Model Transformation Chains in Model-Driven Performance Engineering: Experiences and Future Research Needs

Mathias Fritzsche[1], Wasif Gilani[2], Ralf Lämmel[3] and Frédéric Jouault[4]

Abstract: We gained experiences in implementing rule based model transformations within an industrial case study called Model-Driven Performance Engineering (MDPE). Similar to other MDE scenarios, these transformations have been implemented via multiple transformation steps interconnected in an automated model transformation chain. In this short paper, we use the MDPE case study to demonstrate reasons for decomposing model transformations and discuss disadvantages in terms of execution costs. Based on these experiences, we propose, as an input for future research, an architecture to optimize decomposed model transformation chains.

1 Introduction

Early usage of model transformation approaches has generally been limited to scenarios involving a single transformation from one source model to one target model. This is notably the case for QVT — as illustrated by the list of examples from its specification [Obj08]; it is also the case for ATL [JABK08]. As more complex problems have been tackled, the number of transformations involved in a given solution has increased. Hence, more complex transformations are organized in chains (or more general forms of composite transformations) with the output of some transformations being fed as input to others. In fact, most real-world MDE scenarios seem to involve chains of multiple transformations, e.g., twelve in the interoperability scenario for business rules presented in [FABJ09], five in the interoperability scenario for code clone tools presented in [SDJ+09], and five in the scenario for performance engineering presented in [FPG+09].

In this paper, we use the latter scenario to demonstrate reasons for model-transformation chains to appear in MDE scenarios.

It is clear that modular transformations (say, transformation chains, in particular) is an established technique in the broader field of software transformation. One example is an

[1] SAP Research CEC Belfast, mathias.fritzsche@sap.com
[2] SAP Research CEC Belfast, wasif.gilani@sap.com
[3] Universität Koblenz-Landau, Germany, rlaemmel@acm.org
[4] AtlanMod Team (INRIA-EMN) - Ecole des Mines de Nantes, France, frederic.jouault@inria.fr

aspect-oriented weaver, e.g. for .NET [LC03], which will reasonably operate at the level of byte code so that it can provide flexibility with regard to the specific .NET languages that are used for components and aspects.

We observed that decomposing a transformation into a chain may make *future extensions* less costly in terms of development costs in the case of extensions. Another issue is the decoupling of different concerns by addressing them in distinct transformations. However, we also observed the extra runtime costs that merely arise from managing transformation chains. Therefore, the paper proposes an architecture of a tool for optimizing model transformation chains. This architecture deals with the specific properties of most rule based model to model transformation languages, such as ATL or QVT, compared to traditional programming languages.

The paper is organized as follows. Section 2 presents the MDE case study including a discussion of the MDPE transformation steps. Based on this, we describe future research needs to merge decomposed model transformation steps 3. Section 4 concludes the paper.

2 Industrial Model-Transformation Case Study - Model-Driven Performance Engineering

Model-Driven Performance Engineering (MDPE) [FPG+09] is an architecture to extend existing *Process Modelling Tools* [FG09] with multi-paradigm performance decision support functionality. The BPMN [Obj06] based modelling tool of the SAP NetWeaver BPM Suite [SRMS08] and the JPASS based modelling tool of the jCOM! BPM Suite are examples of Process Modelling Tools which are currently supported via MDPE. The provided performance decision support functionality answers questions like (1) Can available staff handle future business growth? (2) How many employees are needed at which point in time? (3) Which are the most sensitive resources of the process?

All questions can be answered based on discrete event simulations, e.g. the tool AnyLogic [XJ 09] and/or analytical performance analysis approaches, e.g. the FMC-QE approach [PKFR10]. In some cases (questions 2 and 3), additional optimization or sensitivity algorithms are needed to be utilized in order to guide the performance analysis.

The MDPE architecture does not re-implement the required performance analysis engines, but makes reuse of existing ones, for instance, the AnyLogic simulation engine and the analytical FMC-QE tool. Integration of different engines permits to utilize strengths of different underlying performance prediction methodologies as discussed in [PKFR10].

