewpoint. It seems natural that subsets of a transformation chain developed for a given software engineering purpose can be reused without much changes for a similar engineering purpose. Moreover, by identifying and classifying subsets of transformation chains responsible for high level activities in domain specific software development (e.g. requirements development, domain-specific design, verification, simulation, analysis, calibration, deployment, code generation, execution, etc), it is possible to achieve a finer level of understanding and control of such activities – in a domain specific or in a more general context;

– **Traceability**: Traceability is increasingly required in software development at the stakeholder level (e.g. to ensure a given requirement has been implemented in the system), but also at the software development level (e.g. to ensure traceability as high level models are refined along the development process). Because transformation chains explicitly model the relations between the several steps of an MDE process, traceability is a natural consequence of using such chains;

– **Certification**: Finally, and possibly most importantly, by having an explicit representation of such transformation chains and the models (and metamodels) they work on, the certification of such processes becomes possible. In certain domains such as embedded systems, automotive or aerospace, strict norms exist to ensure each step in software production is performed correctly and is properly documented. A large effort has been devoted in the last two decades to developing verification methods for software. The MDE community is now missing studies on how and when those techniques should be applied, but also how they can be composed in a meaningful way. Again, model transformation chains are the ideal context to study the usage and utility of such verification methods for software certification in MDE.

Several studies such as [2;3;4;5;6;7;8], among others, have addressed model transformation chains. However, to perform an investigation on the nature and pragmatic uses of transformation chains we require an environment where all the artifacts involved in such chains are explicitly formalized, easily accessible and easily manipulated. The majority of the approaches in the literature dealing with transformation chains are concerned with automated execution. The explicit and integrated representation of all artifacts involved in model transformation chains in a way that makes them amenable to the formal study of those chains’ characteristics is typically less of a concern. In order to address this issue and to have a solid basis to study the issues mentioned above, we need a framework allowing the modelling of model transformation chains that addresses the following requirements:

1. An explicit representation of both the languages used in the model transformation chains and the relations between those languages should be provided;
2. An explicit representation of the individual model transformations should be available and the means to execute those transformations should exist;