

Constructing and Visualizing Transformation Chains

Jens von Pilgrim¹, Bert Vanhooft², Immo Schulz-Gerlach¹,
and Yolande Berbers²

¹ FernUniversität in Hagen, 58084 Hagen, Germany

{Jens.vonPilgrim, Immo.Schulz-Gerlach}@FernUni-Hagen.de

² DistriNet, K.U.Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium*
{bert.vanhooft, yolande.berbers}@cs.kuleuven.be

Abstract. Model transformations can be defined by a chain or network of sub-transformations, each fulfilling a specific task. Many intermediate models, possibly accompanied by traceability models, are thus generated before reaching the final target(s). There is a need for tools that assist the developer in managing and interpreting this growing amount of MDD artifacts. In this paper we first discuss how a transformation chain can be modeled and executed in a transformation language independent way. We then explore how the available traceability information can be used to generate suitable diagrams for all intermediate and final models. We also propose a technique to visualize all the diagrams along with their traceability information in a single view by using a 3D diagram editor. Finally, we present an example transformation chain that has been modeled, executed and visualized using our tools.

1 Introduction

Monolithic transformations, like most non-modularized software entities have some inherent problems: little reuse opportunities, bad scalability, bad separation-of-concerns, sensitivity to requirement changes, etc. A number of these problems can be solved by decomposing a transformation into a sequence of smaller sub-transformations: a transformation chain or transformation network.

Each stage of a transformation chain produces intermediate models, which means that the total number of models can become very large. In the ideal case this all happens automatically and correctly so we do not have to care much about the intermediate models; only the final target models are to be considered. Unfortunately, we often have to study and possibly modify intermediate models manually in order to get the desired result, if only for debugging purposes. For example, if we detect an unexpected structure in the final output

* Some of the described work is part of the EUREKA-ITEA MARTES project, and is partially funded by the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy, and by the Research Fund K.U.Leuven.

model we must look at the intermediate models to identify the responsible transformation. In another case we might want to optimize the results of a complex sub-transformation manually. In these cases, it may also be useful to inspect the relations between the models under study – i.e. traceability. We have identified three main problems with the scenarios sketched above.

1. Defining, executing and maintaining transformation chains is not a straightforward task. This is particularly difficult if a mixture of different transformation languages and tools is used.
2. Relying on automatic layout mechanisms to generate diagrams for intermediate models is often not satisfactory. Since layout information is neither transformed nor preserved across transformation stages, each model’s diagram can look completely different even if subsequent models have only slightly changed.
3. Trying to interpret traceability information manually is difficult since there is no good graphical representation technique for this kind of information. Therefore, this information can often not be fully exploited.

In this paper we explain our approach to construct a transformation chain and present its output to a human user in a clear and comprehensible fashion.

We have extended UNITI (Unified Transformation Infrastructure) [1], a tool for transformation chain modeling and execution, with the ability to keep track of traceability models and their relation to other models in the chain. We discuss how the output of a transformation chain – an intricate network of models connected by traceability links – can be visualized. We explain how a proper diagram layout can be produced for the generated models by leveraging traceability information. Furthermore, we propose a technique to visualize the diagrams, together with traceability links in a 3D editor.

The remainder of this paper is structured as follows. In Section 2 we give a short overview of UNITI. Since traceability plays a crucial role in transformation chains, we discuss our perspective on that subject in Section 3. In Section 4 we describe how to automatically create the layout for generated models and introduce GEF3D, the framework used for implementing a 3D diagram editor. We demonstrate our solution with an example transformation chain in Section 5 and summarize related work in Section 6. Finally, we wrap up by drawing conclusions and identifying future work in Section 7.

2 Setting Up a Transformation Chain

Currently, most transformation technologies do not offer much support for re-using and composing sub-transformations as high-level building blocks. They focus on offering good abstractions and a clear syntax for implementation. Since many transformation languages are now reaching a certain maturity, we need to start focussing on reuse and composition of transformations.

We have developed an Eclipse plugin called UNITI (Unified Transformation Infrastructure) that manages building blocks of a transformation chain. In this