Concluding, MDPE needs to interconnect *n* Process Modelling Tools with *m* Performance Analysis Tools. Figure1 shows an example transformation chain realizing this interconnection. A detailed description of the chain can be found in [FG09]. In the following subsections, a summary of reasons for the decomposed transformation is provided.

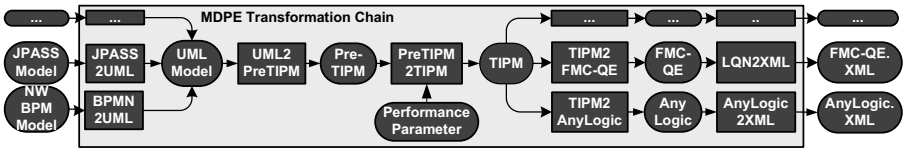

Fig. 1: MDPE Transformation chain for interconnecting two or more Process Modelling Tools with two or more Performance Analysis Tools as Block Diagram ([KGT06])

2.1 Integration of new Performance Analysis Tools

The central part of this chain is the so called Tool-Independent Performance Model (TIPM). This model particularly supports the combined use of m Performance Analysis Engines as it represents the common data base for a number of such engines, similar to the related "Core Scenario Model" in the PUMA architecture [WPP+05].

The TIPM permits the reduction of $n * m$ transformations for interconnecting chains of Process Modelling Tools with Performance Analysis Tools, into $n + m$. However, the TIPM structure is specialized for the use of Performance Analysis Tools and significantly differs from the structure of Process Modelling Languages. Therefore, UML Activity Diagrams are added to the transformation chain to simplify adaptation of MDPE for new Process Modelling Tools.

2.2 Integration of new Process Modelling Tools

The structures of most process modelling languages, such as BPMN, JPASS and UML Activity Diagrams are related, as they are close to Petri-nets. Therefore, it is sufficient to add an intermediate model into the transformation chain of Figure 1 in order to express process behaviour. We chose UML Activity Diagrams for this model due to the fact that this language is broadly used and supported by a numerous tools [FG09]. Additionally, one can apply formally defined Petri-net semantics to a subclass of UML Activity Diagrams, as discussed in more detail by Dehnert [Deh03].

We experienced that especially the transformation from BPMN to UML is close to a one to one mapping, whereas the UML to TIPM transformation is more complex as performance parameters, such as information about resource demands, sharing of resources between different process instances, etc. have to be taken into account. Thus, we would not be able to reuse the already existing complex UML to TIPM transformation every time when a new Process Modelling Tool is integrated into MDPE.

2.3 Separation of Concerns

Figure 1 shows that there is an additional transformation step between the TIPM and the generated input for the Performance Analysis Tools (see "FMC-QE.XML" and "AnyLogic.XML"). This step is caused by the fact that we have to deal with two concerns.

First, as the structural transformation concern, we were required to perform a structural transformation from the TIPM structure to the AnyLogic structure. Second, as the representation concern, the AnyLogic tool uses a specific XML format as input. Therefore, we were required to apply XML formatting so that our generated Performance Analysis Model can be read by the AnyLogic tool.

This is similar to the transformation between UML and TIPM. First, we had to translate the UML structure into the TIPM structure and, second, we had to consider Performance Parameters in order to generate a TIPM. Most of the Performance Parameters, such as number of process instances that are intended to be executed or resource demands of process steps, are simply transformed into attributes of the TIPM. However, some parameters, such as parameters about sharing of resources between different process instances, make it necessary to change the previously generated TIPM structure.

2.4 Penalties for the Transformation Chain in terms of Runtime Costs

The provided reasons for the MDPE model transformation chain can be summarized as means to reduce development effort for likely changes, such as integrating new Process Modelling Tools. A discussion of the runtime costs of this approach follows.

Due to the MDPE transformation chain we have to deal with a number of models which are unnecessary for the original task of transforming a Process Model to a Performance Analysis Model. For instance, one UML model and two different TIPMs need to be generated as intermediate models in case we execute a four-step transformation from BPMN to AnyLogic (see Figure 1). All additional models have to be stored, at least, while the transformation chain is executed. In case of a monolithic transformation approach, these intermediate models would not be required. Moreover, it is necessary to trace performance analysis results back through the model transformation chain. Hence, each transformation in the chain does not only have the direct transformation result as output but also a trace model [FJA+09], which stores information about which model element(s) are transformed to which model element(s).

For the transformation chain, we measured the memory footprint for a BPMN model with 15 process steps. The current implementation of MDPE uses file based model serialization. All transformations are implemented with ATL. For the measurements, we used a Laptop having 2GB of RAM and using a 2GHz Dual Core CPU. The BPMN model that we used as input for our measurements uses 202Kb of memory. However, only 20Kb of the data is behaviour specific and relevant for the simulation. The remaining information mainly concerns modelled rules. Additionally, we injected 47Kb of data representing Performance Parameters into the MDPE transformation chain. This data is transformed with the MDPE transformation chain to an AnyLogic input model which uses 528Kb of memory. Thus, a monolithic transformation would only require 777Kb (202+47+528) of memory to serialize the input- and output- models. However, the measured memory footprint for the chain with 2274Kb was significantly higher. Also, executing the described transformation currently consumes in average 13.6 seconds.

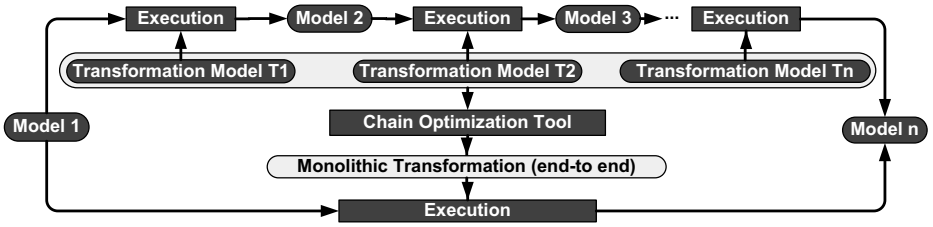

Fig. 2: Transformation chain optimization overview as Block Diagram ([KGT06])

Concluding, we paid the price of additional memory usage. We also believe that the chain significantly contributes to the high transformation execution time. Thus, we experienced the need to optimize model transformation chains to avoid high runtime costs, especially when it comes to larger MDE scenarios. In our case, all intermediate models are serialized text based. Hence, one measure to decrease the data footprint as well as performance would be to access all intermediate models in memory. Another possibility would be to develop an approach for merging decomposed model transformations before they are executed. This topic is explained as a direction for future research in the following section.

3 Proposed Future Research for Model Transformation Chains

Following the experiences that we explained in the previous section we identified the need for *Transformation Chain Optimization Tool* to merge adjacent model transformation steps before they are executed. The following Figure 2 depicts the problem space such tool is required to address. It has to take *n Transformation Models*, such as rule based ATL or QVT scripts, as input and translate them into one monolithic *End-to-End Transformation*. This end-to-end transformation needs to enable, identical to the original chain, the translation of a *Model 1*, which conforms-to a *Meta-Model 1*, into a *Model n*, which conforms-to a *Meta-Model n*.

Figure 3 shows the architecture which we propose to implement such Transformation Chain Optimization Tool. Within this architecture, Higher Order Transformations (HOTs) are employed as one of the underlying concepts. HOTs are transformations that are used to transform one Transformation Model A to a new Transformation Model $A*$. A broad set of applications for HOTs can be found in [TJF+09].

Figure 3 shows the main agents we propose to implement such a Transformation Chain Optimization Tool, namely the *Local Merge (HOT)*, *Analysis HOT* and *Global Merge Function*. The functionality of these agents is explained below.

The Local Merge HOT generates a Transformation Model called *Monolithic Transformation (T1,T2)* based on the *Transformation Model T1* and the *Transformation Model T2*. In our architecture, T1 and T2 represent adjacent transformation steps, which are defined via a rule based transformation language, such as ATL or QVT.

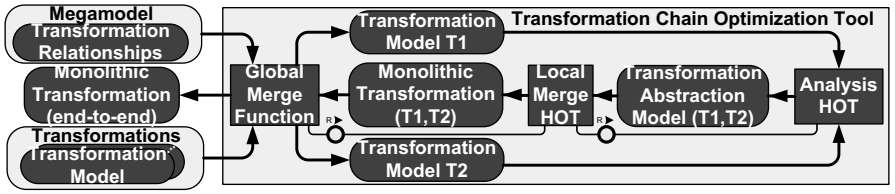

Fig. 3: Direction for Future Research as Block Diagram ([KGT06])

Rule based transformation languages, mix imperative and declarative statements [Jou05]. Therefore, monolithic transformation can not be generated by directly employing state of the art approaches for program optimization, such as deforestation [Wad88] or other kinds of optimization, such as program specialization (i.e. partial evaluation) [MWP+01].

Therefore, the Transformation Model called *Transformation Abstraction Model (T1,T2)* has been added to the Transformation Chain Optimization Tool. This model does not contain imperative statements any more and, thus, represents the pure mapping between the source and target elements of the Transformations Models T1 and T2 on the meta-level. A formal definition of this model is considered as a first topic for future research.

The Transformation Abstraction Model needs to be generated from the transformation scripts T1 and T2. So called *Analysis HOTs* [TJF+09] have already been implemented in order to analyse the input and output of model transformation on the meta-level. The similar implementation of analysis HOTs for the creation of Transformation Abstraction Models based on different rule based model transformation languages, such as ATL or QVT, is considered as a second topic for future research.

The Local Merge HOT takes the Transformation Abstraction Model as an input in order to generate the merged transformation based on T1 and T2. Due to the declarative nature of the Transformation Abstraction Model, we claim that the Local Merge HOT can be implemented more easily as existing kinds of optimization may become applicable more directly. However, as a third topic for future research, these established techniques would still need adaptation to the domain of model transformations because of the special expressivity used for models and transformations. For instance, models can be of graph shape, i.e., they use reference semantics (including aliasing) as opposed to value semantics and tree shape in classical functional programming.

The Local Merge HOT is controlled by the *Global Merge Function* agent (see "R" between the Global Merge Function and the Local Merge HOT). This agent needs to identify adjacent transformation steps. For this task, the application of a *Megamodel* is proposed.

A megamodel expresses relationships between different types of modelling artefacts, such as the definition of a transformation relationship between Transformation Models and intermediate models, which are employed as in- and output of model transformations. The Global Merge Function can iterates over the Megamodel in order to find adjacent transformations, which are then sent to the Local Merge HOT. This HOT sends the merged result back to the Global Merge Function, which then sends the next consecutive transformations

to the Local Merge HOT. Thus, the Global Merge Function works recursively as long as the *Monolithic Transformation(end-to-end)* is found.

4 Conclusions

In this paper we provided an industrial case study to demonstrate reasons for decomposed model transformations. Summarizing, such chains permit a high degree of modularization in order to reduce development effort in the case of likely changes. We also demonstrated increased execution costs in the MDPE transformation chains. Therefore, we identified the need to merge adjacent transformation steps of model transformation chains before they are executed. We proposed an architecture for rule based transformation languages, such as ATL and QVT, which permits merging model transformation chains. The architecture applies different HOTs and a so called Transformation Abstraction Model. The implementation of this architecture for different model transformation languages is proposed as an area for future research.

As another area for future research, we identified that MDE is currently lacking evaluated cost functions, which can be applied for model transformations. With such cost functions in place, we may be able to balance execution and development costs in cases where transformation chain optimizations cannot be applied, such as in the case of a distributed execution of a transformation chain.

References

[Deh03] Juliane Dehnert. *PhD Thesis: A Methodology for Workflow Modeling: From business process modeling towards sound workflow specification*. TU-Berlin, 2003.

[FABJ09] Marcos Didonet Del Fabro, Patrick Albert, Jean Bézivin, and Frédéric Jouault. In *Proceedings of the 5èmes Journées sur l'Ingénierie Dirigée par les Modèles (IDM)*, 2009.

[FG09] Mathias Fritzsche and Wasif Gilani. Model Transformation Chains to integrate Performance related Decision Support into BPM Tool Chains. In *Post-proceedings of the 3rd Summer School on Generative and Transformational Techniques in Software Engineering (GTTSE'09) (to appear)*, LNCS. Springer-Verlag, 2009.

[FJA+09] Mathias Fritzsche, Jendrik Johannes, Uwe Assmann, Simon Mitschke, Wasif Gilani, Ivor Spence, JohnBrown, and Peter Kilpatrick. Systematic Usage of Embedded Modelling Languages in Automated Model Transformation Chains. In *Proceedings of the 1st International Conference on Software Language Engineering (SLE'08), Revised Selected Papers*, volume 5452 of *LNCS*, pages 134–150. Springer-Verlag, 2009.

[FPG+09] Mathias Fritzsche, Michael Picht, Wasif Gilani, Ivor Spence, John Brown, and Peter Kilpatrick. Extending BPM Environments of your choice with Performance related Decision Support. In *Proceedings of the 7th Business Process Management Conference (BPM'09)*, volume 5701 of *LNCS*, pages 97–112. Springer-Verlag, 2009.

[JABK08] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. ATL: A model transformation tool. In *Science of Computer Programming*, volume 72, pages 31–39, 2008.

[Jou05] Frédéric Jouault. Loosely Coupled Traceability for ATL. In *Proceedings of the ECMDA workshop on traceability*, 2005.

[KGT06] Andreas Knöpfel, Bernhard Gröne, and Peter Tabeling. *Fundamental Modeling Concepts: Effective Communication of IT Systems.* John Wiley & Sons, 2006.

[LC03] Donal Lafferty and Vinny Cahill. Language-independent aspect-oriented programming. In *Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications (OOPSLA'03)*. ACM, 2003.

[MWP+01] Dylan McNamee, Jonathan Walpole, Calton Pu, Crispin Cowan, Charles Krasic, Ashvin Goel, Perry Wagle, Charles Consel, Gilles Muller, and Renauld Marlet. Specialization tools and techniques for systematic optimization of system software. *ACM Trans. Comput. Syst.*, 19(2):217–251, 2001.

[Obj06] Object Management Group. Business Process Modeling Notation Specification, Final Adopted Specification, Version 1.0., 2006.

[Obj08] Object Management Group. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Version 1.0, 2008.

[PKFR10] Tomasz Porzucek, Stephan Kluth, Mathias Fritzsche, and David Redlich. Combination of a Discrete Event Simulation and an Analytical Performance Analysis through Model-Transformations. In *Proceedings of the 17th International Conference on the Engineering of Computer-Based Systems (ECBS'10), to appear*. IEEE Computer Society, 2010.

[SDJ+09] Yu Sun, Zekai Demirezen, Frédéric Jouault, Robert Tairas, and Jeff Gray. A Model Engineering Approach to Tool Interoperability. 5452:178–187, 2009.

[SRMS08] Jim Hagemann Snabe, Ann Rosenberg, Charles Molle, and Mark Scavillo. *Business Process Management: The SAP Roadmap.* SAP Press, 2008.

[TJF+09] Massimo Tisi, Frédéric Jouault, Piero Fraternali, Stefano Ceri, and Jean Bézivin. On the Use of Higher-Order Model Transformations. In *Proceedings of the 5th European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA'09)*, volume 5562 of *LNCS*, pages 18–33. Springer-Verlag, 2009.

[Wad88] Philip Wadler. Deforestation: transforming programs to eliminate trees. In *Proceedings of the Second European Symposium on Programming*, pages 231–248, Amsterdam, The Netherlands, The Netherlands, 1988. North-Holland Publishing Co.

[WPP+05] Murray Woodside, Dorina C. Petriu, Dorin B. Petriu, Hui Shen, Toqeer Israr, and Jose Merseguer. Performance by unified model analysis (PUMA). In *Proceedings of the 5th International Workshop on Software and Performance (WOSP'05)*, pages 1–12. ACM, 2005.

[XJ 09] XJ Technologies. AnyLogic — multi-paradigm simulation software. `http://www.xjtek.com/anylogic/`, 2009.